



UNIVERSIDAD TECNOLÓGICA DE LA MIXTECA

**“ESTIMACIÓN DEL ESFUERZO EN PROYECTOS DE SCRUM
UTILIZANDO TÉCNICAS DE APRENDIZAJE
COMPUTACIONAL”**

TESIS

**PARA OBTENER EL GRADO DE
MAESTRA EN INGENIERÍA EN SOFTWARE**

PRESENTA

ING. BEATRIZ BEDOLLA MARTÍNEZ

DIRECTOR DE TESIS

DR. IVÁN ANTONIO GARCÍA PACHECO

CODIRECTOR DE TESIS

DR. RAÚL CRUZ BARBOSA

HUAJUAPAN DE LEÓN, OAXACA, MÉXICO, SEPTIEMBRE DE 2026

Tesis presentada ante los siguientes sinodales:

Dra. María Angélica Astorga Vargas

Dra. Carla Leninca Pacheco Agüero

Dra. Verónica Rodríguez López

M.T.C.A. Moisés Emmanuel Ramírez Guzmán

Director de tesis:

Dr. Iván Antonio García Pacheco

Codirector de tesis:

Dr. Raúl Cruz Barbosa

Dedicatoria

A mis padres, por su amor y motivación a lo largo de mi vida.

Agradecimientos

A mis padres y hermanos por su apoyo incondicional y motivación en cada etapa de mi formación académica.

A mis directores, Dr. Iván Antonio García Pacheco y Dr. Raúl Cruz Barbosa, por su apoyo, comprensión y paciencia durante cada etapa de esta investigación. Por compartir sus conocimientos, experiencia y valiosas observaciones, las cuales contribuyeron significativamente al desarrollo y culminación de este trabajo.

A la Universidad Tecnológica de la Mixteca por brindarme la oportunidad de crecer académica y profesionalmente, así como, a las y los profesores que contribuyeron en mi formación académica.

A mis sinodales por el tiempo dedicado en la lectura y revisión de esta investigación, por sus observaciones, recomendaciones y valiosos comentarios, los cuales contribuyeron a fortalecer la calidad de mi trabajo.

A todo el personal directivo y técnico de la organización desarrolladora de software que facilitó la realización del caso de estudio. Sin su apoyo hubiera sido imposible reportar resultados tanto cuantitativos como cualitativos de esta investigación.

A todas las personas que, de una u otra forma, formaron parte de este proceso y me brindaron su apoyo.

Finalmente, a la Secretaría de Ciencia, Humanidades, Tecnología e Innovación (SECIHTI), antes Consejo Nacional de Humanidades, Ciencias y Tecnologías (CONAHCYT), por brindarme el apoyo económico durante el periodo que comprendió la realización de mis estudios de maestría.

Índice

Índice	ix
Lista de tablas	xiii
Lista de figuras	xv
Resumen	xvii
1.Introducción.....	1
1.1. Contexto del problema.....	1
1.2. Importancia del problema.....	4
1.3. Necesidad de solución	7
1.4. Delimitaciones y limitaciones de la tesis.....	10
1.5. Hipótesis	10
1.6. Objetivos del trabajo.....	11
1.6.1. Objetivo general	11
1.6.2. Objetivos específicos.....	11
1.7. Metodología de trabajo.....	11
1.8. Estructura de la tesis.....	14
1.9. Publicaciones generadas.....	14
2.Marco teórico y estado del arte	15
2.1. Gestión de los proyectos de software	15
2.2. Metodologías para la gestión y desarrollo de un proyecto de software.....	25
2.2.1. La metodología Scrum.....	26
2.2.2. CRISP-DM	29
2.3. Estimación del esfuerzo en el desarrollo de software.....	30
2.3.1. Importancia de la estimación del esfuerzo en los proyectos ágiles	32
2.3.2. Factores que influyen en la estimación efectiva del esfuerzo	34
2.3.3. Técnicas para realizar la estimación de esfuerzo.....	36
2.3.3.1. Técnicas de sobremuestreo de minorías sintéticas para regresión	42
2.3.3.2. Métricas de evaluación de los métodos de estimación	43
2.3.3.3. Técnicas de estimación para proyectos ágiles de software.....	45
2.3.4. Problemas y retos actuales en la estimación de esfuerzo	46

2.4. Estado del arte.....	47
2.4.1. Aplicación de técnicas de cómputo suave para la modernización de la estimación del esfuerzo en los proyectos de software.....	47
2.4.1.1. Objetivo	47
2.4.1.2. Descripción	48
2.4.1.3. Resultados.....	48
2.4.2. Estimación de costo y esfuerzo en el desarrollo ágil de software utilizando las técnicas de árboles de decisión y puntos de historia	49
2.4.2.1. Objetivo	49
2.4.2.2. Descripción	49
2.4.2.3. Resultados.....	52
2.4.3. Una red neuronal LSTM optimizada para la estimación precisa del esfuerzo en el desarrollo de software.....	52
2.4.3.1. Objetivo	52
2.4.3.2. Descripción	52
2.4.3.3. Resultados.....	55
2.4.4. Conclusiones del estado del arte.....	55
3. Metodología: Implementación del método de estimación.....	59
3.1. Especificaciones de hardware y software	59
3.2. Construcción de los conjuntos de datos sintéticos.....	60
3.2.1. Metodología para el diseño y construcción de la base de datos	63
3.3. Método de estimación de esfuerzo	64
3.3.1. Preparación de los datos	65
3.3.2. Obtención del modelo de estimación.....	66
3.4. Desarrollo del prototipo de aplicación web	68
4. Resultados.....	73
4.1. Pruebas de sobremuestreo y submuestreo para la estimación	73
4.2. Creación de conjuntos de datos sintéticos	78
4.2.1. Conjunto de datos sintéticos obtenido para la estimación de proyectos de software	79
4.2.2. Conjunto de datos sintéticos obtenido para la estimación de <i>sprints</i>	81
4.3. Resultados de estimación con proyectos de software.....	83
4.3.1. Preprocesamiento.....	84
4.3.2. Búsqueda de hiperparámetros.....	84
4.3.3. Evaluación de modelos	87
4.3.4. Obtención del mejor modelo de estimación	89
4.4. Resultados de estimación con <i>sprints</i>	90
4.4.1. Preprocesamiento.....	91
4.4.2. Búsqueda de hiperparámetros.....	91
4.4.3. Evaluación de modelos	92

4.4.4. Obtención del mejor modelo de estimación	95
4.5. Aplicación web	96
4.6. Resultados del caso de estudio	101
4.6.1. Participantes	102
4.6.2. Instrumentos	104
4.6.3. Diseño experimental y proceso	105
4.6.4. Hallazgos	107
4.6.4.1. Hallazgos cuantitativos.....	107
4.6.4.2. Hallazgos cualitativos.....	113
4.6.5. Cierre del caso de estudio: Comprobación de la hipótesis de la tesis	115
5.Conclusiones y trabajo futuro.....	117
6.Bibliografía.....	119
7.Anexo A.- Acrónimos	131

Lista de tablas

Tabla 1. Factores identificados en la literatura.....	36
Tabla 2. Métricas de evaluación utilizadas para evaluar los métodos de estimación.....	44
Tabla 3. Algoritmos del aprendizaje computacional más utilizados para la estimación de esfuerzo.	46
Tabla 4. Resultados obtenidos con la evaluación de cada técnica. Nota: Traducido de Sharma y Vijayvargiya (2021).....	49
Tabla 5. Conjuntos de datos utilizado para la evaluación de las técnicas. Nota: Traducido de Iordan (2024)	53
Tabla 6. Comparativa de métricas de evaluación para cada técnica evaluada. Nota: Traducido de Iordan (2024)	56
Tabla 7. Comparativa entre estudios presentados en el estado del arte.....	57
Tabla 8. Especificaciones de hardware.....	59
Tabla 9. Terminología del Método de Estimación Generalizada. Nota: Traducido de Bahlerao e Ingle (2011)	61
Tabla 10. Parámetros ELM.....	66
Tabla 11. Parámetros MLP.....	66
Tabla 12. Parámetros KNN.	66
Tabla 13. Parámetros DT.....	67
Tabla 14. Parámetros RF.	67
Tabla 15. Parámetros SVR.	67
Tabla 16. Parámetros LR.....	67
Tabla 17. Parámetros Lasso.....	68
Tabla 18. Parámetros Ridge.	68
Tabla 19. Ensamblajes heterogéneos utilizados.	68
Tabla 20. Módulos implementados en la aplicación web.....	70
Tabla 21. Rutas relevantes para la navegación.....	71
Tabla 22. Ensamblajes utilizados.	75
Tabla 23. Intervalos de búsqueda de hiperparámetros.	76
Tabla 24. Hiperparámetros encontrados con la búsqueda fina.	76
Tabla 25. Resultados de RMSE sobre el conjunto Desharnais.	77

Tabla 26. Comparativa de métricas por cada conjunto de datos.	78
Tabla 27. Variables del conjunto de datos sintéticos obtenido para proyectos de software.....	80
Tabla 28. Terminología de GEM. Nota: Traducido de Ramessur y Nagowah (2021).	82
Tabla 29. Intervalos de búsqueda de hiperparámetros.....	85
Tabla 30. Intervalos de búsqueda de hiperparámetros para búsqueda fina.	86
Tabla 31. Hiperparámetros encontrados con la búsqueda fina.	86
Tabla 32. Resultados de MAE sobre el conjunto de datos para proyectos de software.	87
Tabla 33. Resultados de RMSE sobre el conjunto de datos para proyectos de software.	88
Tabla 34. Resultados de MMRE sobre el conjunto de datos para proyectos de software.	89
Tabla 35. Resultados de PRED sobre el conjunto de datos para proyectos de software.	90
Tabla 36. Resultados del mejor modelo.	90
Tabla 37. Intervalos de búsqueda de hiperparámetros para búsqueda fina.	92
Tabla 38. Hiperparámetros encontrados con la búsqueda fina.	92
Tabla 39. Resultados de MAE sobre el conjunto de datos para <i>sprints</i>	93
Tabla 40. Resultados de RMSE sobre el conjunto de datos para <i>sprints</i>	94
Tabla 41. Resultados de MMRE sobre el conjunto de datos para <i>sprints</i>	95
Tabla 42. Resultados de PRED sobre el conjunto de datos para <i>sprints</i>	96
Tabla 43. Resultados del mejor modelo.	97
Tabla 44. Características de la organización participante.	102
Tabla 45. Perfiles del personal participante.....	103
Tabla 46. Cuestionario para recoger las percepciones de las y los participantes en el caso de estudio.	105
Tabla 47. Estimación de <i>sprints</i> de P1 utilizando el modelo EH7.	108
Tabla 48. Estimación de <i>sprints</i> de P2 utilizando el modelo EH7.	111
Tabla 49. Percepciones de las y los participantes en el caso de estudio.....	113

Lista de figuras

Figura 1.1. Tasa de éxito de los proyectos de software en el mundo. Nota: Traducida de Johnson (2020).	2
Figura 1.2. Tasa de éxito por tamaño de proyecto. Nota: Traducida de Johnson (2020).	3
Figura 1.3. Marco de trabajo Scrum. Nota: Traducida de Schwaber y Sutherland (2020).	6
Figura 1.4. Fases y actividades realizadas para el desarrollo de la tesis.	12
Figura 2.1. Procesos de la gestión tradicional de proyectos. Nota: Traducida de Salameh (2014). ..	16
Figura 2.2. Grupo de procesos de planificación. Nota: Tomada de PMBOK (PMI, 2021).	18
Figura 2.3. Procesos de la gestión ágil de proyectos. Nota: Traducida de Salameh (2014).	24
Figura 2.4. Representación gráfica de Scrum. Nota: Traducida de Schwaber y Sutherland (2021).	26
Figura 2.5. Proceso estándar intersectorial para la Minería de Datos. Nota: Traducida de Saltz (2021).	29
Figura 2.6. Métodos y técnicas para la estimación del esfuerzo y de los costos en un proyecto de desarrollo de software. Nota: Traducida de Jadhav et al., (2022).	38
Figura 2.7. Proceso de EEE. Nota: Traducida de Hosni et al., (2016).	41
Figura 2.8. Estructura de un árbol de decisión. Nota: Traducida de Rodríguez Sánchez et al., (2023).	50
Figura 2.9. Estructura de un bosque aleatorio. Nota: Traducida de Rodríguez Sánchez et al., (2023).	51
Figura 2.10. Estructura de AdaBoost. Nota: Traducida de Rodríguez Sánchez et al., (2023).	51
Figura 3.1. Metodología utilizada en la construcción de la base de datos. Nota: Traducida de Ramessur y Nagowah (2021).	63
Figura 3.2. Diagrama general de los módulos para obtener el modelo de estimación.	64
Figura 3.3. Particionamiento del conjunto de datos.	65
Figura 3.4. Fases para el desarrollo de la aplicación web.	68
Figura 3.5. Arquitectura del sistema.	69
Figura 3.6. Arquitectura Cliente-Servidor propuesta para el desarrollo de la aplicación web.	70
Figura 4.1. Metodología utilizada para implementar el método de estimación.	74
Figura 4.2. Flujo de trabajo aplicando técnicas de sobremuestreo.	74
Figura 4.3. Distribución de Desharnais.	75
Figura 4.4. Conjunto de datos sintéticos obtenido para proyectos de software.	81

Figura 4.5. Conjunto de datos sintéticos obtenido para <i>sprints</i>	83
Figura 4.6. Distribución del conjunto de datos para proyectos de software.....	84
Figura 4.7. Distribución del conjunto de datos para <i>sprints</i>	91
Figura 4.8. Inicio de sesión.....	97
Figura 4.9. Registro de usuario.....	98
Figura 4.10. Estimación como invitado.....	98
Figura 4.11. Subir archivo a la aplicación web.	99
Figura 4.12. Estimaciones realizadas.....	99
Figura 4.13. Muestras para el archivo .CSV.....	100
Figura 4.14. Página para administrador.....	100

Resumen

La estimación o predicción del esfuerzo en el desarrollo de software es esencial para la gestión eficiente de los proyectos, ya que su precisión influye directamente en el éxito o fracaso de éstos. A pesar de los beneficios documentados, contar con estimaciones realistas y lograr precisión bajo enfoques ágiles de desarrollo, sigue siendo un desafío abierto y nicho de estudio. En entornos ágiles de desarrollo, la complejidad e incertidumbre de los proyectos aumentan debido a factores como los cambios frecuentes en los requisitos, las iteraciones cortas y la falta de detalles iniciales que terminan dificultando la obtención de una estimación precisa. Problema que debe abordarse mediante el desarrollo de propuestas que promuevan la adaptabilidad y mejora continua de la estimación, y prioricen la entrega de valor, ya que a pesar de que la literatura actual proporciona evidencia de los múltiples beneficios que otorga la obtención de estimaciones realistas, la estimación, especialmente en proyectos que adoptan enfoques ágiles de desarrollo, aún es una tarea desafiante. En este sentido, esta tesis presenta una propuesta que hace uso de modelos de aprendizaje computacional para mejorar la estimación del esfuerzo en una pequeña organización desarrolladora de software con un mínimo error relativo. Finalmente, mediante un enfoque cuantitativo/cualitativo se muestra una comprensión integral y holística de los resultados obtenidos que permiten evaluar la utilidad y percepción subjetiva de la estimación.

1. Introducción

1.1. Contexto del problema

De acuerdo con Pressman y Maxim (2019), *“la Ingeniería de Software es la aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento del software... la cual, al mismo tiempo, ofrece métodos, técnicas y herramientas para desarrollar y mantener el software”*. En este sentido, Farley (2021) argumenta que, en la actualidad, tanto el tamaño de los sistemas, su complejidad, la madurez de las herramientas y los procesos de desarrollo de software, continúan dificultando la implementación de productos de calidad. Este problema se hizo evidente a principios de 1970 cuando se identificó la “crisis del software”, lo cual condujo a la creación de una disciplina que se ocupara del diseño y construcción de los productos de software. Fitzgerald (2012), por ejemplo, describió esta crisis como todos los problemas que surgen durante el desarrollo de un proyecto para crear un producto de software. De hecho, la mayoría de las causas que originalmente se identificaron como detonadores del fracaso aún están presentes y ocasionan que estos proyectos no se completen a tiempo, aumentando así, por ejemplo, el presupuesto y el tiempo de entrega. Por lo tanto, es común que en la industria actual de software muchos proyectos de desarrollo sigan generando pérdidas millonarias, además de no cumplir con las necesidades del cliente.

Como consecuencia de lo anterior, a lo largo del tiempo y desde diferentes perspectivas se ha buscado solucionar dicha “crisis”, a tal grado de que se han desarrollado herramientas tecnológicas, compendios de prácticas, modelos, paradigmas y lenguajes de programación, entre otras propuestas, mismas que durante los años 1970 y 1990 fueron consideradas como “balas de plata” (i.e., soluciones que ayudarían a resolver dicha crisis). Sin embargo, a pesar de que ya pasaron más de 60 años del surgimiento de la “crisis del software”, aún no se ha encontrado su solución definitiva. Por lo que, la Ingeniería de Software ha enfrentado, desde su surgimiento, un reto complejo al intentar definir conjuntos de prácticas y métodos efectivos que, al aplicarse de manera correcta, permitan que los especialistas desarrollen software de calidad. Palomo y Gil (2020), por ejemplo, argumentaron que la creación de los denominados ciclos de vida del software introdujo una serie de pasos o fases que formalizaron un enfoque ordenado y sistemático para la creación del software, tales como el análisis, el diseño, la codificación, la integración y el mantenimiento. En este mismo orden de ideas, Camacho Sánchez (2023) explicó que un proceso de software es *“una colección de procedimientos, técnicas, herramientas y documentos auxiliares que ayudan a los desarrolladores de software en sus esfuerzos por implementar nuevos sistemas de información”*. En este contexto, las investigaciones actuales sobre la Ingeniería de Software hacen una distinción entre dos enfoques o metodologías muy diferentes para gestionar los procesos y proyectos de software: el tradicional y el ágil. Las metodologías ágiles representan, por un lado, un enfoque alternativo para la gestión de los proyectos, puesto que se basan en la capacidad de responder rápidamente a los cambios mediante el uso del

desarrollo iterativo e incremental. A diferencia de, por otro lado, el enfoque tradicional que sigue un conjunto predefinido de pasos. Es decir, la ventaja principal de las metodologías ágiles es que permiten la adaptación de las fases en función de las circunstancias cambiantes.

Considerando el entorno industrial, el *Standish Group*¹ ha resaltado recientemente los beneficios que se obtienen al seguir un enfoque ágil en el desarrollo de un proyecto de software, en comparación con aquellos que utilizan, por ejemplo, un ciclo de vida tradicional en cascada, argumentando que los proyectos ágiles tienen tres veces más probabilidades de éxito (véase Figura 1.1). Aunado a esto, es importante mencionar que esta orientación a los proyectos ágiles no es nueva, ya que el Standish Group ha insistido en esta idea durante aproximadamente una década.



Figura 1.1. Tasa de éxito de los proyectos de software en el mundo. Nota: Traducida de Johnson (2020).

Los datos del informe publicado en el 2020 indican que los proyectos pequeños tienen una tasa de éxito dos veces más alta que la de los proyectos grandes, esto debido a que los primeros enfrentan menos riesgos dado que se desarrollan en ciclos de tiempo más cortos y se facilita, en gran medida, la retroalimentación. Por lo que, de ser posible, deben evitarse los proyectos grandes y si no se pudiese, priorizar los esfuerzos por dividirlos en proyectos pequeños con el fin de reducir el riesgo y mejorar las probabilidades de éxito (véase Figura 1.2).

Vale la pena destacar que el informe también identifica un factor clave que hace sobresalir a las metodologías ágiles frente a las tradicionales, su adaptabilidad. Es decir, mientras que las metodologías tradicionales abogan por los procesos predictivos, las metodologías ágiles emplean ciclos cortos e iterativos que generan entregas pequeñas, lo que favorece a la integración continua del producto.

¹ El *Standish Group Inc.* es un organismo internacional independiente que recopila, analiza y describe el rendimiento de las organizaciones de desarrollo de software desde 1985. Los datos reportados en su “informe del caos” miden el éxito y fracaso de los proyectos de software, siendo la fuente de información más utilizada como referencia para analizar el comportamiento de la Ingeniería de Software en la industria moderna. Los datos proporcionados por dicho informe son concentrados cada cuatro años y resumen las respuestas de diversas organizaciones encuestadas en los Estados Unidos de Norteamérica (60%), Europa (25%), y el resto del mundo (15%), pero que bien pueden ser utilizados para entender el fracaso de los proyectos de software a nivel mundial

TASAS DE ÉXITO DEL PROYECTO POR TAMAÑO DEL PROYECTO ÁGIL VS CASCADA

PARA PROYECTOS GRANDES, LOS ENFOQUES ÁGILES TIENEN 2 VECES MÁS PROBABILIDADES DE ÉXITO



Figura 1.2. Tasa de éxito por tamaño de proyecto. Nota: Traducida de Johnson (2020).

De acuerdo con Kuhrmann et al. (2022), en un entorno ágil, si se requiere un cambio, el equipo no debe detener su flujo de trabajo; en su lugar, se enfoca en gestionar los cambios que ocurran en el proyecto. Por otro lado, las metodologías tradicionales no permiten cambios, ya que su proceso define y documenta requisitos estables al inicio del proyecto. Sin embargo, de acuerdo con el portal oficial de Scrum (<https://www.Scrum.org/>), existen variables que son causantes del fracaso de los proyectos ágiles, una de ellas es la falta de liderazgo. Es decir, se argumenta que cada organización debe tener un buen líder que evite obstaculizar el espíritu de comunicación y trabajo en equipo. Pero la realidad es que en la vida real estos líderes a menudo carecen de experiencia y, como consecuencia, tienen problemas para comprender sólidamente los principios ágiles (Attar y Abdul-Kareem, 2020). Considerando lo anterior, el propietario del producto (PO, por sus siglas en inglés), evita asumir las responsabilidades del rol perjudicando así a la toma de decisiones, lo que causa una falta de claridad y dirección en el proyecto.

Ahora bien, Nisyak et al. (2020), Tam et al. (2020), y Traini (2022) coincidieron en que a menudo el fracaso de los proyectos ágiles se genera desde diferentes partes de la organización mediante la denominada “resistencia al cambio” (i.e., cambiar una metodología tradicional por otra ágil), lo que ocasiona que no se tomen las decisiones y acciones correctivas adecuadas durante el desarrollo de un proyecto. Aunado a esto, Ciancarini et al., (2021) y Strode et al., (2022) argumentaron que la falta de comunicación entre los líderes del proyecto con el equipo, así como el constante cambio en los requisitos, afectan negativamente la estabilidad y el enfoque del proyecto. Como consecuencia de ello, se crea un ambiente carente de colaboración entre los integrantes del equipo, lo que pone en riesgo la entrega continua de valor y genera una deficiente gestión del tiempo y los recursos.

Sin embargo, es importante mencionar que independientemente del enfoque que se siga, el desarrollo efectivo del proyecto de software requiere de la organización y gestión de cada recurso necesario para completarlo, considerando las limitaciones, el alcance, la calidad, el tiempo y costo (Srivastava et al., 2020). Para ello, las normas y estándares de la gestión de proyectos facilitan la implementación de metodologías que se adaptan a las necesidades específicas de cada proyecto. En este sentido, la gestión de los proyectos de software se debe llevar a cabo siguiendo un cuerpo de conocimiento o estándar de tal manera que sea posible implementar una metodología de planificación y control que haga uso de diversas técnicas y herramientas. De acuerdo con Fang (2021) y Grebić y

Stojanović (2021), esto ayuda a garantizar que los proyectos se ejecuten de manera más eficiente y se ajusten a las mejores prácticas en la gestión ágil de los proyectos.

Por otro lado, Choraś et al. (2020) y Baham y Hirschheim (2022) mostraron que la gestión de los proyectos de software es un proceso complejo de llevar a cabo puesto que la mayoría de las veces se enfoca en las expectativas de los usuarios y el cumplimiento de los objetivos, algo que puede cambiar sin previo aviso y poner en riesgo el éxito del proyecto. Por lo tanto, la planificación es esencial para procurar el control y monitorización de los proyectos y, para ello, es necesario realizar estimaciones precisas que ayuden a determinar el tamaño y costo del producto, además del esfuerzo que conlleva el desarrollarlos (Sandeep et al., 2022). Así pues, el principal objetivo de la estimación del esfuerzo es delimitar la carga de trabajo, por lo que tales estimaciones deben ser fiables para facilitar el establecimiento de objetivos alcanzables y claros. En este sentido, la estimación del esfuerzo es crucial en la industria de software, puesto que permite a las organizaciones determinar sus necesidades, inversiones en recursos, tiempos de desarrollo y recursos requeridos para realizar una planificación eficiente de los proyectos (Prieto-Bustamante, 2020). Además, la estimación del esfuerzo desempeña un papel fundamental en la toma de decisiones, y su efectividad depende de la confiabilidad del modelo de estimación y del ciclo de vida seleccionado para desarrollar el sistema (Fernández-Diego et al., 2020).

Desafortunadamente, el realizar estimaciones de esfuerzo en un entorno ágil plantea desafíos, debido a la evolución constante de los requisitos. Por lo tanto, se requiere comprender los enfoques individuales para comparar y sintetizar los beneficios, dado que la naturaleza volátil de los procesos de desarrollo de software cambia rápidamente los requisitos del cliente y exige entregables a corto plazo.

1.2. Importancia del problema

En este contexto, la estimación del esfuerzo en el desarrollo ágil de software debe abordarse desde diversos ángulos, incluyendo factores, parámetros, atributos, características y variables, que influyen en la estimación misma. Dado que la experiencia en el desarrollo ágil de software juega un papel crucial para seleccionar las técnicas, métricas y metodologías efectivas (Sudarmaningtyas y Mohamed, 2021), a diferencia de las metodologías tradicionales, donde las estimaciones no están relacionadas con los cambios en los requisitos, ya que éstos están bien definidos y los cambios son controlados rigurosamente. Así pues, errar en la estimación del esfuerzo en un entorno ágil de desarrollo puede ocasionar insatisfacción en los clientes, incrementar los costos de desarrollo y presupuesto del proyecto, e incluso se puede ocasionar la generación de penalizaciones y, como consecuencia más grave, la pérdida de contratos (Govil y Sharma, 2022). Aunado a lo anterior, existen peculiaridades que demuestran que el desarrollo ágil se puede complicar dependiendo las características de las organizaciones en donde se realice.

Una realidad innegable es que las metodologías ágiles de desarrollo son ampliamente adoptadas hoy en día por un sinnúmero de organizaciones de todo tipo. De hecho, diversos estudios informan de los beneficios que genera la adopción de un enfoque ágil de desarrollo, los cuales van desde la disminución del tiempo de comercialización hasta el incremento de la satisfacción del cliente y la reducción de los costos de desarrollo, entre otros (Holgeid y Jørgensen, 2020; Bonner et al., 2021; Almeida et al., 2022). Sin embargo, la gestión de los proyectos ágiles es todo un desafío, especialmente en el caso de las pequeñas y medianas organizaciones desarrolladoras de software. En este sentido, la investigación de Choraś et al. (2020) argumentó que, por ejemplo, la realización de proyectos ágiles en este tipo de organizaciones es una tarea exigente puesto que además de que los proyectos deben comenzar y finalizar rápidamente, también deben de cumplir con los requisitos de

calidad de los clientes; es decir, que el producto funcione correctamente, que se encuentre dentro del presupuesto establecido y que sea entregado a tiempo. Aunado a esto, diversos factores como el cambio cultural, la colaboración del cliente, el desarrollo de competencias y conocimientos, la burocracia y los procesos existentes, el compromiso de la alta dirección, la productividad, y la previsibilidad y reducción de costos del proyecto pueden afectar negativamente a que una “pequeña organización”² pueda implementar exitosamente un enfoque ágil de gestión (Mantovani Fontana y Marczak, 2020; Behrens et al., 2021; Mishra et al., 2021). Considerando que la previsibilidad de un proyecto y, como consecuencia, la reducción del esfuerzo y costos de este son características que deben planificarse y monitorizarse con cuidado, la estimación cobra importancia dentro del enfoque ágil puesto que ésta les proporciona a las organizaciones el soporte necesario para lograr la planificación y ejecución exitosa de los proyectos. La estimación de un proyecto implica, en consecuencia, predecir el esfuerzo, el tamaño, la asignación de personal, el cronograma (i.e., tiempo) y el costo involucrado en la creación de un producto (Lock, 2020). Mendes (2013) argumentó que, en el contexto de un proyecto de software, dicho esfuerzo se relaciona con la cantidad de horas que una persona requiere para desarrollar una aplicación de software. Por lo tanto, la estimación del esfuerzo se define como *“el proceso mediante el cual se determina la cantidad de, valga la redundancia, horas-persona requeridas, considerando los recursos necesarios, para finalizar una actividad del proyecto con el fin de construir un producto o servicio que cumpla con los requisitos funcionales y no funcionales proporcionados por un cliente”* (Trendowicz y Jeffery, 2014).

Como se puede ver, la estimación del esfuerzo se realiza como parte de un proceso para desarrollar software y juega un papel crucial cuando se pretenda cumplir con las expectativas de calidad de un cliente. Ahora bien, considerando un proceso de desarrollo ágil, el software se desarrolla incrementalmente a través de pequeñas iteraciones y la retroalimentación de los clientes es crucial para las iteraciones posteriores. Esto implica, al mismo tiempo, que las estimaciones y los planes deben realizarse de forma progresiva (Usman et al., 2015). De esta manera, si las estimaciones del esfuerzo son precisas, pueden contribuir a lograr el éxito de los proyectos, mientras que las estimaciones incorrectas pueden afectar negativamente el *marketing* y las ventas de las organizaciones, provocando así pérdidas monetarias importantes (Fernández-Diego et al., 2020). En caso de que la estimación se realice correctamente, la información recogida sería utilizada como base para predecir los costos del proyecto y asignar los recursos a, por ejemplo, los desarrolladores.

Ahora bien, la estimación del esfuerzo es independiente de la metodología ágil que cada organización haya implementado, puesto que se basa regularmente en la experiencia del personal y la información disponible. No obstante, las investigaciones de Agrawal et al. (2014), Choraś et al. (2020), y Pastrana-Pardo et al., (2022) afirman que la mayoría de las pequeñas y medianas organizaciones de software utilizan procesos similares a Scrum (véase Figura 1.3). De acuerdo con Rubin (2012) y Schwaber y Sutherland (2020), *“Scrum es un marco ligero que ayuda a las personas, equipos y organizaciones a generar valor a través de soluciones adaptables para problemas complejos”*. Este marco de trabajo está conformado por cinco eventos, reuniones o ceremonias, un

² De acuerdo con la Secretaría de Economía de México, una “pequeña organización” es una compañía dedicada al intercambio comercial mediante las actividades que una plantilla de entre 11 y 30 trabajadores realizan para producir un ingreso anual por encima de los \$4,000,000 y hasta los \$100,000,000. En el contexto del desarrollo de software en México, este tipo de organizaciones representan entre el 95% y 99% de la actividad en el país, aunque, hablando específicamente del software, los ingresos definidos por la Secretaría de Economía no se ajustan necesariamente a la realidad de este sector. Finalmente, la investigación de Pacheco et al., (2023) establece información un poco más precisa sobre las características de estas organizaciones al exponer que el tamaño del personal de desarrollo de software va de los 11 a los 30 empleados, el tamaño de los proyectos asciende de las 50,000 a 100,000 líneas de código, y el tiempo destinado a cada proyecto se encuentra entre los seis y doce meses.

evento contenedor llamado *sprint* y cuatro eventos formales que son la planificación del *sprint*, el Scrum diario, la revisión del *sprint* y la retrospectiva del *sprint*. Es precisamente durante la planificación del *sprint* que se realizan las estimaciones del esfuerzo, considerando el tamaño y los requisitos del proyecto.

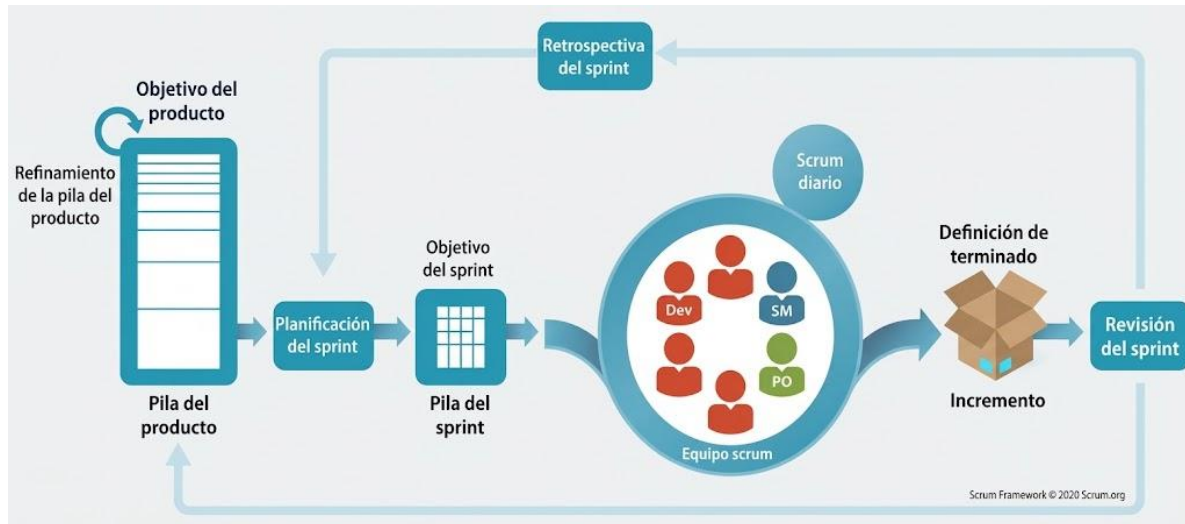


Figura 1.3. Marco de trabajo Scrum. Nota: Traducida de Schwaber y Sutherland (2020).

Considerando la Figura 1.3, la estimación del esfuerzo en Scrum se lleva a cabo durante la planificación del *sprint*, donde se determina la cantidad de trabajo que el equipo puede comprometerse a realizar en dicho *sprint*. Para lograr esta tarea es común que se empleen diferentes técnicas que dependen en gran medida de la experiencia del equipo, como los puntos de historia (SP, por sus siglas en inglés) (Coelho y Basu, 2012), que representan la complejidad de una tarea en lugar del tiempo real, o el póker de planificación o *planning poker* (Mahnič y Hovelja, 2012), para la cual los miembros del equipo asignan valores a las tareas hasta llegar a un consenso sobre su complejidad relativa. Algunos equipos optan por estimar en horas, asignando el tiempo esperado para cada tarea, aunque esta práctica es menos común en Scrum. Como se ha mencionado, durante la estimación se considera la experiencia pasada, la incertidumbre y la complejidad de las tareas (Tanveer et al., 2016, 2017). Una vez que se determina cuánto trabajo puede completarse en el *sprint*, las actividades se agregan al *sprint backlog* como compromisos para ese período de tiempo. Es crucial comprender que estas estimaciones no representan tiempos precisos, sino la complejidad y el esfuerzo requerido para realizar las tareas. De esta manera, el equipo puede planificar y comprometerse de manera realista dentro del *sprint*.

Como se podrá entender, existen técnicas que se utilizan para realizar las estimaciones de esfuerzo en las organizaciones desarrolladoras de software. Sin embargo, como se mencionó anteriormente, las peculiaridades de las organizaciones pequeñas dificultan la adopción de cualquiera de ellas. La investigación de López-Martínez et al. (2016), por ejemplo, evidenció los problemas de adoptar una metodología ágil como Scrum mediante la identificación de cuatro áreas de incidencia: proyecto (i.e., tamaño de los proyectos), proceso (i.e., grado de agilidad, rechazo a los patrones de trabajo), organización (e.g., problemas organizacionales, falta de soporte a la gestión, presión externa para usar prácticas tradicionales), y personal (e.g., falta de experiencia con las metodologías ágiles, falta de entendimiento de los valores ágiles, capacidad del equipo). Años más tarde, Putrianasari et al., (2024) confirmó la permanencia de estos problemas y la identificación de otros nuevos mediante su clasificación en cuatro áreas: personal (e.g., falta de colaboración y comunicación, resistencia al

cambio, carencia de habilidades y conocimientos, falta de una toma de decisiones participativa, alta rotación), procesos (e.g., adaptación de las prácticas ágiles, prácticas inconsistentes, priorización de la planificación, definición del alcance y tamaño del proyecto), organización (e.g., entrenamiento insuficiente o inadecuado, control de la calidad, carencia de una orientación formal y efectiva), y habilidades técnicas (e.g., tecnologías y herramientas inadecuadas, falta de mecanismos para el seguimiento de los procesos, herramientas para validar el proceso).

En resumen, se puede afirmar que, en la actualidad, las actividades relacionadas con la estimación del esfuerzo en las pequeñas organizaciones desarrolladoras de software, que siguen o intentan seguir los principios de Scrum, se basan principalmente en el conocimiento y las percepciones del equipo y se cuenta con poco apoyo para tomar decisiones basadas en datos, por lo que todavía existe una brecha y la necesidad de buscar más soluciones que reflejen la efectividad de las estimaciones que se realizan en tales organizaciones, de tal manera que se logre mejorar la calidad de los procesos y productos de software.

1.3. Necesidad de solución

Teniendo en mente que la estimación del esfuerzo es el proceso de predecir, de forma realista, la cantidad de esfuerzo necesario (expresada en términos de horas-persona o dinero) para desarrollar o mantener el software considerando datos incompletos, inciertos y ruidosos, es importante que las organizaciones las realicen correctamente puesto que éstas pueden utilizarse como insumos para los planes de proyectos, planes de iteración, presupuestos, análisis de inversiones, procesos de fijación de precios y rondas de licitación. En este sentido, con el objetivo de obtener estimaciones efectivas en un entorno industrial, diversas investigaciones se han realizado para alcanzar beneficios importantes.

La investigación realizada por Ferreira Lorenzo et al., (2014) analizó un conjunto de técnicas basadas en principios de la Inteligencia Artificial (IA) que se aplicaron a la estimación del esfuerzo en los proyectos de software, incluyendo al razonamiento basado en casos, los algoritmos genéticos, la programación genética, la regresión de vectores de soporte, las redes neuronales artificiales, los sistemas de inferencia borrosa, la lógica difusa y los árboles de decisión. En estos años, los autores ya destacaban un aumento en el uso de los algoritmos genéticos, la programación genética y la regresión de vectores de soporte, así como la prevalencia de enfoques híbridos en la estimación, liderados por el uso de los algoritmos genéticos y la lógica difusa. La principal contribución de esta investigación fue que se demostró que estas técnicas debían y podían aplicarse sobre los repositorios de proyectos reales, dependiendo el tipo de proyecto que se pretendiera desarrollar. Aunado a lo anterior, también se hizo notar que existía una diversidad importante en las características de los repositorios, aspecto que podría dificultar la selección del mejor modelo de estimación. Además, se identificó que las métricas utilizadas para evaluar a los modelos varían entre investigadores, siendo la preferida el error medio relativo, aunque se sugirió la posibilidad de ampliar la evaluación con el uso de residuales absolutos y técnicas estadísticas. Por otro lado, la principal desventaja del estudio es que no se hace distinción sobre si alguna de las técnicas analizadas podría adaptarse a las características específicas de las pequeñas organizaciones.

Alostad et al. (2017), por otro lado, propusieron un modelo difuso para la estimación del esfuerzo en proyectos desarrollados bajo Scrum. El estudio se centró en factores clave que afectan la estimación del esfuerzo, tales como la experiencia del equipo de desarrollo, la complejidad de la tarea, el tamaño de la tarea y la precisión de la estimación, los cuales son expresados mediante cuantificadores lingüísticos. El modelo constó de tres componentes principales el “fuzzificador”, un motor de inferencia y el “defuzzificador”, los cuales fueron implementados en MATLAB. El motor de inferencia difusa se basó en reglas SI-ENTONCES, para combinar entradas difusas y generar un

punto de historia estimado para cada tarea. Los puntos de historia estimados se acumularon para obtener la estimación total de esfuerzo del *sprint*. El defuzzificador (i.e., el método centroide) convertía las cantidades difusas en valores nítidos para facilitar su procesamiento. La evaluación del modelo se realizó mediante el uso de métricas comunes como la Magnitud del Error Relativo (MRE, por sus siglas en inglés) y el Nivel de Predicción en Nivel 1 (PRED 1), las cuales permitieron determinar la precisión en términos de la estimación del esfuerzo en comparación con el esfuerzo real a lo largo de varios *sprints*.

De manera similar, la investigación de Septian y Gata (2017) propuso la combinación del Marco de Trabajo Ágil para Pequeños Proyectos (AFSP, por sus siglas en inglés) con redes neuronales para realizar la estimación. AFSP se basa en técnicas de Scrum y consta de dos componentes principales: el proceso AFSP, que guía la construcción del software correcto, y el grupo de práctica de AFSP, que determina las técnicas y herramientas para construir el software correctamente. La investigación además consideró una red neuronal de Perceptrones Multicapa (MLP, por sus siglas en inglés) por su capacidad para abordar el aprendizaje complejo y adaptarse a conjuntos de datos en tiempo real, además de su compatibilidad con modelos no lineales. En este sentido, se realizó el proceso de identificación de riesgos en seis proyectos de una organización y se consideró un conjunto de datos con atributos de velocidad inicial, desaceleración, tamaño del *sprint*, esfuerzo, días laborables, tiempo estimado, tiempo real y tiempo MRE. Las estimaciones se realizaron utilizando la red neuronal MLP con retro propagación. La configuración incluyó 500 ciclos de entrenamiento, una tasa de aprendizaje de 0.3, un impulso de 0.2 y una optimización de errores <0.00001 .

Posteriormente, la investigación de Mendoza et al., (2019) propuso una metodología de estimación que fue aplicada en 10 pequeñas organizaciones desarrolladoras de software, de las cuales el 62% utilizaba Scrum, el 25% seguía la Programación Extrema (XP, por sus siglas en inglés) y el 13% hacía uso de Kanban. Dicha investigación identificó varios métodos de estimación utilizados para los *sprints*, como la planificación de póker para el *backlog* y la estimación por horas, basada en la experiencia del programador, la secuencia de Fibonacci para el *backlog* y la metodología de las tallas de las camisetas, o *T-shirt sizing*. Un hallazgo adicional de la investigación fue que la mayoría de las organizaciones basan sus estimaciones en la comparativa de proyectos anteriores, utilizando la experiencia acumulada como referencia clave junto con los datos históricos. De esta manera, la metodología propuesta se basó en la realimentación de los equipos de las organizaciones que habían implementado y mejorado un proceso de Scrum, ajustándose a los lineamientos del marco y aprendiendo de la experiencia. En este sentido, se establecieron dos niveles de estimación: el primero, utilizando técnicas como la planificación de póker, las tallas de las camisetas u ordenamiento con puntos de historia; y el segundo, utilizando, al inicio del *sprint*, las horas para estimar la duración de las tareas, con la colaboración del responsable o un experto para mejorar la precisión. La colaboración entre el Scrum Master (SM, por sus siglas en inglés) y el equipo de desarrollo fue identificada como esencial en la estimación inicial. Se destacó también la importancia de que el SM tenga experiencia y comprenda al equipo de desarrollo, puesto que desempeña un papel clave. Se recomendó además que el equipo de trabajo cuente con un catálogo de características acordadas en términos de complejidad para agilizar la estimación. Los resultados de dicha investigación mostraron que solo dos organizaciones contaban con un mecanismo establecido para realizar la estimación del presupuesto y esfuerzo, demostrando así que la mayoría de las pequeñas organizaciones basa sus estimaciones en la experiencia, sin mantener ni utilizar registros históricos. Por lo tanto, los investigadores argumentaron que, aunque el estandarizar la medición de la complejidad para predecir un presupuesto con más del 90% de certeza es desafiante debido a las variaciones entre los proyectos, la mayoría de las organizaciones debería contar con una herramienta que proporcione certeza en las estimaciones y fomente el autoconocimiento del equipo. Como recomendación adicional, los investigadores

sugirieron ampliar el estudio a más organizaciones, regionalizar un modelo y automatizar su flujo, incluyendo un catálogo estandarizado de historias de usuario y registros históricos. Por último, también se planteó la idea de realizar un análisis comparativo entre diversas técnicas para esbozar conclusiones a partir de los resultados.

La investigación de Prieto-Bustamante (2020) presentó los resultados de realizar un estudio empírico que buscó comprender el estado actual de la práctica en la estimación del esfuerzo dentro de pequeñas organizaciones que implementan el desarrollo ágil de software. Dicho estudio recogió información mediante la aplicación de un cuestionario a expertos y trabajadores de estas organizaciones en Bogotá, Colombia, ambos con experiencia sobre la estimación de esfuerzo. Los resultados revelaron que la mayoría de los equipos de desarrollo ágil de software utilizan técnicas como los puntos de casos de uso y los puntos de historia, puesto que fomentan el consenso entre los miembros del equipo y reducen el tiempo dedicado a la estimación y, en consecuencia, el estrés en el equipo. En este sentido, resultó que las líneas de código y los puntos de historia eran los predictores de esfuerzo más adoptados por los desarrolladores de estas organizaciones, mientras que otras métricas como los puntos de casos de uso, el reporte mensual de actividades, la funcionalidad de la aplicación, y los puntos función se utilizan con menos frecuencia. De acuerdo con la información recogida, el 47% de los encuestados afirmó que la estimación del esfuerzo en el desarrollo de software tiende a estar sub y sobreestimada en un rango entre el 25% y 50%, siendo la subestimación la tendencia dominante con un 33%.

En años más recientes, Hamid et al., (2021) propusieron un sistema basado en lógica difusa para asistir al SM en la estimación del esfuerzo durante la reunión de planificación del *sprint*. Este sistema recibe cuatro valores de entrada: la experiencia de los integrantes del equipo (i.e., principiante, intermedio, y experimentado), la complejidad de la tarea como categorías lingüísticas (i.e., fácil, moderado, y complejo), el tamaño de la tarea (i.e., pequeñas, medianas, y grandes), y el entorno laboral (i.e., peor, normal, y mejor). y la variable de salida proporciona puntos de historia estimados. El sistema facilita respuestas en condiciones SI-ENTONCES, representando una lógica de toma de decisiones que dependen de los cuatro factores con relación a sus valores seleccionados. El motor de inferencia analiza la entrada difusa, identificando cuántos valores de grados coincidentes existen de acuerdo con la base de reglas establecida. Los resultados experimentales revelaron que, cuando la experiencia del equipo es intermedia, la complejidad de la tarea es fácil, el tamaño de la tarea es medio y el ambiente de trabajo es normal, el valor estimado de los puntos de la historia es intermedio.

Finalmente, la investigación de Arora et al., (2022) muestra la aplicación de un Sistema Adaptativo de Inferencia Neurodifusa (ANFIS, por sus siglas en inglés) junto con la Técnica de BAT Energéticamente Eficiente (EEBAT, por sus siglas en inglés), que optimiza la curva de aprendizaje del ANFIS base, para la predicción del esfuerzo en Scrum. ANFIS destacó por su capacidad para mejorar la eficiencia de ANFIS, permitiendo así una adaptación más efectiva a los cambios en los requisitos del software y la optimización de la estimación de esfuerzo en proyectos ágiles. Este enfoque fue validado estadísticamente mediante pruebas no paramétricas para concluir que funciona de manera superior en comparación con varios algoritmos metaheurísticos y de aprendizaje computacional (*machine learning*) de última generación, tales como fuegos artificiales, Optimizador de Hormiga León (ALO, por sus siglas en inglés), murciélago, Optimización de Enjambre de Partículas (PSO, por sus siglas en inglés) y los algoritmos genéticos.

En este sentido, como se muestra en las investigaciones presentadas anteriormente, existen principalmente dos enfoques que abordan la estimación del esfuerzo en las metodologías ágiles que son implementadas tanto en las organizaciones grandes como en las pequeñas. La orientación hacia las grandes organizaciones, por un lado, está indagando y recogiendo lecciones exitosas sobre la

incorporación de diferentes técnicas, métodos y/o algoritmos de la IA, mientras que, por otro lado, la orientación hacia las pequeñas organizaciones se enfoca, en su mayoría, a identificar qué métodos y modelos tradicionales son más eficientes para adaptarse a sus características específicas considerando, mayormente, la experiencia y madurez del personal. Por lo que, esta tesis se orientó a conducir una investigación que mejore la calidad de las estimaciones de esfuerzo que realizan los equipos de Scrum mediante la aplicación del aprendizaje computacional, lo que ayudaría a que en el entorno de las organizaciones pequeñas se obtengan estimaciones fiables, independientemente del entorno cambiante que a menudo se experimenta con la implementación del desarrollo ágil de los proyectos de software.

1.4. Delimitaciones y limitaciones de la tesis

La presente investigación estuvo delimitada por los siguientes aspectos:

- Dada la cantidad de estudios enfocados al contexto de las pequeñas organizaciones desarrolladoras de software, la solución propuesta en esta tesis es el resultado del análisis y comparación de las investigaciones realizadas para incorporar métodos, técnicas y algoritmos del aprendizaje computacional con el fin de mejorar las estimaciones de esfuerzo en proyectos ágiles de desarrollo.
- El método de estimación resultante ha sido integrado en una herramienta con el fin de que se pueda proporcionar un soporte tecnológico para la estimación del esfuerzo en las pequeñas organizaciones que utilizan Scrum.
- La herramienta se probó con datos provenientes de bases de datos públicas sobre estimaciones realizadas para diversos proyectos de software.
- La efectividad de la herramienta se evaluó empíricamente en una pequeña organización desarrolladora de software.

Además, las principales limitaciones que se contemplaron durante la realización de la investigación fueron las siguientes:

- La efectividad de la herramienta, así como del método de estimación creado, se pudo ver afectada por el nivel de conocimiento que poseían tanto el equipo de desarrollo como los líderes de proyectos de la organización participante en la evaluación.
- De manera similar, los resultados de la evaluación empírica se pudieron ver afectados negativamente por la capacidad del personal de la organización para aplicar los principios ágiles definidos en Scrum.

1.5. Hipótesis

De acuerdo con la problemática planteada anteriormente, la hipótesis que condujo el desarrollo de la investigación se enuncia como:

Hi: *“A través de un método de estimación que haga uso del aprendizaje computacional, se podrá determinar el esfuerzo requerido para desarrollar proyectos con Scrum en una pequeña organización desarrolladora de software”.*

1.6. Objetivos del trabajo

Considerando lo anterior, esta tesis estableció los siguientes objetivos:

1.6.1. Objetivo general

Desarrollar un método de estimación del esfuerzo requerido en proyectos de pequeñas organizaciones desarrolladoras de software realizados con Scrum a través del uso del aprendizaje computacional.

1.6.2. Objetivos específicos

Cabe mencionar que el objetivo general de la tesis fue alcanzado a través de la consecución de los siguientes objetivos específicos:

1. Realizar una investigación bibliográfica en la literatura especializada actual con el fin de encontrar y analizar propuestas creadas para la estimación del esfuerzo en equipos Scrum, que sirvan como punto de partida para su adaptación al contexto de las pequeñas organizaciones desarrolladoras de software.
2. Generar modelos para la estimación del esfuerzo requerido en proyectos de software de pequeñas organizaciones aplicando algoritmos de aprendizaje computacional para determinar su efectividad en la estimación.
3. Desarrollar modelos de aprendizaje computacional diferenciados para estimar el esfuerzo a nivel de proyecto (duración total en semanas) y a nivel de *sprint* (puntos de historia acumulados), evaluando su rendimiento mediante métricas estándar de regresión (MAE, RMSE, MMRE y PRED).
4. Evaluar la efectividad y utilidad del método propuesto mediante un caso de estudio de comparación directa en una pequeña organización real, contrastando las predicciones del modelo con las estimaciones del póker de planificación.

1.7. Metodología de trabajo

La estimación del esfuerzo es crucial para el éxito de los proyectos de software, y a lo largo del tiempo se han desarrollado diversos métodos para mejorar dicho proceso (Jadhav et al., 2022). A pesar de estos esfuerzos, la precisión de las estimaciones sigue siendo un desafío significativo en la industria. De hecho, Hameed et al., (2023) consideran que la precisión en las estimaciones es esencial para evitar los retrasos en la entrega, los sobrecostos y otros problemas que afectan la calidad de los resultados. Por lo tanto, independientemente de la metodología de desarrollo, estimar incorrectamente el esfuerzo aumenta la probabilidad de fracaso en los proyectos de software. Sin embargo, como se ha mencionado a lo largo del documento, el escenario se complica al involucrar a las pequeñas organizaciones de software, puesto que la carencia de información histórica, la poca experiencia del personal, y los conocimientos escasos sobre aspectos modernos de la Ingeniería de Software complica la obtención de las estimaciones. Por lo tanto, la investigación desarrollada en esta tesis se compuso de las fases mostradas en la Figura 1.4, las cuales son descritas en las siguientes secciones.

De acuerdo con Arora et al. (2020), la transición de modelos o ciclos de vida de la Ingeniería de Software, como el modelo en cascada, a enfoques ágiles ha generado cambios en los métodos de estimación del esfuerzo. Los métodos como el juicio de expertos (Jørgensen, 2005), la estimación de

arriba hacia abajo y de abajo hacia arriba (Jørgensen, 2004) y la estimación de costos con Delphi (Gandomani et al., 2014) son apropiados para los enfoques ágiles, pero debido a la naturaleza volátil de los requisitos en los proyectos ágiles, no es posible cerrar la brecha entre el esfuerzo estimado y el real en este tipo de proyectos. En este sentido, la literatura especializada sugiere que, en lugar de determinar qué modelo es mejor considerando sus características individuales, es más productivo elegir el modelo que mejor se adapte a un entorno específico. Esto se debe a que los modelos de estimación muestran comportamientos únicos en diferentes conjuntos de datos, volviéndolos menos confiables (Saeed et al., 2018). Considerando esto, las investigaciones de Jodpimai et al. (2018a), Jodpimai et al. (2018b), Karna et al., (2019), y Ogumyele (2020), por ejemplo, muestran que, mediante la aplicación de técnicas de minería de datos, las estrategias grupales ofrecen resultados más precisos en comparación con las estrategias individuales.

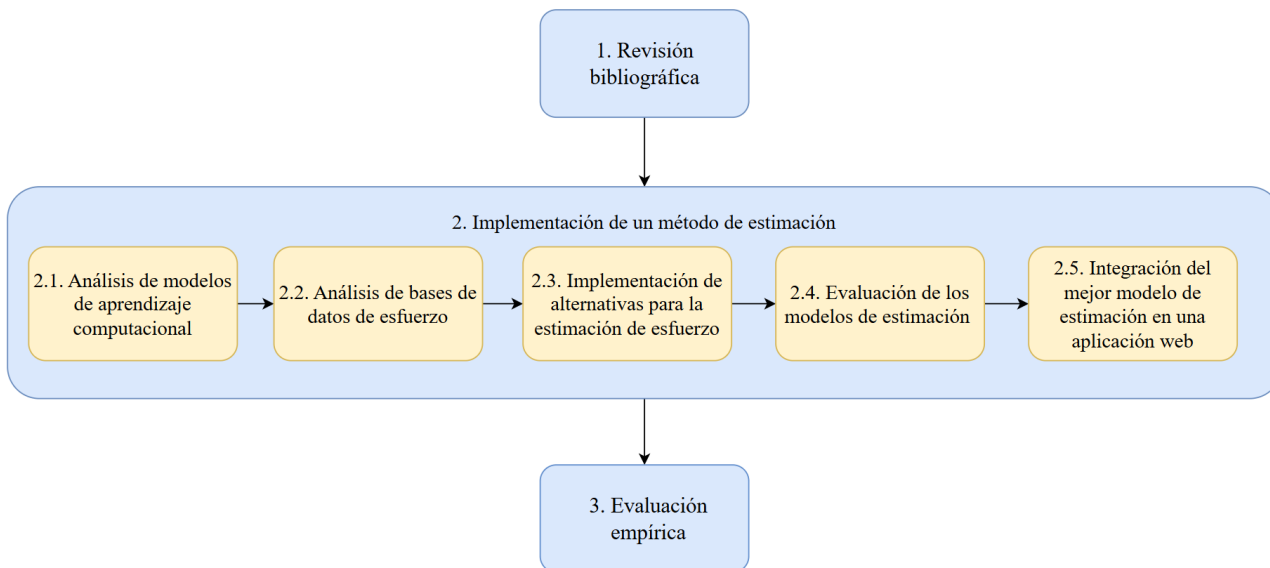


Figura 1.4. Fases y actividades realizadas para el desarrollo de la tesis.

La literatura también proporciona evidencia de que las investigaciones actuales que buscan aplicar el aprendizaje computacional a la estimación del esfuerzo en los proyectos de software apuntan a dos estrategias puntuales: los árboles de decisión y los SVR. De acuerdo con Charbuty y Abdulazeez (2021), los árboles de decisión son estructuras que representan un árbol invertido, cada nodo está dividido por una o más secuencias condicionadas de sí y no. El árbol está compuesto principalmente por nodos, ramas y hojas. Los nodos representan la evaluación de una característica, las ramas representan el resultado de las evaluaciones y las hojas representan la etiqueta de una clase. Por otro lado, el modelo de SVR se considera un algoritmo que se utiliza para realizar tareas de regresión (Zhang y O'Donnell, 2020). A diferencia de otros métodos de regresión, SVR se distingue por utilizar un enfoque basado en vectores de soporte para identificar una función que se adapte a los datos de entrenamiento, permitiendo realizar predicciones precisas en nuevos conjuntos de datos. Sin embargo, aún existen lagunas en la literatura sobre qué método, modelo y/o algoritmo del aprendizaje computacional puede adaptarse a las características de las pequeñas organizaciones de software para mejorar la calidad de las estimaciones del esfuerzo bajo la metodología Scrum.

Por lo tanto, en la primera fase de la tesis se realizó una investigación documental que sirvió para establecer un punto de partida para el diseño y desarrollo de un método alternativo de estimación. En este sentido, se siguieron las recomendaciones y guía puntual de Esquirol-Causa et al., (2017) y Ocaña-Fernández y Fuster-Guillén (2021) para utilizar la revisión bibliográfica como metodología

principal de investigación para analizar los conceptos, enfoques teóricos, estudios y antecedentes de la estimación del esfuerzo utilizando el aprendizaje computacional.

Una vez que se determinó qué técnicas, métodos y/o algoritmos del aprendizaje computacional reciben más atención en los estudios actuales, la segunda fase de la tesis consistió en adaptarlos a las características tanto de las pequeñas organizaciones de software como de Scrum, con el fin de implementarlos y evaluar su rendimiento con datos. Es importante enfatizar que la mayoría de estas organizaciones no dispone de datos históricos y carece de personal bien entrenado. No obstante, se considera que el método de estimación resultante sea integrado en una herramienta que facilite considerablemente la obtención de estimaciones fiables. Para asegurar esto, es común que se utilicen conjuntos de datos de esfuerzo con el objetivo de entrenar un modelo de predicción que pueda, predecir cuánto trabajo requerirá un proyecto. Es decir, para entrenar a un buen “estimador inteligente”, se necesitarán suficientes datos.

Una vez que se mostró la efectividad del método de estimación propuesto, éste fue integrado a una aplicación web con el fin de que fuera utilizado en la gestión de equipos que funcionen bajo la filosofía de Scrum. Es decir, se buscó incorporarla al enfoque de desarrollo iterativo e incremental propuesto por Scrum con el objetivo de mejorar tanto la comunicación del equipo como la calidad del producto final. De acuerdo con Fuentes-Del-Burgo et al. (2022), Scrum utiliza una serie de artefactos y eventos para realizar un seguimiento efectivo del proyecto, maximizando así la transparencia de la información y brindando oportunidades para la inspección y adaptación del equipo. Por lo tanto, la aplicación se alineó con este fin, para la realización de dicho seguimiento.

En el contexto de la Ingeniería de Software, los investigadores pueden evaluar sus propuestas de forma objetiva y científica mediante la realización de una evaluación empírica. De acuerdo con Wohlin et al., (2003), existen tres tipos de investigación empírica que se pueden realizar para la Ingeniería de Software: encuestas, casos de estudio, y experimentos formales. En este escenario, existen tres estrategias para desarrollar un caso de estudio:

1. Comparar los resultados obtenidos de una nueva propuesta contra una la línea base.
2. Desarrollar dos proyectos en paralelo (“proyectos gemelos”), eligiendo uno de ellos como línea base.
3. Aplicar la nueva propuesta a los componentes seleccionados y comparar con los resultados de los componentes a los que no se les había aplicado la nueva propuesta.

Con el objetivo de recoger evidencia que permitiera obtener conclusiones sobre el método de estimación propuesto, la tercera fase de la tesis planteó el diseño y realización de un caso de estudio relacionado con la estrategia de comparación directa dentro de una pequeña organización desarrolladora de software. Esto implicó que se realizaría la estimación del esfuerzo en un proyecto considerando el modelo de estimación basado en aprendizaje computacional para compararla con la estimación realizada por la organización. De esta manera se pretendió recoger información sobre el desempeño del método de estimación para generar estimaciones fiables y las opiniones del equipo de desarrollo sobre la forma de estimar. Es evidente que, por ende, la presente investigación tuvo un enfoque cuantitativo/cualitativo, el cual ayudó a recolectar y analizar los datos obtenidos mediante la aplicación del método de estimación propuesto en una pequeña organización desarrolladora de software que usaba Scrum. En este sentido, Zúñiga et al. (2023) mencionaron que el enfoque cuantitativo se caracteriza por su énfasis en el análisis numérico y en la medición de datos y que, como consecuencia, su principal fortaleza radica en su capacidad para generar resultados comparables entre diferentes variables o grupos. Esta característica lo convierte en una herramienta valiosa para estudios

que requieren mediciones objetivas y la evaluación de efectos cuantificables. Sin embargo, cabe aclarar, que este enfoque podría no capturar la complejidad y riqueza de la experiencia humana, como un enfoque cualitativo. Por lo tanto, se realizó también un análisis cualitativo de la información mediante el diseño y aplicación de cuestionarios al personal de software para determinar sus percepciones sobre el modelo de estimación creado.

1.8. Estructura de la tesis

En el Capítulo 2 se presenta el marco teórico y estado del arte que involucra a las investigaciones actuales que se relacionan con el tópico abordado en esta tesis. Además, se describe paso a paso la metodología seguida para realizar la revisión documental, y se presentan al final los principales hallazgos alcanzados con la misma.

En el Capítulo 3 se presenta el desarrollo e implementación de las alternativas de estimación de esfuerzo. Además, se especifican los recursos necesarios para el desarrollo e integración del modelo de estimación en una aplicación web.

En el Capítulo 4 se presentan los resultados obtenidos en la construcción del conjunto de datos, los diversos métodos de estimación obtenidos, con base a las métricas de evaluación definidas en el Capítulo 2. Aunado a esto, se muestra el *Front-end* de la aplicación web desarrollada y se describe el caso de estudio implementado en una pequeñas organización desarrolladora de software.

En el Capítulo 5 se exponen las conclusiones, de acuerdo con las lecciones aprendidas, y resultados obtenidos en los Capítulos 4 y 5.

La sección de Bibliografía presenta las referencias bibliográficas utilizadas para el desarrollo de la tesis.

Por último, el Anexo A lista todos los acrónimos usados en el documento.

1.9. Publicaciones generadas

A continuación se proporciona la información del producto científico generado como resultado de la presente tesis de grado:

Autoras y autores:	Beatriz Bedolla Martínez, Raúl Cruz-Barbosa, Iván Antonio García Pacheco
Título del artículo:	Mejora de la estimación del esfuerzo en proyectos de software mediante métodos de sobremuestreo y aprendizaje computacional
Nombre de la revista:	Revista de Investigación en Tecnologías de la Información
DOI:	https://doi.org/10.36825/RITL.13.31.008
Volumen:	13
Número:	31
Páginas:	80-93
Mes y año:	Octubre 2025

2. Marco teórico y estado del arte

2.1. Gestión de los proyectos de software

En un entorno empresarial complejo y en constante cambio, la gestión de los proyectos ha evolucionado para transformarse de un enfoque tradicional y exhaustivo en otro más adaptable y ágil. De acuerdo con Leong et al. (2023), este cambio se produjo, en gran medida, por las limitaciones que presenta un enfoque tradicional para hacer frente a la dinámica de trabajo de la industria actual. En este sentido, un número importante de organizaciones están adoptando la gestión ágil de proyectos (APM, por sus siglas en inglés) más allá del ámbito del desarrollo de software, con el fin de ser competitivas en el mercado. La APM se caracteriza por su enfoque gradual y repetitivo, centrado en la flexibilidad, la aceptación del cambio, el progreso continuo y la interacción efectiva. Es decir, además de que la agilidad es un tema predominante en la literatura actual sobre el desarrollo de software, su aplicación se ha extendido a otros sectores empresariales como respuesta a los desafíos y cambios rápidos del mercado (Ciric et al., 2019). No obstante, resulta interesante diferenciar a la gestión tradicional de la gestión ágil.

De acuerdo con Schwalbe (2021), un *enfoque tradicional para gestionar un proyecto* se da por la interacción entre cinco grupos de procesos: inicio, planificación, ejecución, monitorización y control, y cierre (véase Figura 2.1). Para la Guía de los Fundamentos para la Gestión de los Proyectos (PMBOK³, por sus siglas en inglés), estos procesos se categorizan en diez áreas de conocimiento: alcance, tiempo, costo, calidad, riesgo, comunicación, adquisiciones, recursos humanos, partes interesadas y gestión de la integración (PMI, 2021). Dichos procesos se conocen como grupo de procesos de la dirección de proyectos y se describen de la siguiente manera:

- El grupo de procesos de inicio comprende a aquellas actividades relacionadas con la autorización del proyecto, la definición de su alcance inicial, la asignación de recursos financieros y la identificación de las partes interesadas que influyen en su éxito.
- El grupo de procesos de planificación se enfoca en establecer, aclarar y definir el alcance completo del proyecto junto con el esfuerzo requerido para su realización. Este grupo de procesos es fundamental para definir la totalidad de la documentación del proyecto, puesto

³ PMBOK es un documento creado por el Instituto para la Dirección de los Proyectos (PMI, por sus siglas en inglés) con el fin de definir los procesos, prácticas, terminología y directrices para una gestión exitosa de los proyectos (Figueiredo, 2023). El PMI nació hace poco más de 50 años como una organización sin fines de lucro que involucra y asocia a empresas e instituciones relacionadas con la gestión de proyectos con el objetivo de compartir el conocimiento sobre sus actividades, siempre buscando encontrar mejores estrategias para que un proyecto se lleve a cabo de forma eficiente.

que ésta será utilizada para ejecutar, monitorear y controlar el mismo. Dicha documentación incluye la elaboración del cronograma del proyecto, el plan de gestión de riesgos, el plan de gestión de calidad, el plan de gestión del alcance, el plan de gestión de cambios y el presupuesto del proyecto.

- El grupo de procesos de ejecución lleva a cabo las actividades necesarias para completar el trabajo definido en el plan de dirección del proyecto, asegurando que se cumplan con las especificaciones establecidas. Este grupo de procesos implica la coordinación de personas y recursos, la gestión de las expectativas de las partes interesadas, así como la integración y ejecución de las actividades definidas en el plan de dirección del proyecto.
- El grupo de procesos de monitorización y control se encarga de rastrear, revisar y monitorear el progreso y desempeño del proyecto, identificando cualquier área en la que se requieran cambios e iniciando las acciones correspondientes para implementar dichos cambios.
- Finalmente, el grupo de procesos de cierre se enfoca en finalizar todas las actividades necesarias para completar formalmente el proyecto, asegurando que se cumplan con todos los entregables y que se realicen los cierres administrativos y contractuales necesarios.

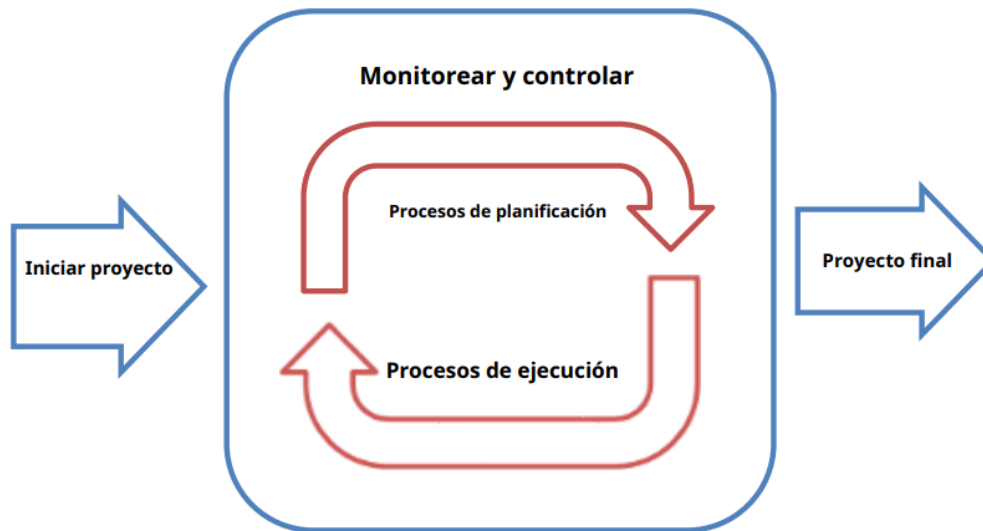


Figura 2.1. Procesos de la gestión tradicional de proyectos. Nota: Traducida de Salameh (2014).

Es importante mencionar que, bajo un enfoque tradicional, los procesos de estos cinco grupos siguen un enfoque predictivo, también conocido como “enfoque en cascada”, el cual es útil cuando los requisitos del producto pueden definirse, recopilarse y analizarse al principio del proyecto. Por lo tanto, esta perspectiva es adecuada para proyectos cuya inversión sea significativa y represente un alto riesgo, puesto que su éxito dependerá de revisiones frecuentes, el uso de mecanismos de control de cambios y, probablemente, el replanteo de estrategias a medida que se avance. De esta manera, cada fase del proyecto debe terminar antes de que la siguiente comience, lo que implica una progresión lineal y secuencial. Sin embargo, la documentación del PMBOK (PMI, 2021) afirma que, a pesar de esta restricción, existen situaciones, tales como que se presenten cambios en los requisitos y/o en el mercado, o que se requiera del aumento del alcance del proyecto, lo que podría generar cambios rigurosos dentro del mismo y generar, como consecuencia, demasiada incertidumbre. Es decir, aunque se pretende que estos procesos se realicen secuencialmente, es común que haya superposiciones informales entre ellos que obliguen a replantear algunas cosas. Por ejemplo, si bien se espera que los planes del proyecto se definan completamente en la fase de planificación, antes de la ejecución, es

posible que se presenten intercambios informales o iteraciones entre las fases. Dada la naturaleza de la gestión de proyectos y el riesgo que existe para terminarlos oportunamente, el grupo de los procesos de monitoreo y control suelen interactuar con los de otros grupos, como se muestra precisamente en la Figura 2.1.

En este sentido, la planificación es una disciplina que establece cómo completar un proyecto en un plazo determinado, generalmente a través de la definición de etapas y la asignación de recursos. De acuerdo con Pérez Pupo et al. (2020), la visión más general sobre la planificación de proyectos es la que divide a este proceso en los siguientes pasos: establecer objetivos mensurables, identificar entregables, calendarizar en el tiempo, y definir tareas de planificación. Considerando lo anterior, el grupo de procesos de planificación establecido por el PMBOK permite determinar el alcance total del proyecto y el esfuerzo requerido para el mismo, definir y refinar sus objetivos, y desarrollar la línea de acción necesaria para satisfacer dichos objetivos. Así, los procesos que se encuentran en este grupo facilitan la elaboración de los componentes del plan necesarios para la gestión del proyecto y de los documentos utilizados durante su ejecución. Por lo tanto, el beneficio principal de estos procesos es establecer una línea de acción clara que permita alcanzar el éxito del proyecto o fase. En otras palabras, una vez que se ha completado la planificación inicial, la versión aprobada del plan se convertirá en una línea base que será utilizada para comparar el desempeño del proyecto a lo largo del tiempo mediante el grupo de los procesos de monitorización y control. La Figura 2.2 muestra que el grupo de procesos de planificación está compuesto por diez áreas de conocimiento, las cuales se describen de la siguiente manera:

- Gestión del alcance del proyecto: Esta área de conocimiento incluye a los procesos necesarios para garantizar que el proyecto involucre a todo el trabajo requerido, y únicamente el trabajo requerido, para completar exitosamente el proyecto. Dichos procesos se definen como:
 - *Planificar la gestión del alcance.* Este proceso permite la creación de un plan que documenta cómo se definirán, validarán y controlarán el alcance del proyecto y del producto. El beneficio principal que se obtiene reside en que el proceso proporciona la guía y dirección para gestionar el alcance a lo largo del proyecto. Por lo tanto, solamente se lleva a cabo una sola vez o en puntos específicos, previamente definidos, del proyecto. Como sus entradas principales se utilizan el acta de constitución del proyecto, el plan para la dirección del proyecto, los factores ambientales de la empresa y activos de los procesos de la organización; mientras que se obtiene como salida el plan para la gestión del alcance y el plan de gestión de los requisitos.
 - *Recopilar los requisitos.* Este proceso facilita la determinación, documentación y gestión de las necesidades y requisitos de los interesados con el fin de cumplir con los objetivos del proyecto. Así, el beneficio principal de realizarlo radica en su capacidad de proporcionar la base para definir el alcance del producto y del proyecto. De igual manera, este proceso se realiza una sola vez o en puntos específicos, previamente definidos, del proyecto.
 - *Definir el alcance.* Este proceso conduce al desarrollo de una descripción detallada del proyecto y del producto. El beneficio principal de su realización es que permite describir los límites del producto, servicio o resultado, junto con los criterios de aceptación. Este proceso también se realiza una sola vez o en puntos específicos predefinidos del proyecto.
 - *Crear la Estructura de Descomposición del Trabajo (WBS, por sus siglas en inglés).* Este proceso establece la necesidad de subdividir los entregables y el trabajo del proyecto en componentes más pequeños y manejables. El beneficio de hacer esto es

que se vislumbra un marco claro de lo que se debe entregar. Igual que los procesos anteriores, se realiza una sola vez o en puntos específicos predefinidos del proyecto.

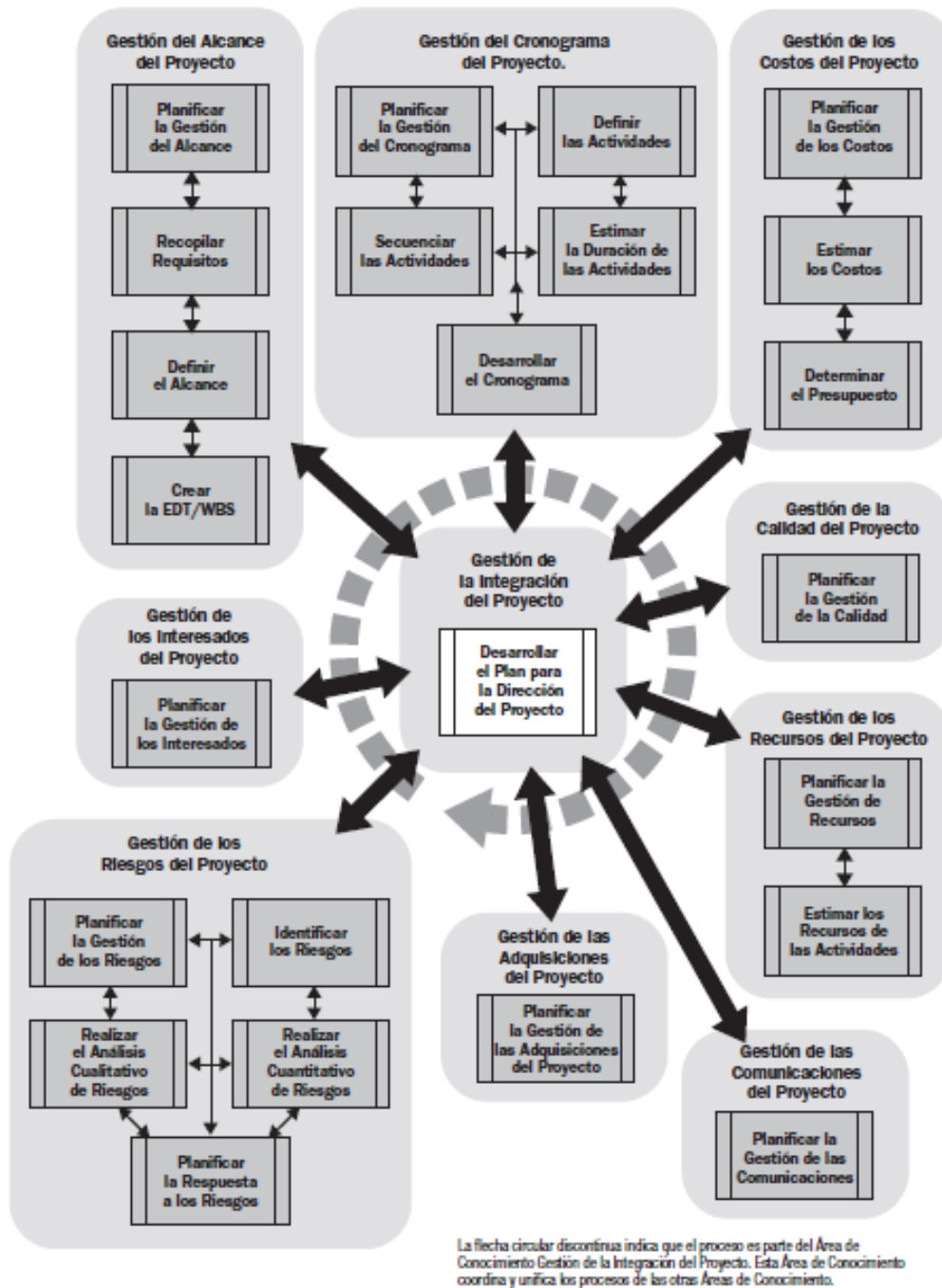


Figura 2.2. Grupo de procesos de planificación. Nota: Tomada de PMBOK (PMI, 2021).

- **Gestión del cronograma del proyecto:** Esta área de conocimiento incluye a los procesos requeridos para administrar, como su nombre lo indica, la finalización del proyecto dentro del tiempo establecido. Dichos procesos son los siguientes:
 - *Planificar la gestión del cronograma.* Este proceso facilita el establecimiento de políticas, procedimientos y la documentación necesaria para definir y gestionar el

cronograma del proyecto. El beneficio principal que se obtiene con este proceder es que se genera la guía y dirección adecuada sobre la gestión del cronograma a lo largo del proyecto. Este proceso se realiza una sola vez o en puntos específicos predefinidos del proyecto.

- *Definir las actividades.* Este proceso establece la necesidad de identificar y documentar las acciones específicas necesarias para elaborar los entregables del proyecto. Su beneficio clave es descomponer los paquetes de trabajo en actividades del cronograma, lo que proporciona una base para la estimación, programación, ejecución, monitoreo y control del trabajo del proyecto. A diferencia de los anteriores, este proceso se lleva a cabo a lo largo de todo el proyecto.
- *Secuenciar las actividades.* Este proceso permite identificar y documentar las relaciones que existen entre las actividades del proyecto. El beneficio principal que se obtiene de su realización es la definición de una secuencia lógica de trabajo que maximiza la eficiencia, considerando todas las restricciones del proyecto. Este proceso también se realiza a lo largo de todo el proyecto.
- *Estimar la duración de las actividades.* Este proceso facilita la estimación de la cantidad de períodos de trabajo necesarios para completar las actividades individuales considerando los recursos estimados. Su beneficio principal es establecer la cantidad de tiempo requerido para finalizar cada actividad. Este proceso se realiza a lo largo de todo el proyecto.
- *Desarrollar el cronograma.* Este proceso permite analizar secuencias de actividades, duraciones, requisitos de recursos y restricciones del cronograma con el objetivo de crear un modelo de cronograma que guíe la ejecución, monitorización y control del proyecto. El beneficio clave de hacerlo es generar un modelo de calendarización con fechas planificadas para completar las actividades del proyecto. Igual que los anteriores, este proceso se realiza a lo largo de todo el proyecto. Cabe mencionar que como parte de este proceso es que se realiza la estimación del esfuerzo una vez que el cronograma se ha definido.
- **Gestión de los costos del proyecto:** Esta área de conocimiento involucra a los procesos requeridos para planificar, estimar, presupuestar, financiar, obtener financiamiento, gestionar y controlar los costos, de modo que se complete el proyecto dentro del presupuesto aprobado. Tales procesos se definen de la siguiente manera:
 - *Planificar la gestión de los costos.* Este proceso define cómo se estimarán, presupuestarán, gestionarán, monitorearán y controlarán los costos del proyecto. El beneficio más importante de esto es que se genera la guía y dirección sobre cómo se gestionarán los costos del proyecto a lo largo del mismo. El proceso se realiza una sola vez o en puntos predefinidos del proyecto.
 - *Estimar los costos.* Este proceso facilita el desarrollo de una aproximación de los recursos monetarios requeridos para completar el trabajo del proyecto. Su beneficio clave es que así se determinan los recursos monetarios requeridos para el proyecto. Este proceso se lleva a cabo periódicamente a lo largo del proyecto, según sea necesario.
 - *Determinar el presupuesto.* Este proceso establece la necesidad de sumar los costos estimados para las actividades individuales o paquetes de trabajo con el fin de establecer una línea base de costos autorizada. El beneficio directo de hacerlo radica en que se determina una línea base de costos, que servirá como referencia para

monitorizar y controlar el desempeño del proyecto. Este proceso se realiza una sola vez o en puntos predefinidos del proyecto.

- Gestión de la calidad del proyecto: Esta área de conocimiento incluye a los procesos que permiten incorporar la política de calidad de la organización en cuanto a la planificación, gestión y control de los requisitos de calidad del proyecto y el producto, a fin de satisfacer las expectativas de los interesados. Para lograr este cometido, esta área define a un solo proceso que se define como:
 - *Planificar la gestión de la calidad.* Este proceso establece la necesidad de identificar los requisitos y/o estándares de calidad para el proyecto y sus entregables, y de documentar cómo el proyecto demostrará su cumplimiento. El beneficio de hacerlo es que se obtiene la guía y dirección sobre cómo se gestionará y verificará la calidad a lo largo del proyecto. De igual manera, este proceso se realiza una sola vez o en puntos predefinidos del proyecto.
- Gestión de los recursos del proyecto: Esta área de conocimiento incluye a los procesos para identificar, adquirir y gestionar los recursos necesarios para la conclusión exitosa del proyecto. Estos procesos son:
 - *Planificar la gestión de los recursos.* Es el proceso que define cómo estimar, adquirir, gestionar y utilizar los recursos físicos y del equipo. Su beneficio clave es establecer el enfoque y nivel de esfuerzo de gestión necesarios para manejar los recursos del proyecto según el tipo y complejidad del proyecto. Este proceso se lleva a cabo una sola vez o en puntos predefinidos del proyecto.
 - *Estimar los recursos de las actividades.* Este proceso facilita la estimación de los recursos del equipo, materiales, equipamiento y suministros necesarios para ejecutar el trabajo del proyecto. El principal beneficio de realizar dicha estimación es identificar el tipo, cantidad y características de los recursos necesarios para completar el proyecto. Este proceso también se lleva a cabo periódicamente a lo largo del proyecto, según sea necesario.
- Gestión de las comunicaciones del proyecto: Esta área de conocimiento incluye a los procesos necesarios para garantizar que la planificación, recopilación, creación, distribución, almacenamiento, recuperación, gestión, control, monitorización y disposición final de la información del proyecto sean oportunos y adecuados. El único proceso para realizar esta función se define de la siguiente manera:
 - *Planificar la gestión de las comunicaciones.* Es el proceso que permite desarrollar el enfoque y plan apropiados para las actividades de comunicación del proyecto, considerando las necesidades de información de los interesados, los activos de la organización y las necesidades del proyecto. El beneficio más significativo de tomar en cuenta esto es que se logra obtener un enfoque documentado para involucrar a los interesados de manera eficaz y eficiente mediante la presentación oportuna de información relevante. Este proceso se lleva a cabo periódicamente a lo largo del proyecto, según sea necesario.
- Gestión de los riesgos del proyecto: Esta área de conocimiento incluye a los procesos que permiten planificar la gestión, identificar, analizar, planificar e implementar una respuesta, y monitorizar los riesgos del proyecto. Los procesos mencionados se definen como:
 - *Planificar la gestión de los riesgos.* Este proceso define cómo llevar a cabo las actividades de la gestión de riesgos de un proyecto. De esta manera, se garantiza que

el nivel, tipo y visibilidad de la gestión de riesgos sean proporcionales a los riesgos como a la importancia del proyecto para la organización y otros interesados. Este proceso se realiza una sola vez o en puntos predefinidos del proyecto.

- *Identificar los riesgos.* Este proceso permite identificar los riesgos individuales del proyecto y las fuentes generales de los mismos, al mismo tiempo que se documentan sus características. Su beneficio clave es la documentación de los riesgos individuales y fuentes de riesgo del proyecto, proporcionando así información para que el equipo del proyecto pueda responder oportuna y adecuadamente a los riesgos identificados. Este proceso se lleva a cabo a lo largo de todo el proyecto.
- *Realizar el análisis cualitativo de los riesgos.* Este proceso facilita la priorización de los riesgos del proyecto para un análisis o acción posterior, evaluando la probabilidad de ocurrencia, impacto y otras características. El principal beneficio de hacer esto es que se logra enfocar los esfuerzos en los riesgos de alta prioridad. Este proceso también se realiza a lo largo de todo el proyecto.
- *Realizar el análisis cuantitativo de los riesgos.* Este proceso establece la necesidad de analizar numéricamente el efecto combinado de los riesgos individuales del proyecto y otras fuentes de incertidumbre sobre los objetivos generales del proyecto. Con esta acción es posible beneficiarse de la cuantificación de la exposición del proyecto a los riesgos y proporcionar información cuantitativa adicional sobre éstos para respaldar la planificación de la respuesta. De igual manera, el proceso se realiza a lo largo de todo el proyecto.
- *Planificar la respuesta a los riesgos.* Este proceso permite desarrollar opciones, seleccionar estrategias y acordar acciones para abordar la exposición al riesgo del proyecto en general y los riesgos individuales. Así, el beneficio más importante se relaciona con la identificación de formas adecuadas de enfrentar los riesgos y asignar recursos, incorporando acciones en la documentación del proyecto y en el plan para la dirección del proyecto, según sea necesario. Este proceso se lleva a cabo a lo largo de todo el proyecto.
- **Gestión de las adquisiciones del proyecto:** Esta área de conocimiento involucra a los procesos requeridos para comprar o adquirir productos, servicios y/o resultados que es preciso obtener fuera del entorno del proyecto. Para realizar esta tarea se define el siguiente proceso:
 - *Planificar la gestión de las adquisiciones.* Este proceso permite documentar las decisiones sobre las adquisiciones del proyecto, especificar el enfoque e identificar a los proveedores potenciales. El beneficio que se obtiene con esto es que se logra determinar si es necesario adquirir bienes y servicios externos, y si este fuera el caso, se sabrá qué adquirir, cómo y cuándo hacerlo. Este proceso se realiza una sola vez o en puntos predefinidos del proyecto.
- **Gestión de los interesados del proyecto:** Esta área de conocimiento incluye a los procesos necesarios para identificar a las personas, grupos u organizaciones que pueden afectar o ser afectados por el proyecto; para analizar las expectativas de los interesados y su impacto en el proyecto; y para desarrollar las estrategias de gestión adecuadas a fin de lograr la participación eficaz de los interesados en las decisiones y en la ejecución del proyecto. El proceso que permite realizar esta tarea define como:
 - *Planificar el involucramiento de los interesados.* Este proceso facilita el desarrollo de enfoques para involucrar a los interesados del proyecto, basándose en sus necesidades,

expectativas, intereses y el posible impacto en el proyecto. El principal beneficio que se obtiene con esto es proporcionar un plan factible para interactuar de manera efectiva con los interesados. Este proceso también se lleva a cabo periódicamente a lo largo del proyecto, según sea necesario.

- Gestión de la integración del proyecto: Finalmente, esta área de conocimiento incluye a los procesos y actividades para identificar, definir, combinar, unificar y coordinar los diversos procesos y actividades de gestión del proyecto dentro del grupo de procesos de la dirección de proyectos. Para lograr este cometido, el área de conocimiento involucra a un solo proceso:
 - *Desarrollar el plan para la dirección del proyecto.* Este proceso implica el definir, preparar y coordinar todos los componentes del plan y consolidarlos en un documento integral. Así, el objetivo principal es producir un plan completo que sirva como base para todo el trabajo del proyecto y establezca cómo se llevará a cabo dicho trabajo. Este proceso se realiza una sola vez o en puntos específicos previamente definidos del proyecto. El proceso recibe como entrada el acta de constitución del proyecto, las salidas de otros procesos de inicio, factores ambientales de la empresa y activos de los procesos de la organización; mientras que su principal salida es el plan de dirección del proyecto.

Como se puede observar, el grupo de procesos de planificación contempla la mayoría de las actividades cruciales para la definición de un proyecto, ya que establece que debe promoverse una relación estrecha entre las áreas de conocimiento para mantener el proyecto dentro de los parámetros planeados y vigilar, mediante los procesos de ejecución y monitorización y control, el avance del proyecto (PMI, 2022). En este sentido, durante la ejecución del proyecto, las estimaciones juegan un papel crucial puesto que sirven como puntos de referencia para comparar el progreso real del proyecto contra las expectativas iniciales establecidas durante su planificación. De acuerdo con Obondi (2022), tales estimaciones proporcionan una guía para evaluar si el proyecto avanza según lo previsto y si se están utilizando los recursos de manera eficiente. Además, permiten al equipo tomar medidas correctivas sobre la marcha en caso de detectar desviaciones significativas entre los resultados reales y las estimaciones iniciales, lo que ayuda a mantener el proyecto en el camino correcto mientras que se minimizan los efectos de los riesgos para evitar desviaciones mayores (Bhardwaj y Arora, 2021). Por otro lado, Pinto (2020) considera que el proceso de monitorización implica la evaluación continua y periódica de la adherencia a los planes establecidos mediante el análisis de los resultados y las condiciones de finalización de cada tarea, así como con la evaluación de los entregables de acuerdo con las características requeridas a través de revisiones y auditorías. Durante este proceso, se investigan los gastos de esfuerzo, el cumplimiento del cronograma, los costos y el uso de recursos, y se revisa el perfil de riesgo del proyecto para identificar posibles desviaciones. Considerando lo anterior, Laplante y Kassab (2022) argumentan que es necesario modelar y analizar los datos de la medición para identificar desviaciones en los resultados esperados, lo que proporcionaría una base sólida para tomar decisiones en el proyecto y realizar los ajustes necesarios para garantizar su éxito. Aunado a esto, herramientas como los informes de estado, la cadena crítica de Goldratt, los diagramas de Gantt, los hitos y la gestión del valor ganado ayudan en la monitorización y control del proyecto (Barghoth et al., 2020).

En este sentido, la monitorización, el control y la medición del proyecto requieren de la estimación. Las estimaciones se llevan a cabo utilizando diferentes técnicas y herramientas, dependiendo del tipo de proyecto y de los recursos disponibles. Algunas técnicas comunes para llevarla a cabo incluyen la estimación analógica, la estimación paramétrica, la estimación de tres

puntos y la estimación basada en expertos. Khan et al., (2021) consideraron que estas técnicas implican la evaluación de los datos disponibles, la realización de análisis comparativos, el uso de modelos matemáticos y la recopilación de opiniones de expertos para determinar los recursos, costos y tiempo necesarios para completar las actividades del proyecto. Por lo tanto, las estimaciones son fundamentales para lograr la planificación y gestión eficaz del proyecto puesto que proporcionan una base sólida para la toma de decisiones, facilitando así a los líderes de proyectos e interesados la visualización de una idea clara de los recursos y costos involucrados en el proyecto (Rashid et al., 2023). Además, las estimaciones también ayudan a identificar posibles riesgos y desviaciones antes de que éstas ocurran, lo que permite una gestión proactiva de los problemas y una mitigación de los riesgos. Cabe mencionar que las estimaciones se realizan en diferentes etapas del proyecto, desde la fase de planificación hasta la ejecución y la monitorización y control de este. En la fase de planificación inicial, por ejemplo, se realizan estimaciones preliminares para facilitar el establecimiento del alcance, objetivos y presupuesto del proyecto. A medida que el proyecto avanza, las estimaciones se refinan y actualizan para tener en cuenta nuevos datos, cambios en los requisitos y lecciones aprendidas. Es decir, es posible realizar nuevas estimaciones a lo largo del proyecto con el fin de ajustar los planes y pronósticos en función de la información disponible más reciente (Babaei et al., 2022). La investigación de Hill (2010) destacó que existían dos enfoques principales para realizar la estimación del software, los cuales continúan vigentes en la actualidad: macro (i.e., de arriba hacia abajo o también llamado paramétrico) y micro (i.e., de abajo hacia arriba o también conocido como basado en tareas). Cada enfoque involucra a varias técnicas de estimación que pueden utilizarse, o incluso combinarse, en cualquier momento del ciclo de vida del proyecto bajo la premisa de que la precisión de las estimaciones resultantes deberá coincidir con la precisión de las entradas utilizadas. Sin embargo, también es importante ser consciente de que todas las técnicas macro pueden experimentar dificultades con proyectos de tamaño pequeño debido a que se presenta una mayor variación en las relaciones de tamaño, esfuerzo y duración en los proyectos (Filippetto et al., 2021).

Alternativamente, un enfoque ágil de gestión se basa en iteraciones cortas de entrega y el aprendizaje continuo. De hecho, Dybå et al., (2014) afirmaron hace más una década que el cambio de un enfoque tradicional de gestión a otro ágil implicaba un cambio que debía iniciar con la planificación, puesto que es el proceso crucial para que la toma de decisiones tenga sentido a lo largo de la ejecución del proyecto. Este cambio, además, debería de apoyarse en la creatividad de las personas, en lugar de los procesos estándar, dadas la complejidad e imprevisibilidad de los proyectos, promoviendo la cultura del desarrollo compartido basada en la toma de decisiones y la autogestión de los equipos de software. De esta manera, al inicio del proyecto, el equipo realiza una planificación simplificada, define requisitos y diseña soluciones para iniciar el proceso (véase la Figura 2.2). Posteriormente, el equipo se involucra en oleadas sucesivas de iteraciones que incluyen la planificación, el análisis de los requisitos, el diseño, la ejecución, las pruebas y la entrega detallada a clientes y partes interesadas. De acuerdo con Salameh (2014), este ciclo iterativo permite una adaptación continua y una mejor alineación con las necesidades del proyecto a medida que éste avanza.

En este sentido, Marnada et al., (2022) afirmaron que la planificación de un proyecto ágil implica la colaboración estrecha con el cliente para desarrollar una visión clara del producto, establecer la frecuencia de las iteraciones y planificar la entrega de las capacidades evolutivas a los usuarios. A diferencia de otros modelos de desarrollo iterativo, los desarrolladores deben crear una metáfora de diseño común, y la versión específica del proceso ágil que se utilizará debe ser revisada y aceptada por todas las partes interesadas del proyecto. Mientras que un modelo tradicional, durante la ejecución del proyecto, es esencial realizar revisiones por iteración con los clientes, desarrolladores

y otras partes interesadas para evaluar el estado actual del producto, los lanzamientos programados, la visión del producto y la metáfora del diseño, así como para alinear los planes futuros (Agbejule y Lehtineva, 2022).

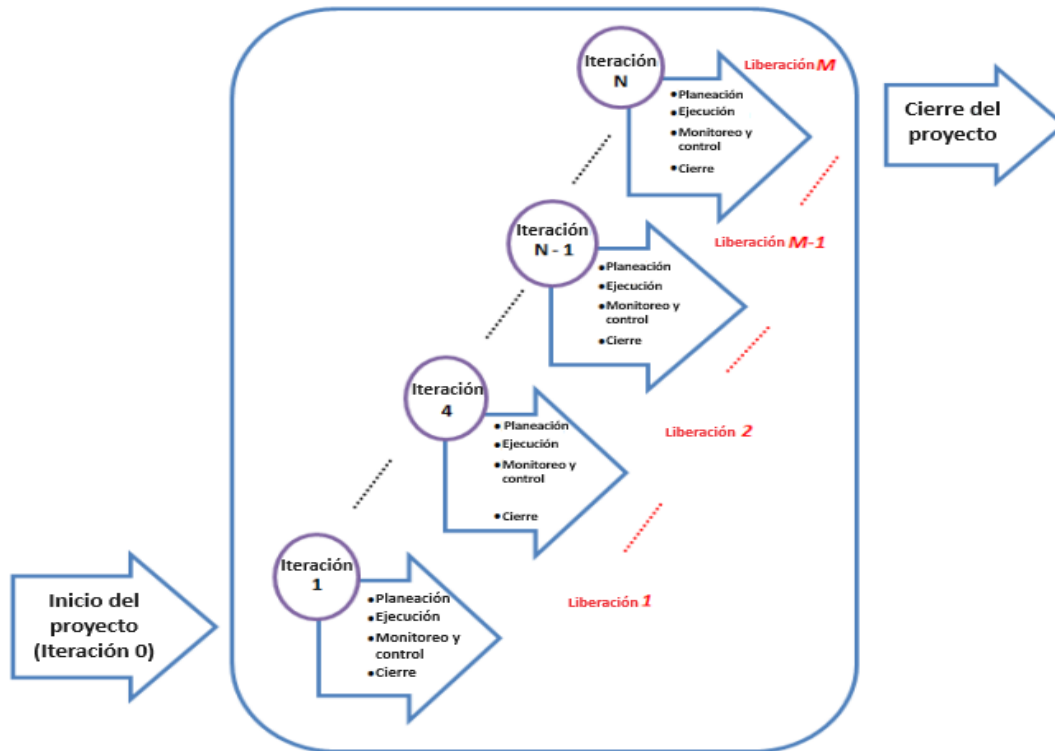


Figura 2.3. Procesos de la gestión ágil de proyectos. Nota: Traducida de Salameh (2014).

Considerando lo anterior, la investigación de Sandeep et al., (2022) argumentó que, en los proyectos ágiles, la estimación del esfuerzo y la planificación se realizan de manera diferente a los enfoques tradicionales. Una técnica comúnmente utilizada en los métodos ágiles para estimar el trabajo es el uso de los puntos de historia. Los puntos de historia son una medida relativa de la complejidad, el esfuerzo y el tiempo necesario para completar una historia de usuario o una tarea (Mallidi y Sharma, 2021). Por lo tanto, en lugar de estimar el tiempo real que tomará completar una tarea, los equipos ágiles asignan puntos de historia basándose en la complejidad percibida de la tarea en comparación con otras tareas. Esto permite una estimación más flexible y adaptable que se ajusta a la incertidumbre inherente en los proyectos ágiles. Los puntos de historia suelen asignarse en una escala relativa, como por ejemplo una secuencia de Fibonacci (e. g., 1, 2, 3, 5, 8, 13, etc.), donde los números más altos representan tareas más complejas. Así, el equipo revisa durante la planificación todas las historias de usuario y asigna puntos de historia basándose en discusiones y consenso (experiencia) del equipo, sobre la complejidad relativa de cada tarea. Una vez que se han asignado los puntos de historia, el equipo puede estimar la cantidad de trabajo que puede completar en una iteración determinada, utilizando el historial de velocidades pasadas como guía. La velocidad del equipo, expresada en puntos de historia por iteración, se utiliza para planificar cuánto trabajo se puede incluir en cada iteración futura. De acuerdo con Iqbal et al. (2024), el uso de puntos de historia permite una mayor flexibilidad en la planificación y la gestión del proyecto, ya que se centra en la complejidad y el valor de las tareas en lugar de en el tiempo real necesario para completarlas. Esto facilita en la práctica la adaptación a medida que cambian los requisitos o surgen nuevos desafíos durante el desarrollo del proyecto.

La investigación realizada por Zasa et al., (2020) evidenció que la planificación y la estimación del esfuerzo son fundamentales en proyectos de software que siguen un enfoque tradicional de gestión. Esto implica que es imperante la identificación de fases e hitos del proyecto, así como de las tareas específicas para cada fase. Las tareas se evalúan considerando factores como calidad y complejidad, y se estima el tamaño y la productividad requeridos para completarlas. Por ejemplo, para la codificación de un componente se considera su tamaño y la experiencia del desarrollador. Estas estimaciones se aplican a todas las tareas del proyecto para calcular así la duración total. Por lo tanto, es esencial calcular el esfuerzo para las diferentes fases del ciclo de vida del software. Por otro lado, los métodos ágiles, iterativos y extremos difieren significativamente del enfoque tradicional, ya que implican iteraciones a lo largo de una o varias fases del ciclo de vida del desarrollo de software (SDLC, por sus siglas en inglés). Es decir, la estimación del esfuerzo y del cronograma para estos proyectos varía en comparación con un enfoque tradicional. De acuerdo con Sudarmaningtyas y Mohamed (2021), el enfoque ágil de gestión es más adecuado para proyectos con alto riesgo de desarrollo de software, especialmente aquellos que implican innovación y creatividad. Inicialmente, se adoptó para proyectos con productos de software pequeños o requisitos no típicos. En tales casos, los contratos basados en el tiempo y el material eran más apropiados que los basados en precio y una duración fija. No obstante, a medida que los modelos iterativos han madurado, se están utilizando para una variedad creciente de proyectos, incluso para proyectos grandes (Butt et al., 2021a).

Como se puede observar, independientemente del enfoque que se siga en la gestión, la estimación en los proyectos de software representa un desafío continuo debido a la naturaleza cambiante de sus requisitos y sus condiciones. Las investigaciones de Fernández-Diego et al., (2020), Alsaadi y Saeedi (2022) y Rodríguez Sánchez et al. (2023), por ejemplo, han destacado la importancia de obtener una alta precisión en la predicción del esfuerzo, ya que esta está fuertemente correlacionada con el éxito y la calidad del producto final. De hecho, Cao (2022) argumentó que cuando las estimaciones son inexactas, se producen dos resultados negativos principales: las subestimaciones, por un lado, pueden llevar a la terminación prematura de proyectos, ya que hacen que se excedan los presupuestos y los cronogramas previstos; las sobreestimaciones, por otro lado, pueden resultar en el desperdicio de recursos, ya que se asignan más recursos de los necesarios para completar el proyecto. Por lo tanto, dada la importancia de una estimación precisa del esfuerzo, los equipos de desarrollo de software deben emplear técnicas y enfoques adecuados para mejorar la precisión de sus estimaciones (Jadhav et al., 2022). Esto puede incluir el uso de métodos ágiles, como la asignación de los puntos de historia, y la revisión y ajuste continuo de las estimaciones a medida que avanza el proyecto. La capacidad para adaptarse a medida que surgen nuevos desafíos y cambios en los requisitos es crucial para minimizar el riesgo de la subestimación o sobreestimación del esfuerzo requerido para completar el proyecto con éxito (Rodríguez et al., 2023).

2.2. Metodologías para la gestión y desarrollo de un proyecto de software

En la gestión y desarrollo de los proyectos, tanto en el ámbito de la Ingeniería de Software como en la Ciencia de Datos, la adopción de metodologías adecuadas es crucial para asegurar una planificación estructurada, una ejecución eficiente y la entrega de productos o resultados de alta calidad (Amirian et al., 2024). Estas metodologías ofrecen marcos de trabajo que no solo organizan el proceso, sino que también promueven la colaboración entre los equipos, mejoran la capacidad de adaptación a cambios, y optimizan el uso de los recursos. A continuación, se describen dos metodologías fundamentales en estos campos: Scrum, un marco ágil popular en el ámbito del desarrollo ágil de software y CRISP-DM, una metodología ampliamente utilizada en proyectos de Minería y Ciencia de Datos.

2.2.1. La metodología Scrum

De acuerdo con la guía creada por Ken Schwaber y Jeff Sutherland en 2010 (Schwaber y Sutherland, 2020), Scrum es un marco de trabajo diseñado para ayudar a los especialistas del software a encontrar soluciones efectivas a proyectos complejos e inciertos, caracterizados por su gran tamaño o requisitos cambiantes (véase Figura 2.4). Dado este nivel de incertidumbre, Scrum no impone procesos ni herramientas específicas, sino que se basa en la teoría del empirismo y el pensamiento *Lean* (filosofía que pretende maximizar el valor de los productos y eliminar los desperdicios que hay en el desarrollo de ellos) para tomar decisiones fundamentadas en la experiencia y centrarse así en lo esencial. Este enfoque ágil promueve la autoorganización del equipo y la mejora continua, facilitando la entrega constante de valor a través de ciclos cortos de trabajo denominados *sprints*. En cada *sprint*, el Equipo Scrum (Scrum Team), compuesto por el Propietario del producto (Product Owner), el Maestro Scrum (Scrum Master) y el Equipo de desarrollo (Development Team), participa en eventos clave como la reunión diaria (Daily Scrum), la reunión de Planificación del Sprint (Sprint Planning meeting), la reunión de revisión (Sprint Review) y la reunión de Retrospectiva del Sprint (Sprint Retrospective). Estos eventos permiten al equipo colaborar de manera efectiva y adaptarse a los cambios. Además, se utilizan artefactos como la Pila del Producto (Product Backlog) y la Pila del Sprint (Sprint Backlog) para gestionar el trabajo y el incremento (Increment), asegurando que se alineen con los pilares de transparencia, inspección y adaptación, así como con los objetivos del proyecto. Los valores fundamentales de Scrum, que incluyen al compromiso, coraje, apertura y respeto, son esenciales para fomentar un ambiente de trabajo positivo y productivo.

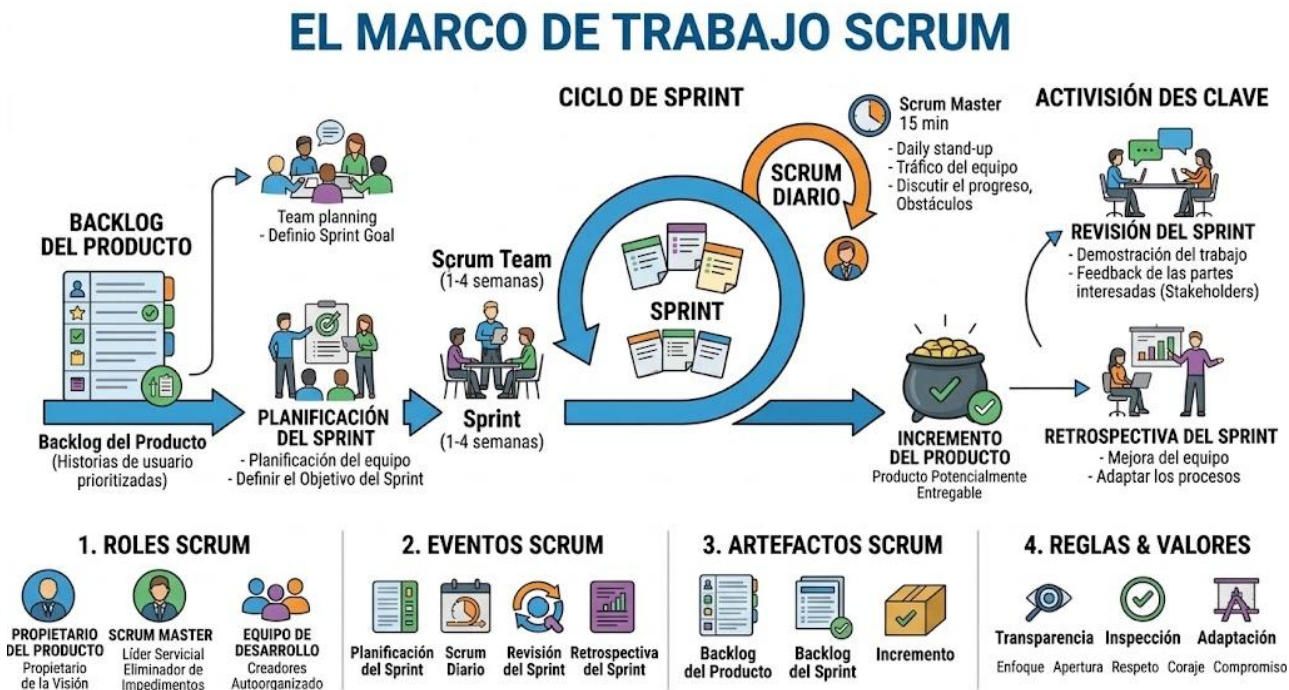


Figura 2.4. Representación gráfica de Scrum. Nota: Traducida de Schwaber y Sutherland (2021).

El Equipo Scrum (ST, por sus siglas en inglés) unifica a un pequeño grupo de personas mediante tres roles esenciales ya antes mencionados, el cual incluye al PO, el Maestro Scrum y el Equipo de desarrollo. Este grupo tiene que ser multifuncional, es decir, debe contar con todas las habilidades necesarias para llevar a cabo el trabajo requerido y, además, debe ser autogestionado, lo que implica que se decide internamente quién hace qué, cómo y cuándo. El equipo debe estar compuesto por un

número reducido de personas, típicamente no más de 10, con el fin de fomentar la agilidad, una comunicación efectiva y la adaptabilidad. Equipos demasiado grandes pueden volverse difíciles de gestionar, por lo que, si es necesario, se puede considerar la creación de varios equipos Scrum trabajando en un mismo producto, y compartan al PO y la Pila del producto (PB, por sus siglas en inglés). El PO es el responsable de maximizar el valor del producto resultante del trabajo del equipo. Este rol es ocupado por una sola persona, no por un grupo o comité, y su principal tarea es gestionar y priorizar el PB. El PO trabaja en estrecha colaboración con los *stakeholders* (i.e., partes interesadas) y con el Equipo de desarrollo, asegurando que se aborden las funcionalidades más importantes o que generen más valor de acuerdo con las necesidades del negocio. Su responsabilidad es definir claramente las tareas y priorizarlas de manera que el equipo se concentre en lo más valioso durante cada *sprint*.

El SM desempeña un papel fundamental en la implementación y correcta aplicación de Scrum dentro del equipo y la organización. Como facilitador y no como líder tradicional, su principal responsabilidad es eliminar obstáculos que puedan interferir en el desempeño del equipo, garantizando que se sigan los principios de Scrum y que el equipo pueda optimizar su rendimiento. De esta manera se fomenta la autoorganización, se promueve una cultura de aprendizaje y se facilita la realización de eventos claves como reuniones diarias (DS, por sus siglas en inglés), la reunión de Planificación del Sprint, la reunión de revisión (SR, por sus siglas en inglés) y la reunión de retrospectiva del *sprint* (SRP, por sus siglas en inglés). Al actuar como un defensor del marco ágil, el SM se asegura de que el equipo mantenga un ritmo de trabajo sostenible y efectivo, lo que contribuye a la creación de un entorno colaborativo y proactivo. Su influencia se extiende más allá del equipo, educando a los *stakeholders* sobre Scrum y garantizando que todos los involucrados trabajen en sinergia para lograr los objetivos del proyecto. Por otro lado, el Equipo de desarrollo es un grupo multifuncional que tiene la capacidad de transformar una funcionalidad en un incremento del producto. El equipo decide cómo abordar y completar el trabajo necesario para cada *sprint* y es responsable de gestionar la Pila del Sprint (SB, por sus siglas en inglés), que contiene las tareas seleccionadas para dicho *sprint*. Este equipo debe estar compuesto por profesionales que, en conjunto, posean todas las habilidades necesarias para cumplir con los objetivos del *sprint*, ya sea en áreas de desarrollo, diseño, pruebas u otros aspectos técnicos. Los desarrolladores trabajan de manera autónoma, distribuyen sus tareas y colaboran para alcanzar la meta de cada *sprint*, que siempre debe ser un producto potencialmente entregable.

Los roles anteriores participan en eventos clave que aseguran la inspección y la adaptación del proceso. Éstos incluyen al evento contenedor *sprint*, la reunión de Planificación del Sprint (donde se decide qué trabajo se realizará), las DS (que permiten al equipo sincronizar sus esfuerzos), la SR (donde se presenta el incremento del producto a los *stakeholders*) y la SRP (en la que el equipo reflexiona sobre cómo mejorar su forma de trabajar). Por lo tanto, el *sprint* es el núcleo de Scrum, un ciclo de trabajo que dura de una a cuatro semanas como máximo. Durante este tiempo, el equipo se enfoca en desarrollar un incremento del producto que sea funcional y esté listo para ser evaluado. Cada *sprint* comienza con una planificación y finaliza con una retrospectiva, creando un ciclo continuo de mejora. La reunión de Planificación del Sprint marca el comienzo de cada *sprint*. En él, el PO y el Equipo de desarrollo colaboran para identificar las tareas del PB que se abordarán. Además, se definen los objetivos del *sprint* y se estima el esfuerzo necesario para completar las tareas seleccionadas, asegurando que todos estén alineados con el enfoque de trabajo. Mientras que el SM debe asegurarse que la reunión se mantenga dentro de la caja de tiempo de máximo ocho horas, se defina el plan que se desarrollará durante todo el *sprint*, se comprendan y refinen cada uno de los elementos seleccionados del PB y que formarán parte del SB. Aunado a esto, la reunión diaria es una reunión breve, con duración máxima de quince minutos, donde el equipo comparte lo que ha logrado desde la

última reunión, lo que planea hacer a continuación y cualquier impedimento que esté enfrentando. Esta práctica permite ajustar el enfoque diario y mantener la alineación con los objetivos del *sprint*. El SM y el PO no participan activamente, pero a petición del Equipo de desarrollo pueden hacerlo. En el SR, antes de finalizar el *sprint*, se reúnen el ST y los *stakeholders* para evaluar el incremento del producto. El Equipo de desarrollo muestra los resultados alcanzados, los problemas que tuvieron y cómo los solucionaron, y se recopila realimentación en caso de que existan nuevas modificaciones o mejoras. Esta reunión es esencial para adaptar y ajustar el enfoque de trabajo en función de los comentarios recibidos, en ella, el PO puede informar al patrocinador sobre los recursos utilizados, la reunión debe durar máximo cuatro horas. De manera similar, la SRP se lleva a cabo después de la SR y antes de la planificación del siguiente *sprint*. En este sentido, el equipo reflexiona sobre el trabajo realizado, identifica aspectos positivos y áreas de mejora, y define acciones concretas para optimizar su proceso en el futuro. Este evento, que fomenta una cultura de aprendizaje y mejora continua, no debe durar más de tres horas.

Dentro de los eventos es fundamental visualizar el progreso del proyecto, por lo que Scrum define tres artefactos clave. Estos artefactos son herramientas que proporcionan claridad sobre el trabajo realizado y el avance del equipo. Al ofrecer una visión transparente, son esenciales para que tanto el equipo como los *stakeholders* puedan monitorizar el progreso a través de entregas frecuentes y tomar decisiones informadas basadas en la realidad del proyecto. El PB es una lista priorizada de todas las funcionalidades, mejoras y correcciones que se requieren en el producto, es vital porque permite al PO y al Equipo de Desarrollo enfocarse en lo más importante para maximizar el valor entregado. Mantener actualizado el PB garantiza que el equipo trabaje en las prioridades correctas y que se puedan adaptar rápidamente a los cambios en los requisitos del cliente. Por otro lado, el SB es un subconjunto del PB que el equipo selecciona para trabajar en un *sprint* específico. Incluye las tareas que el equipo se compromete a completar durante ese *sprint*. La importancia de la SB radica en que permite a los miembros del equipo organizar su trabajo diario, fomentar la autoorganización y proporcionar un enfoque claro para la entrega de un incremento del producto. Finalmente, el *Incremento* es la suma de todos los elementos del PB completados durante un *sprint* y todos los *sprints* anteriores. Un *Incremento* debe ser potencialmente entregable, lo que significa que cumple con los criterios de calidad y está listo para ser utilizado por el cliente. La entrega de un *Incremento* funcional en cada *sprint* asegura que el equipo esté constantemente proporcionando valor y obtenga realimentación rápida por parte de los *stakeholders*.

Ahora bien, para cada evento y artefacto deben de aplicarse los pilares de Scrum con el fin de obtener los mejores resultados. La transparencia garantiza que todos los aspectos del proceso de desarrollo sean visibles para los participantes. Esta transparencia se logra a través de los artefactos, que permiten que el progreso y el trabajo en curso sean fácilmente comprensibles para todos los involucrados, incluidos los *stakeholders*. Esto fomenta la confianza y asegura que todos los participantes estén alineados con los objetivos del proyecto. La inspección regular del trabajo y los procesos permiten a los equipos identificar cualquier desviación del plan y áreas que necesitan mejora. A través de eventos como el DS, SR y SRP, el equipo puede evaluar su rendimiento y el estado del Incremento. Este pilar es crucial porque proporciona oportunidades para realizar ajustes oportunos y mejorar continuamente. Una vez que se han realizado las inspecciones, el equipo debe ser capaz de adaptarse a los nuevos hallazgos. Esto implica la necesidad de hacer cambios en el PB, el SB y el proceso en sí para abordar cualquier problema identificado. Como se podrá observar, la capacidad de adaptarse rápidamente a los cambios en las condiciones o requisitos del proyecto es fundamental en un entorno ágil, ya que maximiza la efectividad del equipo y la satisfacción del cliente.

En esencia, Scrum no dicta procesos, herramientas ni prácticas específicas, sino que proporciona una estructura que permite la adaptación a los cambios, promoviendo la transparencia y la inspección

a lo largo de todo el proceso de desarrollo. Esta flexibilidad hace de Scrum un marco ideal para gestionar proyectos con un alto nivel de incertidumbre, proporcionando un camino claro hacia soluciones adaptativas y efectivas.

2.2.2. CRISP-DM

A diferencia de Scrum, el Proceso Estándar Intersectorial para la Minería de Datos (CRISP-DM, por sus siglas en inglés) es un modelo ampliamente utilizado en proyectos de Minería de Datos que se compone de seis fases iterativas: comprensión del negocio, comprensión de los datos, preparación de los datos, modelado, evaluación y despliegue. Este enfoque flexible permite al equipo avanzar y retroceder entre fases según sea necesario, ajustando el proceso en función de los resultados y definiendo hitos clave. Su adaptabilidad lo convierte en el método estándar más utilizado para el descubrimiento de conocimiento en diversas industrias (Saltz, 2021). La Figura 2.5 muestra que el proceso de CRISP-DM no sigue una secuencia rígida, lo que permite flexibilidad en su implementación. Así pues, se trata de un marco completo y bien documentado, con etapas claramente organizadas y definidas, lo que facilita la comprensión o revisión de un proyecto.

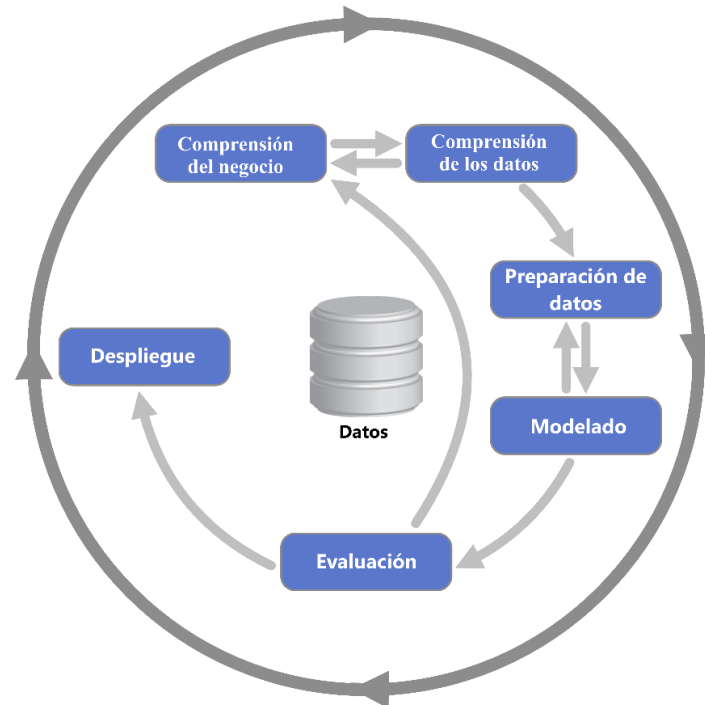


Figura 2.5. Proceso estándar intersectorial para la Minería de Datos. Nota: Traducida de Saltz (2021).

En este sentido, Azevedo y Santos (2008) establecieron que la *Comprensión del Negocio* es crucial para entender los objetivos y requisitos del proyecto desde una perspectiva empresarial. Por lo tanto, se requiere identificar lo que el cliente quiere lograr, evaluar la situación actual, los recursos disponibles, los riesgos y hacer un análisis costo-beneficio. Además, se definen los objetivos técnicos del proyecto y se elabora un plan detallado para guiar su desarrollo. Los entregables claves son la documentación de los objetivos, riesgos y requisitos, así como el plan de proyecto inicial. La *Comprensión de los Datos*, por otro lado, se enfoca en identificar, recopilar y analizar los datos relevantes para el proyecto. Las tareas incluyen recopilar los datos necesarios, describir sus propiedades (e.g., como formato y registros), explorar las relaciones entre ellos y verificar su calidad. Los entregables son informes que documentan la recopilación, descripción, exploración y calidad de

los datos. Esta etapa es fundamental para garantizar que los datos sean adecuados para los objetivos del proyecto y permitan avanzar de manera precisa en las etapas posteriores del ciclo. La *Preparación de Datos* es fundamental para el modelado y puede consumir entre el 50% y el 80% del esfuerzo del proyecto. Regularmente incluye cinco tareas: seleccionar atributos, donde se eligen los atributos a utilizar y se documentan las razones; limpiar datos, que implica corregir o eliminar ciertos registros; construir datos, que consiste en crear nuevos atributos útiles; integrar datos, que combina información de diversas fuentes; y dar formato a los datos, que ajusta los datos según sea necesario. Los entregables incluyen justificaciones, informes de limpieza y documentación sobre la integración y el formato de los datos, asegurando así la calidad necesaria para el modelado.

En la etapa de *Modelado*, considerada una de las más emocionantes en la Ciencia de Datos, el equipo crea y evalúa múltiples modelos utilizando diversas técnicas. Aunque la guía de CRISP-DM recomienda iterar hasta encontrar el “mejor modelo”, en la práctica los equipos suelen buscar un modelo “suficientemente bueno”. Esta etapa incluye cuatro tareas clave: seleccionar técnicas de modelado (i.e., elegir algoritmos como regresión o redes neuronales), generar diseños de prueba (i.e., dividir los datos en conjuntos de entrenamiento, prueba y validación), construir el modelo (i.e., entrenamiento de un algoritmo para obtener un modelo) y evaluar el modelo (i.e., comparar los resultados de distintas métricas de rendimiento y considerar el conocimiento del dominio). Los entregables de esta etapa comprenden un informe de análisis de técnicas, configuraciones de parámetros, un diseño de prueba y una evaluación de los modelos utilizados (Suhendar y Widayani, 2023). En la *Evaluación* se analiza cómo el modelo se alinea con los objetivos comerciales, y no solo se enfoca en su rendimiento técnico. Esta etapa incluye tres tareas principales: evaluar los resultados (i.e., determinar si el modelo cumple con los criterios de éxito empresarial), proceso de revisión (i.e., examinar si se omitieron pasos o si se ejecutaron adecuadamente todas las tareas) y determinar los pasos siguientes (i.e., decidir si se avanza hacia el despliegue, se realizan más iteraciones o se inician nuevos proyectos). Los entregables de esta etapa son una evaluación de los resultados en relación con los criterios de negocio y una lista de posibles pasos a seguir (Karna et al., 2020). Finalmente, el *Despliegue* se centra en la utilización práctica del modelo desarrollado, ya que un modelo carece de valor si no se aplica. Dependiendo del proyecto, el despliegue puede ir desde compartir un informe simple hasta implementar un modelo predictivo en tiempo real. Esta etapa incluye cuatro tareas clave: planificar la implementación (i.e., desarrollar un plan para poner en marcha el modelo), planificar el seguimiento y mantenimiento (i.e., crear un plan para supervisar y mantener el modelo en funcionamiento), elaborar el informe final (i.e., documentar un resumen del proyecto y sus resultados) y realizar una revisión del proyecto (i.e., reflexionar sobre lo que funcionó, lo que se puede mejorar y cómo avanzar en futuros proyectos). Los entregables de esta etapa abarcan un plan de implementación, un plan de monitoreo y mantenimiento, un informe final y una evaluación general del proyecto (Kraut y Transchel, 2022).

2.3. Estimación del esfuerzo en el desarrollo de software

Considerando las metodologías anteriores se puede afirmar que la estimación es una actividad crucial que se lleva a cabo inicialmente durante la fase de planificación de un proyecto. En esta etapa, se realizan evaluaciones cuantitativas de los diversos aspectos del proyecto, como los costos, el tiempo y los recursos necesarios para completarlo, las cuales reciben el nombre de “estimaciones”. Estas estimaciones proporcionan una base sólida para la planificación del proyecto, puesto que son cruciales para establecer expectativas realistas y asignar los recursos de manera eficiente. Sin embargo, es importante comprender que las estimaciones son solo predicciones y pueden estar sujetas a cambios a medida que avanza el proyecto. Por lo tanto, durante las etapas de ejecución y control del proyecto,

es fundamental medir y monitorear continuamente el progreso real en comparación con las estimaciones iniciales.

De acuerdo con la PMBOK (2021), la planificación implica el realizar estimaciones para varios aspectos importantes del proyecto, como el esfuerzo laboral, la duración, los costos y los recursos necesarios. Estas estimaciones están sujetas a cambios que se presentan a lo largo del proyecto a medida que se obtiene más información y cambian las circunstancias. En este sentido, es importante considerar que, durante el ciclo de vida de un proyecto, tales cambios afectan regular y significativamente cuatro aspectos relacionados directamente con la estimación: rango, exactitud, precisión y confianza. Estos aspectos se describen de la siguiente manera:

- El rango de las estimaciones puede variar desde uno amplio al inicio del proyecto, cuando hay poca información disponible, hasta uno más estrecho en etapas avanzadas del proyecto.
- La exactitud se vincula al rango y se refiere a la precisión de lo que se estima, siendo esta menor al inicio del proyecto y mejorando a medida que éste avanza.
- La precisión, por otro lado, se refiere al grado de exactitud asociado con la estimación, siendo una estimación específica más precisa que una estimación general.
- La confianza en las estimaciones aumenta con la experiencia, siendo menor para componentes tecnológicos nuevos y en evolución. La experiencia previa en proyectos similares puede incrementar significativamente la confianza en las estimaciones.

Tomando en cuenta lo anteriormente expuesto, Tsui et al., (2022) argumentaron que las estimaciones que se realizan sobre los proyectos de desarrollo de software son la base para llevar a cabo su planificación, puesto que de esta manera se puede obtener una visión general de, entre otras cosas, los esfuerzos necesarios para el desarrollo de dicho software y el diseño de las variables que influyen positiva o negativamente en el proyecto, como la productividad del personal o la complejidad del dominio. No obstante, es importante mencionar que esta visión sobre la estimación ha cambiado a lo largo del tiempo. Por ejemplo, McConell (2006) definió que una buena estimación era *“aquella que proporciona una visión lo suficientemente clara de la realidad del proyecto como para permitir que el liderazgo se base en la toma de buenas decisiones sobre cómo controlar el proyecto para alcanzar sus objetivos”*. Años más tarde, Cardozzo (2016) argumentó que la estimación del esfuerzo era una actividad esencial en la industria del desarrollo de software, puesto que, ya sea a través de modelos simples basados en el juicio de expertos o modelos más complejos que emplean datos históricos y algoritmos, el objetivo es obtener una estimación precisa del esfuerzo necesario para completar un proyecto, reafirmando que *“la estimación proporciona una base para la planificación y gestión del proyecto, así como para la toma de decisiones en cuanto a costos, recursos, duración y entregables”*. Más recientemente, en la documentación de PMBOK se afirma que *“las estimaciones son una evaluación cuantitativa del valor o resultado probable de una variable, tal como pueden ser los costos del proyecto, los recursos empleados, el esfuerzo requerido o la duración considerada”* (2021).

Ahora bien, se puede establecer que la estimación es fundamental para evaluar de manera sistemática y cuantitativa el alcance y los recursos necesarios para la realización exitosa de un proyecto de desarrollo de software. En este análisis, se pronostican los costos, el esfuerzo requerido y la duración estimada del proyecto, proporcionando así una visión clara y fundamentada que sirve como base para la planificación, gestión y toma de decisiones estratégicas. La estimación permite anticipar y mitigar riesgos potenciales, asignar adecuadamente los recursos disponibles y establecer expectativas realistas tanto para el equipo de trabajo como para las partes interesadas.

2.3.1. Importancia de la estimación del esfuerzo en los proyectos ágiles

La investigación de Huynh Thai et al., (2021) expuso que el desarrollo de software se enfrenta a restricciones tanto técnicas como económicas, las cuales se establecen en un plan de proyecto que comprende actividades y tareas definidas en función de los requisitos establecidos. Por lo tanto, la estimación del esfuerzo requerido para llevar a cabo tales actividades y tareas resulta esencial para lograr una planificación y ejecución exitosa del proyecto. Así, la precisión de las estimaciones tiene una importancia significativa, puesto que éstas impactarán directamente en el éxito del proyecto. La estimación del costo de desarrollo y del esfuerzo requerido, por ejemplo, es una actividad vital en el ciclo de vida del desarrollo de software puesto que resulta fundamental asignar la cantidad necesaria de recursos y determinar la duración del proyecto, respectivamente. Por lo tanto, las estimaciones precisas son cruciales para evitar problemas durante la gestión de los proyectos, tales como la baja calidad del producto y la falta de cumplimiento de los hitos (Azzeh et al., 2022). El proceso de estimación tiene pues como objetivo proporcionar una aproximación de los recursos necesarios para completar un proyecto, garantizando la entrega de productos o servicios de software que correspondan con los requisitos funcionales y no funcionales especificados al inicio (Saif, 2020). Es importante considerar que las estimaciones se generan internamente y se deben revisar periódicamente, pero, además, es necesario asumir que para hacerlas tempranamente se requiere del compromiso entre las partes interesadas para revisarlas y llegar a un acuerdo sobre los recursos y el tiempo necesarios para completar cada proyecto. Dado que medir con precisión una única estimación de esfuerzo puede resultar difícil, es mejor comunicarla en forma de intervalos. No obstante, aunque no existe una conexión directa entre los intervalos de esfuerzo implícitos y el nivel de confianza, un estimador (e.g., el líder del proyecto) debe confiar en que los esfuerzos reales se mantendrán siempre dentro del rango estimado (Sudarmaningtyas y Mohamed, 2021). Por lo tanto, la precisión en las estimaciones juega un papel crucial para evitar desviaciones en el cronograma real del proyecto y/o identificar acciones correctivas en el momento apropiado (Sehra et al., 2014).

Por otro lado, Somerville (2019) argumentó que la importancia de considerar los contratiempos en las estimaciones de un cronograma, por ejemplo, radica en la necesidad de garantizar la viabilidad y el éxito de un proyecto de desarrollo de software. Por lo tanto, al reconocer la posibilidad de que pueden ocurrir problemas durante el avance del proyecto (e.g., enfermedades entre los integrantes del equipo, cambios de personal o fallas en el hardware), los líderes de proyectos pueden prepararse mejor para enfrentar estas eventualidades. Así, al establecer una estimación realista y agregar un margen de contingencia adecuado al proyecto, se está pensando en mitigar el impacto de los contratiempos que se pudieran presentar. De esta manera se estaría realizando una planificación más efectiva, puesto que se evitaría el incumplimiento de plazos y presupuestos y aseguraría la entrega oportuna del software o hardware requerido por los clientes. Además, al considerar los posibles desafíos desde el principio, se fomenta una mayor transparencia y comunicación dentro del equipo y con las partes interesadas en el proyecto (Srivastava et al., 2022). Esto contribuye a mantener expectativas realistas y a gestionar mejor los riesgos asociados al mismo. Sin embargo, Vickler (2021) considera que tanto la planificación como la estimación siguen siendo procesos difíciles de realizar no solamente en los entornos ágiles de desarrollo, puesto que, sin el conocimiento y experiencia requeridas, puede resultar imposible la obtención de una estimación precisa de manera temprana. Entonces, ¿por qué el empeño en realizar correctamente estos procesos? La investigación de Butt et al., (2021b) argumentó que, a pesar de esta dificultad, una variedad de organizaciones requiere de estimaciones para planificar campañas de marketing, programar el lanzamiento de sus productos, calendarizar la capacitación de usuarios internos, o analizar los costos y tiempos para entregar una obra arquitectónica. Estas necesidades son legítimas y evidencian la importancia de generar planes y cronogramas que toda

organización debe utilizar para controlar sus respectivos proyectos. Pero la planificación no se trata solamente de establecer plazos y calendarios de entrega, sino que es una búsqueda de valor. De acuerdo con Tam et al. (2020), la planificación debe ser iterativa e incremental para reducir el riesgo y la incertidumbre, apoyar a los líderes en la toma de decisiones, establecer confianza y transmitir información crucial entre los participantes y partes interesadas del proyecto. A medida que el proyecto avance, es posible que las prioridades cambien o que se requieran ajustes en el alcance y la planificación inicial. No obstante, la planificación iterativa y continua permite adaptarse a estos cambios de manera efectiva y asegura que el equipo esté alineado con los objetivos del proyecto en todo momento.

En este contexto, se ha dicho que un método ágil se enfoca en la codificación, ciclos de entrega cortos y múltiples iteraciones para desarrollar eficientemente el software, pero esto dependerá significativamente de una estimación correcta. En consecuencia, la estimación del esfuerzo en proyectos ágiles constituye un pilar fundamental que sustenta el éxito y la eficacia de todo el proceso. En un entorno ágil de desarrollo, donde la adaptabilidad y la entrega incremental son prioritarias, la capacidad de estimar con precisión el esfuerzo requerido para cada tarea o iteración es crucial. Es decir, la estimación del esfuerzo permitirá la realización de una planificación efectiva puesto que, al tener una idea clara del tiempo y los recursos necesarios para completar cada aspecto del proyecto, los equipos podrán estructurar su trabajo de manera más eficiente y asignar recursos de manera adecuada (Cao, 2022). Además, la estimación del esfuerzo contribuirá a realizar una toma de decisiones más informada. De esta manera, con una comprensión clara de los recursos disponibles y los tiempos de entrega estimados, los líderes de proyectos y las partes interesadas podrán tomar decisiones más acertadas sobre la priorización de las tareas y la asignación de los recursos. La investigación realizada por Rashid et al., (2023) afirma que las estimaciones del esfuerzo proporcionan además una base objetiva para discutir y negociar los planes del proyecto, dado que facilitan la monitorización de su progreso. Así, al comparar las estimaciones iniciales con los resultados reales, los equipos pueden identificar desviaciones tempranas y tomar medidas correctivas según sea necesario para mantener el proyecto en curso. Considerando lo anteriormente expuesto, Sudarmaningtyas y Mohamed (2022) argumentan que la estimación del esfuerzo es crucial para un proyecto ágil, ya que el tiempo de entrega y la velocidad del proyecto se basarán en las estimaciones que de éste se hagan. En este sentido, el proceso de estimación en los proyectos ágiles se divide en dos fases: la estimación temprana y la estimación iterativa. La *estimación temprana* se utiliza para definir el alcance inicial del proyecto, mientras que la *estimación iterativa* anticipa cambios en los requisitos al comienzo de cada iteración. Los métodos principales de estimación en los proyectos ágiles incluyen al aprendizaje computacional y al juicio de expertos. Sin embargo, la elección de un método dependerá en gran medida de la experiencia del estimador y a menudo en el proceso de estimación no se consideran otros factores.

Por otro lado, el estudio realizado por Kuan (2017) mostró que la inversión total en el desarrollo y mantenimiento de software fue excesiva durante la última década. En este sentido, los datos recogidos con dicha investigación mostraron que la sobreestimación del esfuerzo varía significativamente entre un 41% y un 258%, mientras que la sobreestimación de la inversión total oscila entre un 97% y un 151%, afectando negativamente la terminación exitosa de un proyecto. Peor aún, de acuerdo con investigaciones más actuales realizadas por agencias gubernamentales de EE. UU., como la Oficina de Responsabilidad Contable (GAO, por sus siglas en inglés) (GAO, 2023), el Departamento de Defensa (DoD, por sus siglas en inglés) (Rosa et al., 2021; DoD, 2023) y el Subcomité de Cibernética, Tecnologías de la Información e Innovación del Comité de Servicios Armados de la Cámara de Representantes (Patt, 2024), aproximadamente el 60% de los proyectos de software, desarrollados entre 2022 y 2023, superó el tiempo de finalización estimado y el 50% excedió

los costos estimados originalmente. Considerando lo anteriormente expuesto, Matsubara et al., (2022) argumentaron que la estimación del esfuerzo para un proyecto ágil puede verse influenciada negativamente en precisión y efectividad tanto por factores relacionados con el proyecto en sí (e.g., tipo de proyecto, requisitos de calidad, requisitos de hardware y software, facilidad de operación, complejidad, manejo de los datos y las múltiples ubicaciones de estos) como por factores relacionados con las personas (e.g., habilidades de comunicación, familiaridad con el equipo, aptitudes de gestión, seguridad, tiempo de trabajo, experiencia previa en proyectos y habilidades técnicas). Estos factores son presentados y analizados a continuación.

2.3.2. Factores que influyen en la estimación efectiva del esfuerzo

El estudio realizado por Sehra et al., (2016) argumentó que la estimación del esfuerzo era una actividad crítica en la gestión de los proyectos de software, dado que resultaba difícil pensar que una única técnica de estimación sería eficiente para todas las situaciones. Pero además de esto, Noteboom et al., (2021) también consideraron que las organizaciones desarrolladoras de software debían contar con personal que tuviera las habilidades necesarias para realizar las estimaciones y de condiciones que permitieran desarrollar correctamente un proyecto, de tal manera que fuera posible la producción de software de calidad de manera consistente y confiable. Es decir, se hizo evidente la importancia que tenían ya los factores humanos y los factores que pueden afectar al proyecto en la obtención de una buena estimación. Sin embargo, de acuerdo con la literatura especializada, no todos los factores que influyen en la obtención y/o éxito de una estimación en general tienen el mismo impacto en la estimación del esfuerzo de un *sprint*. Ramessur y Nagowah (2021) argumentaron que los factores relacionados con el proyecto, como la transacción de datos, los requisitos de hardware y software, la facilidad de operación, el proceso y la gestión, tienen menos relevancia. En contraste, la experiencia del personal, las habilidades técnicas y la complejidad de los requisitos destacan como los factores más influyentes en todos los proyectos. De acuerdo con Swandari et al. (2023), otros factores relacionados con las personas que deben considerarse para lograr una estimación efectiva del esfuerzo incluyen a la participación de los *stakeholders*, la dinámica y el compromiso del equipo, y las responsabilidades externas al proyecto. Factores como la configuración, la seguridad, los defectos y los cambios en implementaciones anteriores, la calidad y la volatilidad de los requisitos también deben ser tomados en cuenta.

Por otro lado, Fürst et al., (2024) consideran que las habilidades individuales no deberían afectar significativamente la precisión de una estimación del esfuerzo en los proyectos de software, pero coincide con los investigadores anteriores al confirmar que sí influyen en la eficiencia del proceso de estimación. Para llegar a esta conclusión se realizó un estudio que incluyó a estudiantes de maestría, quienes fueron divididos en equipos de tres a cuatro integrantes para evaluar diferentes técnicas ágiles de estimación. Como principal resultado se observó que las técnicas conocidas como “sistema de cubos” y “estimación de afinidad” fueron más eficientes en tiempo que la “planificación de póker”. Además, como dato relevante, se identificó que los equipos cuyos integrantes poseían mejores habilidades demostraron ser más eficientes en el proceso de estimación que el resto de los participantes. De acuerdo con los investigadores, esta diferencia podría atribuirse al efecto *Dunning-Kruger*⁴, dado que los participantes menos hábiles tendieron a sobreestimar su capacidad, generando una mayor fluctuación y, por ende, consumiendo un mayor tiempo para completar la estimación.

⁴ El efecto *Dunning-Kruger* resalta el sesgo cognitivo por el cual las personas con habilidades pobres para realizar una tarea sobrestiman su habilidad. Esta definición también suele incluir al efecto opuesto para las personas de alto rendimiento, es decir su tendencia a subestimar sus habilidades (Dunning, 2011).

Interesantemente, en el caso de las técnicas del sistema de cubos y la estimación de afinidad, los equipos heterogéneos dedicaron menos tiempo que los homogéneos, lo que sugiere una posible influencia del liderazgo percibido en la eficiencia del proceso de estimación. Sin embargo, se considera que se requiere de más investigaciones que ayuden a comprender completamente este fenómeno.

Con el fin de ampliar la visión que se tiene sobre los factores que afectan tanto la precisión como la efectividad de la estimación del esfuerzo en un proyecto ágil, la Tabla 1 resume los hallazgos más recientes al respecto que se encuentran en la literatura. En esta, se observa que su presencia o no en los proyectos dependerá en gran medida de las acciones realizadas por el humano. Es decir, es importante reconocer que el desempeño del ser humano puede beneficiar o perjudicar la precisión de una estimación. En este sentido, al considerar los principios ágiles que priorizan las interacciones humanas sobre las herramientas, resulta fundamental examinar de cerca aquellos factores humanos que impactan en la estimación del esfuerzo. Para ello, es esencial considerar la experiencia del equipo al realizar las tareas de estimación, sus habilidades de comunicación, su capacidad para gestionar eficientemente el tiempo, su motivación y compromiso, así como su destreza para resolver problemas durante el proceso de estimación. Estos factores humanos desempeñarán un papel crucial en la obtención de estimaciones precisas y realistas, y su adecuada consideración contribuye significativamente al éxito del proyecto (Dantas et al., 2018). Por lo tanto, la precisión en la estimación del esfuerzo ágil está estrechamente ligada a la experiencia de los profesionales en el desarrollo de software (Sudarmaningtyas y Mohamed, 2022). Sin embargo, los factores que influyen en la obtención de una precisión efectiva son diversos y a menudo se vinculan con las características individuales del estimador (e.g., líder del proyecto) cuando, por ejemplo, se depende del consenso entre los expertos.

En los métodos ágiles, como Scrum, la estimación se realiza a menudo a través de la participación humana y se basa, como consecuencia, en la experiencia y los datos históricos de los proyectos anteriores. Sin embargo, estos proyectos generalmente se enfocan en el esfuerzo de desarrollo de cada funcionalidad, sin considerar el contexto específico en el que se llevaron a cabo. No obstante, es crucial reconocer que algunos factores humanos, como la rotación de los integrantes del equipo y la falta de experiencia en determinadas tecnologías o tipos de proyectos, son difíciles de medir y que por ende pueden influir negativamente en la terminación exitosa del proyecto. De acuerdo con Fávero et al. (2018), estos factores aumentan la probabilidad de generar errores en la estimación. Aunado a esto, es importante destacar que, en la práctica actual de la estimación del esfuerzo en el desarrollo ágil de software, predomina la subestimación del esfuerzo. De hecho, la investigación de Conoscenti et al., (2019) argumentó que las estimaciones inexactas en los proyectos ágiles de desarrollo de software a menudo se deben a errores humanos que se relacionan con el equipo, como la falta de experiencia en la tecnología y las prácticas de estimación, la insuficiente participación de las stakeholders, problemas de intercambio de conocimientos y/o la influencia de personalidades dominantes dentro del equipo. Además, las estimaciones de los expertos a menudo son susceptibles a sesgos y errores humanos puesto que se ven influenciadas por factores psicológicos comunes en las estimaciones humanas, como los rasgos de carácter o los intereses personales de los expertos, que pueden introducir sesgos que afectan la precisión de las estimaciones. En este contexto, la falta de una formación adecuada y la ausencia de una metodología estándar para que los equipos puedan estimar su trabajo son las causas más relevantes del fracaso en los proyectos ágiles de desarrollo (Fernández-Diego et al., 2020). Estos factores pueden llevar a suposiciones erróneas y decisiones sesgadas que afecten negativamente la precisión de las estimaciones (Pasuksmit et al., 2024). Por lo tanto, considerando la influencia significativa de los factores humanos en la precisión de las estimaciones, es fundamental buscar y desarrollar técnicas de estimación que mitiguen estos errores humanos.

Tabla 1. Factores identificados en la literatura.

Título del estudio	Referencia	Tipo de factor	
		Relacionados con los proyectos	Relacionados con las personas
<i>“Impact of competence on agile effort estimation in academic setting”</i>	(Fürst et al., 2024)	--	Habilidades sociales, conocimientos sobre el dominio, experiencia en proyectos similares.
<i>“Significant factors in agile software development of effort estimation”</i>	(Sudarmaningtyas y Mohamed, 2022)	Requisitos del proyecto, implementación del diseño.	Experiencia (datos) del proyecto anterior, conocimientos sobre el dominio, habilidad técnica.
<i>“Factors affecting sprint effort estimation”</i>	(Ramessur y Nagowah, 2020)	Seguridad, configuración, cambios en los requisitos, complejidad de los requisitos, volatilidad de los requisitos, calidad de los requisitos.	Dinámica del equipo y compromiso, comunicación, experiencia del personal, experiencia (datos) del proyecto anterior, responsabilidades fuera del proyecto, participación de los <i>stakeholders</i> .
<i>“Software development effort estimation: A systematic mapping study”</i>	(Carbonera et al., 2020)	Metodologías de desarrollo, plataformas de implementación, patrones de interacción.	Conocimientos sobre el dominio, problemas culturales, estados emocionales, género, interés individual y grupal.
<i>“An update on effort estimation in agile software development: A systematic literature review”</i>	(Fernández-Diego et al., 2020)	Complejidad o el carácter innovador del proyecto, la calidad, la asunción de riesgos y la claridad de los requisitos	Niveles de conocimiento y habilidad del grupo de expertos, capacidad de formar equipos, la familiaridad con el equipo/proyecto, el tamaño y la velocidad del equipo, la comunicación, las horas/días de trabajo, la disponibilidad de los desarrolladores, motivación y el liderazgo.
<i>“Effort estimation in agile software development: Case study and improvement framework”</i>	(Tanveer et al., 2017)	Dependencias entre el <i>backlog</i> , complejidad del proyecto.	Habilidad técnica, conocimientos sobre el dominio, experiencia (datos) del proyecto anterior.

2.3.3. Técnicas para realizar la estimación de esfuerzo

Sharma y Vijayvargiya (2021) consideran que los métodos convencionales que se utilizan en las organizaciones desarrolladoras de software a menudo no son suficientes para abordar las necesidades actuales impuestas en el mercado. La sobreestimación y subestimación del esfuerzo pueden tener implicaciones financieras importantes en términos de costos de personal, recursos y presupuesto del proyecto. En este contexto, la precisión en la estimación ha sido, a lo largo del tiempo, un factor determinante en la aceptación o rechazo de un método o técnica de estimación del esfuerzo en los proyectos de software. Sin embargo, aunque la precisión puede ser alta, esto no garantiza el éxito del proyecto (Mahmood et al., 2022). En años recientes, los responsables de la toma de

decisiones en el ámbito del desarrollo de software se han enfrentado a este problema y han buscado un método de estimación que sea aplicable a una organización específica con el fin de que contribuya al logro de los objetivos organizacionales. Como una posible orientación para alcanzar este fin, la investigación realizada por Pasuksmit et al., (2024) establece que la efectividad de un método de estimación debe basarse en dos criterios principales:

- La aceptación necesaria, que implica que la organización desarrolladora de software tenga la capacidad de cumplir con los requisitos establecidos para utilizar un método específico de estimación, tales como la disponibilidad de los datos o la participación de los expertos humanos.
- La aceptación suficiente, que se refiere a la contribución del método en el cumplimiento de los objetivos organizacionales.

Además de la precisión, se espera que los métodos o técnicas de estimación que se empleen respalden una variedad de actividades de la gestión de los proyectos, como el análisis de los riesgos, la comparación de los proyectos o la mejora de los procesos. Desafortunadamente, como ya se ha mencionado, las estimaciones realizadas por las personas son inherentes a la naturaleza humana y están sujetas a una serie de variables y factores cambiantes que pueden influir en su precisión (Kula et al., 2021). Por lo tanto, dada la dinámica y la evolución constante de los proyectos, así como la incertidumbre inherente en el proceso de desarrollo de software, es crucial contar con un método que facilite la obtención de las estimaciones y reduzca al mínimo el margen de error. De acuerdo con Resmi y Anitha (2023), sería recomendable utilizar un enfoque que integre técnicas avanzadas de modelado y análisis de datos con el fin de obtener una estructura más sólida y precisa que fundamente la estimación del esfuerzo, facilitando así la adaptación continua a medida que cambian los requisitos y las condiciones del proyecto. Además, de acuerdo con estos investigadores, de esta manera no solo se estaría mejorando la capacidad de prever y gestionar los recursos necesarios, sino que también se estaría ayudando a la organización a lograr una planificación más efectiva y en la consecución exitosa de los objetivos del proyecto. Sin embargo, existe una amplia variedad de métodos y técnicas para realizar la estimación y resulta difícil que una organización con poco conocimiento sobre el área haga la selección correcta (Rao et al., 2024). El estudio de Jadhav et al., (2022) sobre los métodos y técnicas para la estimación del esfuerzo y los costos en el desarrollo de software (SDECE, por sus siglas en inglés) presentó un análisis detallado sobre los más utilizados durante las últimas cinco décadas, con el objetivo de identificarlos y clasificarlos (véase Figura 2.6).

Dicho estudio revela que técnicas como las redes neuronales artificiales, la lógica difusa y la regresión han ganado popularidad a lo largo del tiempo. A continuación, se destacan las principales fortalezas y debilidades de las técnicas identificadas en dicho estudio:

- La regresión lineal presenta diversas fortalezas, como su fácil comprensión, implementación, interpretación y explicación. Además, esta técnica puede funcionar eficazmente con conjuntos de datos pequeños y no es computacionalmente costosa. Sin embargo, también tiene limitaciones que deben considerarse, como suponer que existe una relación lineal entre las variables, lo que, en consecuencia, la hace inadecuada para datos con relaciones no lineales. Además, la efectividad de la técnica puede verse afectada por problemas de multicolinealidad y es sensible a valores atípicos, lo que puede distorsionar los resultados. En este contexto, el clasificador bayesiano ingenuo (NB, por sus siglas en inglés), también destaca en esta categoría. Este clasificador se desarrolla bajo la suposición de que las variables, dada la variable de clase, son condicionalmente independientes. Gracias a sus ventajas frente a otros clasificadores, el NB ha sido ampliamente utilizado como un método efectivo de clasificación

puesto que destaca por su rendimiento sobresaliente en una variedad de dominios, su eficiencia, su robustez ante datos ruidosos o faltantes, y su relativa facilidad de aplicación (Wu y Gao, 2016).

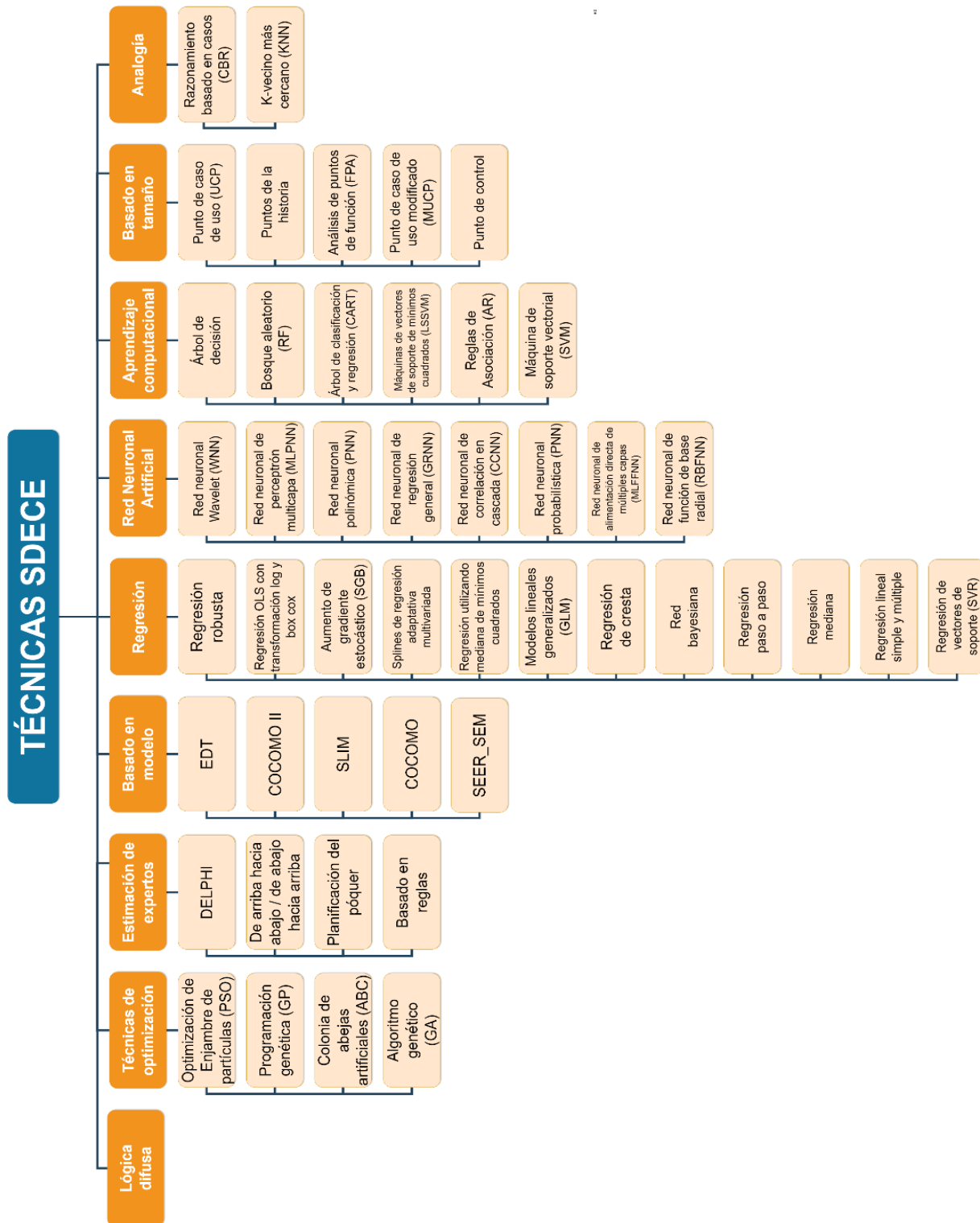


Figura 2.6. Métodos y técnicas para la estimación del esfuerzo y de los costos en un proyecto de desarrollo de software. Nota: Traducida de Jadhav et al., (2022).

- Las técnicas de aprendizaje computacional más utilizadas para estimar el esfuerzo y los costos del desarrollo de software son los modelos basados en árboles y las SVM. Es importante recalcar que también cada técnica tiene sus propias fortalezas y debilidades. Las principales fortalezas de los modelos basados en árboles, como los árboles de decisión⁵ (DT, por sus siglas en inglés), los árboles de clasificación y regresión (CART, por sus siglas en inglés) y los bosques aleatorios (RF, por sus siglas en inglés), son su intuitividad, que facilita la comprensión e interpretación de los resultados, y su capacidad para manejar datos tanto categóricos como numéricos. Sin embargo, estos mismos modelos están propensos al sobreajuste con conjuntos de datos pequeños, no pueden manejar valores faltantes y pueden volverse computacionalmente costosos con conjuntos de datos grandes. En contraste, las SVM son adecuadas para datos de alta dimensionalidad y pueden aprender en presencia de relaciones no lineales en los datos. Pero, por otro lado, son intensivas en memoria y pueden no escalar bien con grandes conjuntos de datos.
- La fortaleza de la estimación de expertos reside en su utilidad cuando no se cuenta con datos cuantificables o empíricos, pero su principal debilidad es que se fundamenta únicamente en el conocimiento y la experiencia de los expertos. Es decir, se corre el riesgo de que la estimación sea una mera opinión que esté sesgada por la experiencia de cada participante, lo que podría derivar en resultados inexactos o erróneos. Un ejemplo de una técnica que se basa en el conocimiento de expertos y que de hecho es bastante utilizada en entornos ágiles de desarrollo, es precisamente la planificación de póker que se ha mencionado recurrentemente en este documento. De acuerdo con Sudarmaningtyas y Mohamed (2020), cada iteración en la que se realice esta estimación inicia con una explicación breve por parte del usuario o propietario del producto sobre una historia de usuario específica. Una vez que dicha historia ha sido discutida exhaustivamente, cada integrante del equipo debe realizar sus estimaciones seleccionando cartas que representan el esfuerzo requerido para desarrollarla basándose únicamente en su experiencia previa. Posteriormente, todos los participantes muestran simultáneamente sus cartas y, cuando todas las estimaciones coinciden en un mismo valor numérico, se establece este valor como la estimación acordada. En caso contrario, será necesario realizar una discusión más detallada para las estimaciones más bajas o altas. Después de la discusión, cada participante realiza nuevas estimaciones de manera individual y cerrada, para mostrar las cartas nuevamente. Este proceso se repite hasta que se alcanza un consenso sobre la estimación de la respectiva historia de usuario.
- Las estimaciones basadas en tamaño y modelos facilitan considerablemente la planificación, el control y la elaboración de presupuestos para los proyectos. Sin embargo, su debilidad radica en que se depende considerablemente de la calibración en base a la experiencia previa. Por lo tanto, la principal complicación de la estimación surge cuando se enfrentan circunstancias sin precedentes.
- Las ANN se fundamentan en una analogía burda con el funcionamiento del cerebro humano, especialmente del sistema nervioso, el cual está compuesto por redes de neuronas biológicas que, si bien tienen capacidades de procesamiento limitadas individualmente, su capacidad cognitiva total se deriva de la conectividad entre ellas. La unidad básica de una ANN es la

⁵ Un árbol de decisión es representado por un algoritmo que asigna datos vectoriales de entrada a una única decisión de salida. Dicha decisión se toma realizando pruebas que comienzan desde la raíz del árbol y siguen una rama hasta llegar a una hoja. Cada rama del árbol proviene de una condición o prueba y, dependiendo de la respuesta, los datos evaluados pueden dirigirse hacia una rama específica (Rodríguez Sánchez et al., 2023).

neurona, que calcula una suma ponderada de sus entradas y aplica una función de activación para generar una señal transmitida a la siguiente neurona. Estas neuronas artificiales se organizan en capas o niveles y están altamente interconectadas mediante conexiones ponderadas por pesos. A través de algoritmos de aprendizaje supervisado o no supervisado, las ANN ajustan su arquitectura y parámetros para minimizar una función de error, lo que indica el grado de ajuste a los datos y la capacidad de generalización de la red (Abdolrasol et al., 2021). En este sentido, las ANN ofrecen diversas fortalezas para la estimación, como su capacidad para aprender relaciones complejas de los datos y su capacidad para hacerlo sin necesidad de realizar ingeniería de características⁶. Sin embargo, también se tienen ciertas debilidades, como la necesidad de contar con una cantidad importante de datos para realizar el entrenamiento, lo que puede ser computacionalmente costoso. Además, es difícil explicar los resultados obtenidos, lo que dificulta su interpretación.

- La estimación basada en la analogía facilita la comprensión del razonamiento en el que se basa el resultado y destaca por su capacidad para manejar valores atípicos. Sin embargo, también existen deficiencias que deben considerarse, como la intensidad computacional que requiere, la sensibilidad a la función de similitud y la necesidad de convertir variables categóricas a tipo numérico. Además, no es posible manejar los valores faltantes y es posible que se dificulte su implementación si no se cuenta con experiencia previa. Dos algoritmos destacan en esta categoría por ser los más empleados para la estimación del esfuerzo en los proyectos de software. Por un lado, el razonamiento basado en casos se ha destacado por su capacidad de utilizar experiencias pasadas para la resolución de problemas (Wu et al., 2018). Es decir, a diferencia de otros métodos que se basan en el conocimiento general del dominio o en asociaciones entre las descripciones y las conclusiones, este algoritmo recurre a situaciones pasadas similares (i.e., casos) para abordar nuevos problemas. Además, se caracteriza por utilizar un aprendizaje incremental, puesto que cada nueva experiencia es almacenada para futuras aplicaciones. Ahora bien, el algoritmo conocido como *K*-vecinos más cercanos (KNN, por sus siglas en inglés) ha figurado por facilitar la estimación mediante el uso de la distancia euclidiana con el fin de calcular las distancias entre diferentes ubicaciones. Así, el algoritmo hace una predicción sobre la clase a la que pertenece un punto al reunir las muestras más cercanas.
- Las técnicas de optimización son ideales para ponderar y seleccionar las características, además de que contribuyen a mejorar la precisión de la estimación cuando se combinan con otras técnicas, ya sean de aprendizaje computacional o no. Por otro lado, sus principales debilidades radican en su carácter no determinista, es decir, estas técnicas no son capaces de predecir o determinar un resultado específico bajo las mismas condiciones iniciales o aplicando las mismas técnicas al mismo conjunto de datos, lo que puede conducir a obtener variaciones en los resultados para cada instancia, además de que son computacionalmente costosas.
- La lógica difusa permite manejar la incertidumbre en los datos que es causada por errores de medición durante la recopilación de los datos y lidiar con la incertidumbre en el modelo mismo. Además, es posible obtener un rendimiento mejorado cuando se combina con modelos de aprendizaje computacional. Sin embargo, su principal debilidad radica precisamente en que esta combinación puede ser computacionalmente intensiva.

⁶ La ingeniería de características comprende la extracción y transformación de variables a partir de datos sin procesar, de modo que sea posible utilizarlas para el entrenamiento y la predicción (Paulson et al., 2022).

Como se observa, la variedad de técnicas es considerable; sin embargo, es preciso recordar al mismo tiempo que la estimación del esfuerzo en el contexto del desarrollo ágil de software presenta todo un desafío, ya que los requisitos están en constante evolución y se desarrollan a medida que el proyecto avanza (Sandeep et al., 2022). En este escenario, actualmente se utilizan diferentes propuestas de técnicas de aprendizaje computacional. Sin embargo, existe una técnica que ha ganado popularidad en los últimos años, esta técnica es el modelo de ensamble, el cual está integrado por varios algoritmos de aprendizaje computacional, que favorecen a la predicción de la estimación del esfuerzo (Carvalho et al., 2020).

En el ámbito específico de la estimación del esfuerzo, se ha introducido la llamada Estimación del Esfuerzo por Ensamble (EEE, por sus siglas en inglés), la cual tiene como propósito combinar más de una técnica, o modelos base, utilizando reglas de combinación para predecir el esfuerzo y clasificando dos tipos de técnicas, las heterogéneas y las homogéneas (Hosni et al., 2016; Ali et al., 2023). La Figura 2.7, muestra el proceso de Estimación del Esfuerzo por Ensamble. La EEE homogénea, combina modelos del mismo tipo, pero utilizando diferentes configuraciones en sus parámetros, un ejemplo son los bosques aleatorios (Hosni et al., 2018). Se clasifican en dos tipos:

- *Bagging*: Consiste en entrenar los modelos del mismo tipo, con diferentes subconjuntos del conjunto de datos, haciendo que cada modelo aprenda patrones distintos y combine las predicciones de la estimación, aplicando una regla de combinación, que puede ser por promedio simple, promedio ponderado y suma ponderada, para obtener un mejor resultado de la estimación.
- *Boosting*: Los modelos del mismo tipo son entrenados en forma secuencial, cada modelo se enfoca en corregir los errores cometidos por el anterior modelo, la estimación final es obtenida mediante la suma ponderada, como regla de combinación (Mishra et al., 2025).

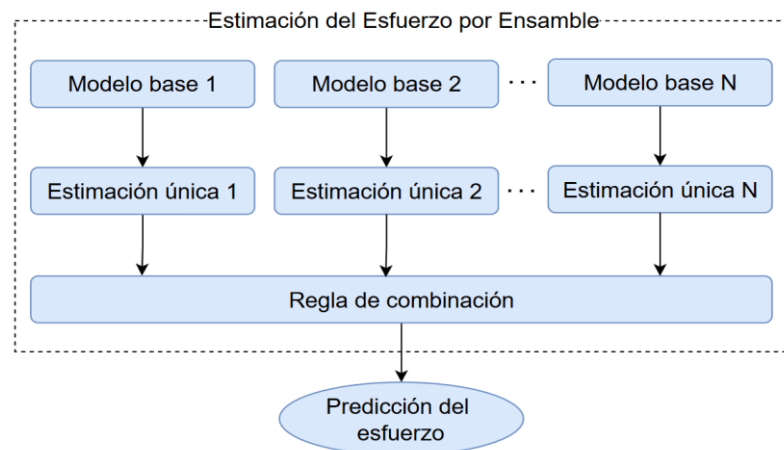


Figura 2.7. Proceso de EEE. Nota: Traducida de Hosni et al., (2016).

El modelo de *bagging* es eficaz con conjuntos de datos de alta varianza, previniendo el sobreajuste mediante el promediado o votación de la predicción, sin enfocarse en corregir los errores de los modelos anteriores como el *boosting* (Kumar et al., 2025). Por otro lado, EEE heterogéneo, combina al menos dos técnicas diferentes como modelos base (e.g. utilizar SVR, DT, RF y RL en conjunto) (Hosni, 2023), el enfoque más utilizado es:

- *Stacking*: Consiste en entrenar varios modelos para integrar sus predicciones de manera conjunta. Para ello se utilizan varios modelos que individualmente son distintos, como

modelos base, sus predicciones son utilizadas como entrada para un modelo de segundo nivel, denominado meta-modelo, el cual, combina estas predicciones y produce una estimación final (Mishra et al., 2025).

Estas técnicas son utilizadas en diferentes ámbitos del software, Sakhravi et al. (2022), por ejemplo, desarrollaron un modelo de stacking para estimar el esfuerzo en proyectos Scrum, mostrando que la combinación de regresores logró reducir de manera significativa el error respecto a modelos aislados. De manera similar, Rodríguez Sánchez et al., (2023) encontraron cuales técnicas basadas en árboles de decisión, al integrarse en esquemas de ensamble como *Random Forest* o *AdaBoost*, alcanzaron un mejor desempeño al estimar tiempos y costos a partir de puntos de historia, reforzando la idea de que la agregación de predictores incrementa la precisión. En la misma línea, Alsaadi y Saeedi (2022) propusieron un modelo de ensamble orientado a equipos ágiles novatos, evidenciando que la combinación de estimadores logra compensar la falta de experiencia y reduce la variabilidad en la predicción del esfuerzo de las historias de usuario. Incluso enfoques más complejos, como la integración de sistemas neuro-difusos con algoritmos metaheurísticos (caso del ANFIS-EEBAT aplicado a proyectos Scrum) han mostrado ser más efectivos que los modelos individuales al proporcionar mejores métricas de precisión y, al mismo tiempo, ofrecen un grado mayor de interpretabilidad (Arora et al., 2022). Estos resultados coinciden con las conclusiones de revisiones sistemáticas (Idri et al., 2016; Carvalho et al., 2020), las cuales señalan que los ensambles constituyen una tendencia consolidada en la estimación de esfuerzo, destacando su capacidad de reducir errores, manejar datos heterogéneos y adaptarse mejor a la incertidumbre inherente al desarrollo ágil. En conjunto, la literatura muestra que los modelos de ensamble representan un avance sustancial respecto a los métodos individuales. Al integrar múltiples perspectivas de predicción, logran resultados más robustos y confiables, lo que los convierte en una alternativa especialmente adecuada para proyectos ágiles y en particular para Scrum, donde la incertidumbre y el cambio constante en los requisitos hacen indispensable contar con técnicas de estimación más precisas y adaptativas. Considerando la clasificación de las técnicas de estimación, presentadas en la Figura 2.6, en la siguiente sección se ahondará sobre algunos aspectos específicos de éstas.

2.3.3.1. Técnicas de sobremuestreo de minorías sintéticas para regresión

Un problema que afrontan los investigadores en la estimación del esfuerzo utilizando métodos de aprendizaje computacional es el desbalance, esto es, conjuntos de datos con presencia de valores extremos donde la variable objetivo cuenta con valores continuos, por lo que se conoce como regresión desbalanceada (Jawa y Meena, 2022). Para balancear los conjuntos de datos se pueden utilizar las técnicas de sobremuestreo de datos. Algunos de estos métodos se describen a continuación.

SMOTER (*Synthetic Minority Over-sampling Technique for Regression*) es una variante del popular algoritmo SMOTE adaptado para problemas de regresión. La principal diferencia de SMOTER con SMOTE convencional es que, en lugar de generar ejemplos sintéticos en el espacio de clases, se enfoca en el espacio continuo de la variable dependiente (la variable objetivo). SMOTER crea instancias sintéticas a partir de la interpolación entre puntos cercanos en el espacio de características, pero los valores de la variable objetivo se generan de acuerdo con una media ponderada entre las instancias seleccionadas y sus respectivas etiquetas, sin ningún tipo de “decisión binaria”. Esta técnica está dirigida principalmente a corregir el problema de subrepresentación de valores en ciertas regiones del espacio de la variable objetivo (Chawla et al., 2002). KNNSMOTE es una variante de SMOTE que utiliza el algoritmo de KNN para mejorar la interpolación de los puntos sintéticos. En lugar de generar los nuevos puntos sintéticos interpolando entre instancias cercanas de manera

aleatoria, KNNSMOTE utiliza la técnica de KNN para seleccionar los vecinos más relevantes en el espacio de características. La idea es que los puntos generados no solo se basen en la proximidad, sino también en la similitud de los vecinos en términos de las variables de entrada. Al aplicar KNN, KNNSMOTE mejora la coherencia de los ejemplos sintéticos generados en relación con los datos existentes y con el valor de la variable objetivo (Chawla et al., 2002).

SMOBN (*Synthetic Minority Over-sampling Technique for Regression with Gaussian Noise*) es una extensión de SMOTE diseñada específicamente para problemas de regresión donde la variable objetivo está desbalanceada. En lugar de simplemente generar instancias sintéticas mediante interpolación, SMOBN introduce ruido gaussiano a los valores generados para las variables dependientes. El ruido Gaussiano agrega una perturbación controlada a los valores generados, imitando la variabilidad natural que podría ocurrir en escenarios del mundo real. Esto permite que el modelo de regresión generalice mejor y reduzca el riesgo de sobreajuste, ya que los ejemplos sintéticos no son exactamente iguales a los existentes, sino que tienen una dispersión más natural (Branco et al., 2017). SMOTEBoost es una combinación de la técnica de sobremuestreo SMOTE y el algoritmo de *boosting* AdaBoost. El proceso comienza con la generación de instancias sintéticas mediante SMOTE. Luego, el algoritmo de *boosting* AdaBoost se aplica para ajustar un modelo de regresión, enfocado especialmente en aquellos puntos sintéticos que son más difíciles de predecir o que se encuentran en áreas del espacio de características menos representadas. El objetivo principal es mejorar la precisión del modelo al priorizar los puntos sintéticos difíciles de predecir durante el proceso de entrenamiento.

WSSMOTER es una extensión de SMOTER que introduce un mecanismo de ponderación de las instancias generadas. La idea principal detrás de WSSMOTER es que no todas las instancias sintéticas tienen la misma importancia. Por lo tanto, en lugar de generar instancias de manera uniforme, WSSMOTER pondera las instancias en función de su relevancia o de los errores de predicción observados en el modelo de regresión. Esto significa que se prioriza la generación de ejemplos sintéticos que sean más relevantes o informativos para la tarea de predicción, en lugar de simplemente generar puntos sintéticos de forma aleatoria.

2.3.3.2. Métricas de evaluación de los métodos de estimación

La evaluación de los resultados de rendimiento de las técnicas empleadas es necesaria para determinar su eficacia y utilidad en el contexto específico de la estimación. Para este fin se emplean criterios de evaluación que permiten analizar y valorar los resultados obtenidos. De acuerdo con Rastogi et al. (2014), estos criterios proporcionan una medida objetiva para determinar si la técnica ha cumplido con sus objetivos y ha generado resultados satisfactorios. Por lo tanto, evaluar el rendimiento de una técnica permite identificar áreas de mejora y optimización, así como validar su aplicabilidad en diferentes situaciones y escenarios (Piñeros et al., 2023). En este sentido, la Tabla 2 presenta un resumen de las métricas de evaluación que son utilizadas en la actualidad en los diferentes métodos de estimación de software. Estas métricas se utilizan para evaluar la precisión y efectividad, por lo tanto, proporcionan una visión clara sobre la exactitud y consistencia de cada uno.

La MRE se define como la relación entre la diferencia absoluta del esfuerzo real y el esfuerzo estimado y es esencial para evaluar la precisión y eficacia de los modelos de estimación del esfuerzo (Mahmood et al., 2022), como se define en la Ecuación (1).

$$MRE_i = \frac{|EsfuerzoActual_i - EsfuerzoEstimado_i|}{EsfuerzoActual_i} \quad (1)$$

Tabla 2. Métricas de evaluación utilizadas para evaluar los métodos de estimación.

Título del estudio	Referencia	Métricas
<i>“Effort estimation in agile software development: A systematic map study”</i>	(Rodríguez et al., 2023)	MMRE, Evaluación de la predicción (PRED), MAE y MRE.
<i>“Evolution of software development effort and cost estimation techniques: Five decades study using automated text mining approach”</i>	(Jadhav et al., 2022)	MRE, MMRE, y PRED.
<i>“Comparative analysis of machine learning and deep learning algorithms for software effort estimation”</i>	(Varshini et al., 2021b)	MAE, Error cuadrático medio (MSE, por sus siglas en inglés), RMSE y Coeficiente de determinación (R^2).
<i>“A systematic mapping study on software effort estimation”</i>	(Al Kharusi, 2020)	MMRE, PRED, MRE, Mediana del error relativo medio (MdMRE, por sus siglas en inglés) y RMSE.
<i>“An update on effort estimation in agile software development: A systematic literature review”</i>	(Fernández-Diego et al., 2020)	MRE, PRED y R^2 .

La MMRE representa el promedio de los valores del MRE sobre todos los proyectos, como se define en la Ecuación (2). Por lo general, se considera aceptable un MMRE del 25%, lo que indicaría que, en promedio, la precisión de los modelos de estimación establecidos sería inferior al 25% (Abdelali et al., 2018; Mustapha y Abdelwahed, 2019).

$$MMRE = \frac{1}{N} \sum_{i=1}^N MRE_i \quad (2)$$

donde N es el número total de estimaciones y MRE_i es la Magnitud del Error Relativo de la i -ésima estimación.

Otro criterio ampliamente utilizado es el PRED(l), que representa el porcentaje de MRE que es menor o igual al valor de $l=1$ entre todos los proyectos. Esta medida muestra la proporción de proyectos para un nivel específico de precisión. La definición de PRED(l) se expresa en la Ecuación (3), donde N , es el número total de observaciones y k es el número de observaciones cuyo MRE es menor o igual que l (Mustapha y Abdelwahed, 2019). Un valor común para l es 0.25, pero puede variar entre los valores 0.25, 0.50, 0.75 o 1.0 (Arora et al., 2022). El PRED (0.25) representa el porcentaje de proyectos cuyo MRE es menor o igual al 25%.

$$PRED(l) = \frac{k}{N} \quad (3)$$

La investigación de Sembhoo y Gobin-Rahimbux (2023) argumenta que el MAE se utiliza como una medida para evaluar el rendimiento de un modelo de estimación. La Ecuación (4) muestra que el MAE se calcula como la diferencia absoluta entre el valor real y el valor estimado para cada punto de

historia en el conjunto de prueba, y luego se promedia sobre todos los puntos de historia. Cuanto menor sea el valor de MAE, más preciso será el modelo en sus estimaciones.

$$MAE = \frac{1}{N} \sum_{i=1}^N |EsfuerzoActual_i - EsfuerzoEstimado_i| \quad (4)$$

donde N es el número total de proyectos en el conjunto de prueba, $EsfuerzoActual_i$ representa el valor real del punto de historia para el i -ésimo problema, y $EsfuerzoEstimado_i$ es el valor estimado del punto de historia para el i -ésimo proyecto.

Finalmente, el coeficiente denominado R^2 es una medida estadística que indica la proporción de varianza en la variable dependiente, que se puede predecir a partir de la variable independiente. En este sentido, el cálculo se realiza dividiendo la suma residual de cuadrados (RSS, por sus siglas en inglés) por la suma total de cuadrados (TSS, por sus siglas en inglés), y luego restando este resultado de 1, como lo muestra la Ecuación (5).

$$R^2 = 1 - \frac{RSS}{TSS} \quad (5)$$

La RSS representa el error cuadrático promedio entre los valores originales y las predicciones del modelo, mientras que la TSS representa el error cuadrático entre los valores originales y el promedio de todos los valores. El valor de R^2 más cercano a 1 indicaría un mejor ajuste del modelo, mientras que un valor negativo revelaría que no existe correlación entre los datos y el modelo (Varshini et al., 2021a).

2.3.3.3. Técnicas de estimación para proyectos ágiles de software

De acuerdo con la literatura, dos técnicas de estimación destacan en los proyectos ágiles de desarrollo de software. Como ya se mencionó, la primera de las técnicas de estimación más utilizadas en Scrum es la planificación de póker, puesto que facilita la estimación colaborativa del esfuerzo requerido para desarrollar funcionalidades descritas en los puntos de historia. De acuerdo con Canedo et al. (2018), cada integrante del equipo recibe una baraja de cartas con valores predefinidos, como 0, 1, 2, 3, 5, 8, 13, 20, 40, 100. Popularmente, a esos valores se les llama unidades de esfuerzo y son relativos entre diferentes tareas contadas en los puntos de historia. Durante una ronda de la planificación de póker, después de una discusión detallada sobre una característica o una historia de usuario, cada participante deberá seleccionar una tarjeta con un número que cree representará su estimación. Si todos los integrantes coinciden, se elige ese valor. De lo contrario, los expertos con las cifras más altas y bajas explicarán sus razones, y se realizará otra ronda de estimación hasta alcanzar un consenso. Esta técnica no solo promueve una comprensión compartida de las tareas, sino que también ayuda a predecir la velocidad del equipo y la duración del proyecto al dividir el número total de puntos de historia por la velocidad y multiplicar el número de iteraciones por la longitud de cada iteración.

Por otro lado, la segunda técnica de estimación dominante en varios campos, no solamente en el desarrollo ágil de software, es el juicio de expertos. En este caso, una persona experta en la tarea realiza estimaciones a través de un proceso de razonamiento intuitivo. Además, los 12 principios establecidos por la técnica pueden proporcionar pautas útiles para el proceso de estimación. Estos principios abordan la evaluación de la precisión en las estimaciones, así como también el *feedback* y

entrenamiento necesario a los estimadores. Por último, pero no menos importante, se debe prestar especial atención a la utilización de datos documentados en las tareas anteriores, la conexión con expertos que tengan conocimiento relevante, y la creación de procesos para evaluar y gestionar la incertidumbre en las estimaciones. De esta manera se contribuye a la mejora continua del proceso de estimación, puesto que se implementan prácticas y estrategias adecuadas (El Beggar, 2024).

Las técnicas anteriores, aunque son fundamentales en Scrum, a menudo pueden afectar las estimaciones debido a su dependencia de la intuición y el juicio subjetivo. Esto puede introducir sesgos y variabilidad, ya que las personas pueden tener opiniones diferentes y no siempre ser consistentes. De acuerdo con Pospieszny et al. (2018), en las últimas dos décadas, se ha realizado una cantidad considerable de investigación en técnicas de minería de datos para la estimación del software, con un enfoque especial en los algoritmos predictivos de aprendizaje computacional. Una de las ventajas de estos algoritmos es que se encuentran entre los más efectivos para tratar adecuadamente la incertidumbre y suministrar estimaciones precisas del esfuerzo y duración en etapas tempranas del ciclo de vida del proyecto. Otra de las ventajas es su funcionamiento automatizado de predicción, que se deriva de los datos históricos; por ende, las influencias psicológicas o políticas, y los posibles sesgos de la condición humana son mucho menores. La Tabla 3 presenta las técnicas del aprendizaje computacional más utilizadas de acuerdo con la literatura más relevante.

Tabla 3. Algoritmos del aprendizaje computacional más utilizados para la estimación de esfuerzo.

Título del estudio	Referencia	Técnicas
<i>“Effort estimation in agile software development: A systematic map study”</i>	(Rodríguez et al., 2023)	NB, KNN, Memoria a largo y corto plazo (LSTM, por sus siglas en inglés), DT.
<i>“Comparative analysis of machine learning and deep learning algorithms for software effort estimation”</i>	(Varshini et al., 2021b)	<i>Deepnet</i> , <i>Neuralnet</i> , SVM, RF.
<i>“An update on effort estimation in agile software development: A systematic literature review”</i>	(Fernández-Diego et al., 2020)	SVM, KNN, NB, DT, RF.
<i>“A systematic literature review of software effort prediction using machine learning methods”</i>	(Ali y Gravino, 2019)	ANN y SVM.
<i>“Systematic literature review of ensemble effort estimation”</i>	(Idri et al., 2016)	DT, Programación genética, Razonamiento basado en casos (CBR, por sus siglas en inglés), SVR, ANN.

Como se podrá observar, los DTs representan a la técnica más utilizada en la literatura actual, lo que sugiere su alta efectividad y popularidad en la estimación del esfuerzo para los proyectos de software.

2.3.4. Problemas y retos actuales en la estimación de esfuerzo

A pesar de los esfuerzos de las investigaciones realizadas y los recursos invertidos en mejorar la precisión de los modelos, la incertidumbre inherente a los proyectos de desarrollo de software, los factores complejos de interacción, la complejidad del software, la presión por la estandarización y la

falta de datos precisos dificultan la obtención de estimaciones realistas de esfuerzo. De acuerdo con Pandey et al. (2020), los modelos actuales de estimación no garantizan una precisión constante dentro del 75% del tiempo estimado, pero siguen siendo útiles para apoyar las actividades previstas en la medida de lo posible. Asimismo, la investigación de Almutlaq y Jawawi (2020) destacó que la estimación del esfuerzo en los proyectos de software enfrenta diversos problemas que afectan la precisión de los resultados. La disponibilidad de datos, por ejemplo, es crucial para verificar la validez de los resultados y para que los modelos puedan ajustarse adecuadamente. Sin embargo, la inestabilidad del entorno de software complica el proceso de estimación, ya que los sistemas están en constante cambio y crecimiento, lo que dificulta la predicción de problemas futuros y la corrección de los resultados. Otro problema es la sensibilidad de los datos, ya que diferentes empresas utilizan diferentes tipos de datos y esto debe considerarse en los modelos de estimación para manejar estos cambios de manera efectiva. Sin embargo, estos desafíos no suelen ser abordados directamente en los modelos de estimación actuales.

Aunado a lo anterior, la falta o el difícil acceso a datos históricos en los proyectos de software representa uno de los principales obstáculos para las organizaciones al realizar estimaciones para nuevos proyectos (Moradbeiky et al., 2022). La importancia de los datos históricos como base de la estimación radica en proporcionar, tanto a los líderes de proyectos como al personal administrativo, referencias concretas y evidencia tangible de estimaciones de proyectos anteriores. Un ejemplo de esto es la base de conocimiento proporcionada por el Grupo de Estándares Internacionales de Evaluación Comparativa de Software (ISBSG, por sus siglas en inglés), una organización global e independiente que proporciona datos y análisis para la industria de las Tecnologías de la Información. El ISBSG cuenta actualmente con dos repositorios de datos, que son recursos valiosos para mejorar la precisión de las estimaciones y apoyar la toma de decisiones en los proyectos de software (<https://www.isbsg.org/>).

Finalmente, con el objetivo de analizar la forma en que diferentes investigadores han abordado la aplicación del aprendizaje computacional en la obtención de estimaciones de esfuerzo en proyectos de desarrollo de software, se proporciona a continuación un análisis sobre tres investigaciones relevantes a modo de estado del arte.

2.4. Estado del arte

En los últimos años, diversas técnicas han ganado popularidad debido a su capacidad para manejar las complejidades inherentes a los proyectos de desarrollo de software. Por lo tanto, el presente estado del arte se centró en buscar, seleccionar, leer y analizar investigaciones que representaran innovaciones y mejoras significativas en la precisión y eficiencia de las estimaciones del esfuerzo a través de la aplicación de técnicas del aprendizaje computacional. Este análisis brinda una descripción detallada de los avances actuales y destaca tanto los desafíos como las oportunidades en la aplicación del aprendizaje computacional en la estimación del esfuerzo en proyectos de software.

2.4.1. Aplicación de técnicas de cómputo suave para la modernización de la estimación del esfuerzo en los proyectos de software

2.4.1.1. Objetivo

La investigación de Sharma y Vijayvargiya (2021) evaluó y comparó cinco técnicas de estimación del esfuerzo en proyectos de software con el fin de identificar aquella que tuviera el mejor rendimiento. De esta manera se buscó validar la precisión del modelo de estimación del esfuerzo

mediante la comparación con datos reales, calculando y documentando métricas de evaluación relevantes.

2.4.1.2. Descripción

El conjunto de datos “China” utilizado contiene 499 instancias de proyectos, de las cuales 399 fueron utilizadas para el entrenamiento (i.e., 80% de las instancias) y el resto para pruebas (i.e., 20% de las instancias). Además, el conjunto contenía 19 atributos o características, de las cuales se seleccionaron nueve variables independientes y una dependiente. Las técnicas incluidas en los estudios fueron las siguientes: LR, SVR, DT, RF y *bagging*. A continuación, se proporciona información más detallada sobre éstas:

- La regresión lineal (LR, por sus siglas en inglés) es una técnica utilizada para modelar una variable dependiente (Y) basada en una o más variables independientes (X). La técnica establece una relación entre las variables, encontrando una línea recta que minimiza la suma de los cuadrados de las distancias perpendiculares de los puntos a la línea. Así, se trata de una técnica fundamental para determinar la interdependencia entre las variables. La regresión logística, aunque está relacionada con la regresión lineal, emplea tanto el aprendizaje computacional como la estadística y es especialmente útil para clasificar datos en categorías binarias, como sí/no o verdadero/falso.
- Las SVM representan a uno de los algoritmos de aprendizaje computacional más conocidos y utilizados, especialmente en problemas de clasificación. Similar a la regresión logística, el propósito principal de las SVM es encontrar el hiperplano que maximiza el margen en un espacio N -dimensional, donde ‘ N ’ es el número de características para organizar los puntos de datos en distintas categorías. El hiperplano separa los datos en diferentes clases.
- Los DTs se utilizan tanto para la clasificación como para la regresión y son componentes esenciales de los RFs. Estos árboles analizan y pronostican modelos a través de la evaluación de la importancia de los atributos y el manejo de datos categóricos. La arquitectura de un DT incluye nodos (i.e., raíz, nodos internos y hojas) y arcos, para organizar los conjuntos de datos de manera jerárquica. En cada nodo interno se aplica una técnica de medición de impurezas para determinar la mejor división posible, optimizando así la estructura del árbol para mejorar la precisión del modelo.
- Los RFs están representados por un algoritmo de aprendizaje supervisado que es utilizado tanto para la clasificación como para la regresión. Un RF, además de ser adaptable y fácil de usar, está compuesto por muchos DTs, por lo que cuantos más árboles contiene, más robusto es el bosque. En el aprendizaje por conjuntos, los RFs crean numerosos árboles durante el entrenamiento y luego determinan la clase final mediante el modo de las clases individuales o el promedio de las predicciones de los árboles.
- Por último, la agregación de *Bootstrap* (*bagging*, por sus siglas en inglés) es una técnica de aprendizaje computacional que ayuda a reducir la varianza de los algoritmos con el fin de evitar el sobreajuste. Esta técnica implica el muestreo repetido con reemplazo de un conjunto de datos, lo que genera muestras de entrenamiento únicas.

2.4.1.3. Resultados

Después de entrenar y probar con los conjuntos de datos, se aplicaron varias métricas de evaluación para analizar la precisión de las técnicas empleadas, entre éstas destacaron la MMRE y la

Predicción al 25% (PRED (25)), que son ampliamente reconocidas en la evaluación de modelos. Aunado a esto, la Tabla 4 resume los resultados obtenidos al recoger cada una de las métricas de evaluación analizadas. En este sentido, los resultados obtenidos muestran que el RF y la regresión logística obtuvieron los mejores valores con MMRE del 15.37% y 15.78%, MAE de 0.98% y 1.24% y para PRED (25) con 92% y 84% respectivamente, por lo que se concluyó que son significativamente más efectivas en comparación con las demás técnicas de estimación.

Tabla 4. Resultados obtenidos con la evaluación de cada técnica. Nota: Traducido de Sharma y Vijayvargiya (2021).

Métricas de evaluación	Técnicas					
	SVR	DT	Regresión lineal	<i>Bagging</i>	Regresión logística	RF
MMRE	60.22	39	49.02	57.97	15.37	15.78
MAE	5.70	0.0	3.85	4.04	1.24	0.98
RMSE	7.91	0.01	7.47	8.21	3.55	1.82
MER	54.44	0.37	72.14	48.37	14.3	13.82
MdMRE	38.94	0.0	29.6	42.47	8.85	9.13
R^2	0.87	1.0	0.82	1.0	0.98	1.0
PRED (25%)	43.0	100	46	30	92	84

2.4.2. Estimación de costo y esfuerzo en el desarrollo ágil de software utilizando las técnicas de árboles de decisión y puntos de historia

2.4.2.1. Objetivo

Por otro lado, la investigación de Rodríguez Sánchez et al., (2023) se enfocó en la implementación de tres técnicas de estimación de software (i.e., DTs, RFs y *AdaBoost*), con el fin de determinar cuál era la que tenía mayor precisión en las predicciones. Al aplicar estas técnicas al conjunto de datos proporcionado, se utilizó el error relativo como medida de comparación para evaluar su rendimiento. Por lo tanto, el propósito principal del estudio fue demostrar que la combinación de estas técnicas puede proporcionar estimaciones más precisas.

2.4.2.2. Descripción

El conjunto de datos consistió en 21 proyectos ágiles realizados por seis organizaciones desarrolladoras de software de Pakistán. Esta información, originalmente presentada en Ziauddin et al. (2012), está organizada en una tabla donde cada fila representa un proyecto y cada columna muestra una característica. Es verdad que la extracción y selección de características es crucial en el preprocesamiento de los datos, pero, debido a que el conjunto de datos era pequeño, se decidió expandir el conjunto de datos generando 42 proyectos sintéticos que fueron destinados para entrenamiento, mientras que los 21 proyectos originales fueron utilizados para pruebas, obteniendo así un total de 63 proyectos. Además, debido a que existen datos categóricos dentro de los valores de algunas características, se utilizó una convención específica de “pequeño”, que se convirtió en 2, “mediano” en 1 y “grande” en 0. Cada algoritmo fue configurado con hiperparámetros que ofrecían la mayor precisión, y fueron entrenados por separado mediante una validación cruzada repetida de 10 veces. Los investigadores también argumentan que un DT puede verse como un algoritmo que asigna datos de entrada a un único valor o decisión de salida. La estructura del árbol se inicia en la raíz y

sigue las ramas hasta llegar a una hoja, donde se toma la decisión final. Cada rama surge de una condición o prueba, y los datos se dirigen a diferentes ramas según el resultado de esa prueba. Por lo tanto, un DT está compuesto por nodos internos que evalúan condiciones y hojas que representan los valores de salida (véase la Figura 2.8).

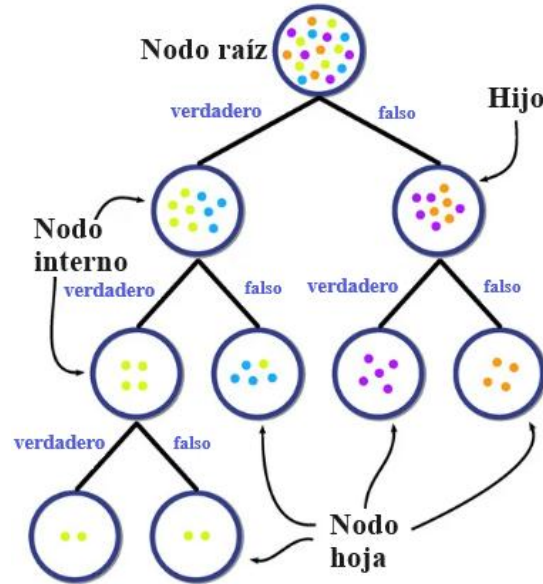


Figura 2.8. Estructura de un árbol de decisión. Nota: Traducida de Rodríguez Sánchez et al., (2023).

La Figura 2.8 muestra que el modelo que representa al RF emplea un tipo de embolsado de árboles de decisión, donde se selecciona una muestra aleatoria de atributos para determinar cuáles producen los mejores resultados. Los parámetros para configurar el método incluyen una profundidad máxima de seis y un mínimo de dos muestras por división. Además, en el estudio se aplicó una validación cruzada de diez veces, con dos repeticiones. El proceso también implica evaluar el esfuerzo de entrada y el tamaño con etiquetas que indican el esfuerzo y el costo de tiempo de salida.

Esta estructura jerárquica permite dividir los datos de forma recursiva para identificar regiones locales en pocos pasos. Para configurar el método, se utilizó una profundidad máxima de seis con un mínimo de dos muestras por división. Además, se aplicó una validación cruzada de diez veces, repetida dos veces, para evaluar el rendimiento del modelo. El aprendizaje por conjuntos, por otro lado, es una técnica valiosa en el aprendizaje computacional puesto que combina varios modelos individuales para mejorar la precisión de las predicciones y reducir el sesgo. Aunque requiere más cálculo al utilizar n modelos base, disminuye la varianza, ya que el promedio o voto mayoritario del conjunto se aproxima más al valor objetivo. Una de las técnicas más populares dentro de este enfoque es el RF (véase la Figura 2.9).

La Figura 2.9 muestra que el modelo que representa al RF emplea un tipo de embolsado de árboles de decisión, donde se selecciona una muestra aleatoria de atributos para determinar cuáles producen los mejores resultados. Los parámetros para configurar el método incluyen una profundidad máxima de seis y un mínimo de dos muestras por división. Además, en el estudio se aplicó una validación cruzada de diez veces, con dos repeticiones. El proceso también implica evaluar el esfuerzo de entrada y el tamaño con etiquetas que indican el esfuerzo y el costo de tiempo de salida.

Finalmente, *AdaBoost* es un método que mejora un algoritmo de aprendizaje débil combinando diferentes modelos de éste y aumentando la precisión (véase la Figura 2.10). El algoritmo de AdaBoost asigna pesos a los puntos de datos, donde los errores obtienen pesos más altos para que el siguiente

modelo base débil les preste más atención, reduciendo el error gradualmente. La configuración de AdaBoost es similar a las demás técnicas con una profundidad máxima de seis, un mínimo de dos muestras por división, y una validación cruzada de diez veces repetida dos veces y, además, se evaluó el esfuerzo real de entrada mediante la obtención del esfuerzo y el costo estimados de salida.

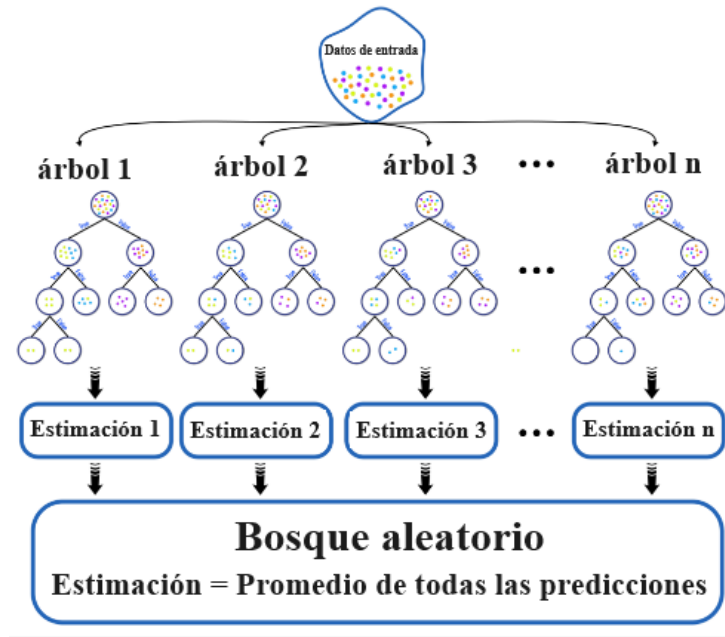


Figura 2.9. Estructura de un bosque aleatorio. Nota: Traducida de Rodríguez Sánchez et al., (2023).

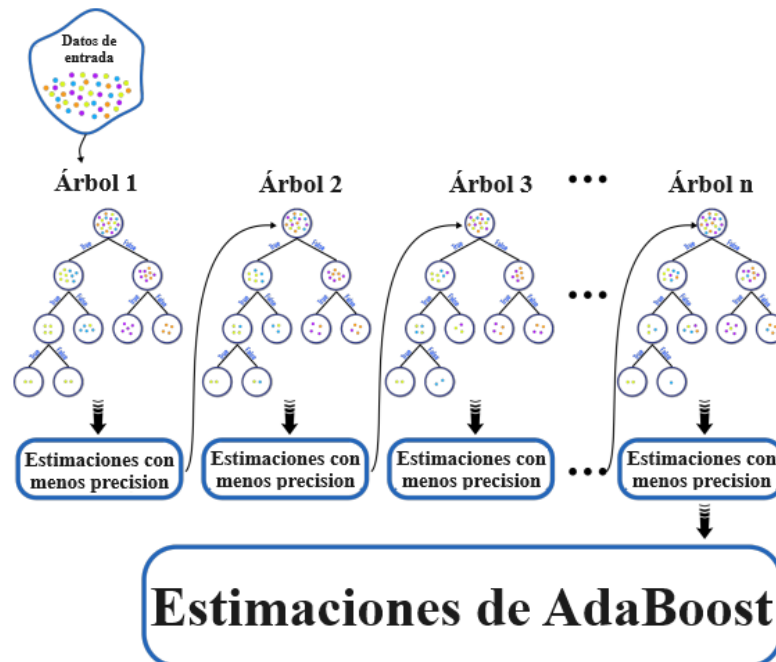


Figura 2.10. Estructura de AdaBoost. Nota: Traducida de Rodríguez Sánchez et al., (2023).

2.4.2.3. Resultados

Las tres técnicas antes mencionadas fueron evaluadas con diversas métricas, como MAE, MdMRE, MMRE y PRED. Los resultados revelaron que, aunque la técnica de RF mostró la menor precisión, con un valor de 96.92% y un R^2 de 0.9869, AdaBoost demostró un mejor rendimiento en la estimación del tiempo, mientras que los DTs destacaron en la estimación del costo total. Un hallazgo importante de la investigación fue que los DT tendieron a sobre ajustar los datos, destacando así la importancia de ajustar adecuadamente los parámetros del modelo para evitar esto. En general, las predicciones realizadas por las tres técnicas se acercaron a los valores reales, sugiriendo que los modelos de aprendizaje computacional son capaces de aproximarse efectivamente a la función que representan los proyectos de software.

2.4.3. Una red neuronal LSTM optimizada para la estimación precisa del esfuerzo en el desarrollo de software

2.4.3.1. Objetivo

Por último, la investigación de Iordan (2024) propuso un método optimizado de red neuronal de LSTM que utiliza el PSO para ajustar tres hiperparámetros con el fin de obtener la estimación más precisa del esfuerzo requerido para desarrollar un producto de software. Aunado a esto, se proporcionaron los resultados de comparar dicho método contra seis técnicas de aprendizaje computacional: KNN, DT, RF, Árbol Potenciado por Gradiente (GBT, por sus siglas en inglés), MLP y redes neuronales de LSTM.

2.4.3.2. Descripción

Para este estudio se utilizaron cinco conjuntos de datos: Albrecht, China, COCOMO81, Desharnais y Kemerer, los cuales contienen muestras relacionadas con la estimación de esfuerzo en proyectos de software, implementados bajo una metodología tradicional de desarrollo de software. Como se puede observar en la Tabla 5, de cada conjunto de datos se seleccionaron solamente seis características, las cuales representan los parámetros que utilizarían las técnicas para ajustar su aprendizaje con el fin de obtener mejores resultados. Los conjuntos de datos se dividieron en un 80% para el entrenamiento y el 20% restante para pruebas. A continuación, se proporcionan detalles de la configuración. La técnica de KNN fue implementada utilizando seis variables de entrada y una de salida (i.e., el esfuerzo estimado), considerando las siguientes características: medida de Minkowski para calcular la distancia, tamaño de hoja de 30 y pesos uniformes. Para su implementación se utilizó la función *KNeighborsRegressor* y la biblioteca *SKlearn* 1.0.1. Durante el proceso de ajuste de los parámetros se evaluaron dos parámetros principales para determinar el rendimiento de la técnica KNN: k , que es el número de vecinos considerados para predecir nuevas instancias (se probaron ocho valores de k , que varían entre 3 y 10 para predecir el esfuerzo) y p , que es el parámetro de potencia para la distancia de Minkowski. La distancia de Minkowski fue utilizada para seleccionar los vecinos más relevantes al comparar características con una nueva instancia. En este sentido, se evaluaron tres valores de p , entre 1 y 3. Con $p = 1$, la distancia de Minkowski equivale a la distancia de Manhattan, y con $p = 2$, equivale a la distancia Euclidiana. Con los valores seleccionados para los parámetros (i.e., ocho valores para k y tres valores para p), se entrenaron y evaluaron 24 variantes de KNN.

Para la técnica de DT se utilizaron también seis características y una salida que es el esfuerzo estimado. Las características configuradas fueron las siguientes: un mínimo de dos muestras requeridas para dividir un nodo interno y una estrategia de división basada en la mejor división del

nodo. Este método se implementó mediante la función *DecisionTreeRegressor* y la biblioteca *SKlearn* 1.0.1.

Tabla 5. Conjuntos de datos utilizado para la evaluación de las técnicas. Nota: Traducido de Iordan (2024)

Conjunto de datos	Proyectos	Características	Características seleccionadas	Esfuerzo
Albrecht	24	8	6 (Número de funciones de entrada, Número de funciones de salida, Recuento de procesamiento de archivos, Puntos de función sin procesar, Puntos de función ajustados, y Recuento de funciones de consulta)	Meses-persona
China	499	19	6 (Puntos de función de entrada, Puntos de función de salida externa, Puntos de función de archivos lógicos internos, Puntos de función de funciones agregadas, Tipo de equipo, y Puntos de función ajustados)	Horas-persona
COCOMO81	63	17	6 (Tamaño de la base de datos, Restricción de tiempo, Complejidad del producto, Restricción de almacenamiento, Uso de herramientas de software, y Líneas de código)	Meses-persona
Desharnais	81	12	6 (Experiencia del equipo, Experiencia del gerente, Duración del proyecto, Numero de entidades, Lenguaje de programación, y Factor ajustado)	Horas-persona
Keremer	15	8	6 (Lenguaje de programación, Puntos de función no ajustados, Duración del proyecto, Puntos de función ajustados, Recursos de hardware, y Miles de líneas de código)	Meses-persona

Además, para evaluar la eficacia del método DT en los cinco conjuntos de datos, se ajustaron dos parámetros clave: d , que es la profundidad máxima del árbol, con seis valores diferentes que oscilan entre 5 y 10, y n , que es el número máximo de nodos hoja, con 10 valores distintos que van de 11 a 20. Se entrenaron y probaron un total de 60 variantes del método DT (i.e., seis valores para d y 10 valores para n). Como se mencionó anteriormente, el RF es un modelo que utiliza varios árboles de decisión sobre distintas submuestras del conjunto de datos y promedia los resultados para mejorar la precisión de las predicciones. En este estudio se utilizó la función *RandomForestRegressor* y la biblioteca *SKlearn* 1.0.1 para implementar la técnica de RF. El modelo también se configuró tomando como base seis características de entrada y se obtuvo una de salida que es el esfuerzo estimado. Además, se optó por usar el MSE como medida de la calidad de las divisiones y se estableció un mínimo de dos muestras, tanto para dividir un nodo como para permanecer en un nodo hoja. Para

evaluar el rendimiento del RF se ajustaron dos parámetros principales: d , que es la profundidad máxima del árbol, con valores que van de 5 a 10, y t , que es el número de árboles en el bosque, con valores de 50, 100, 150, 200, 250 y 300. En total se probaron 36 combinaciones diferentes del RF, resultantes de las variaciones en estos dos parámetros.

El GBT es una técnica que construye un modelo aditivo de manera progresiva, optimizando funciones de pérdida diferenciables en cada etapa. Cada paso implica ajustar un árbol de regresión en el gradiente negativo de la función de pérdida específica. Para implementar esta técnica, se utilizó la función *GradientBoostingRegressor* y la biblioteca *SKlearn* 1.0.1. El GBT se diseñó con seis características de entrada y una salida que estima el esfuerzo requerido. Algunas de las configuraciones clave incluyeron el uso de MSE para optimizar la función de pérdida, la función *friedman_mse* para evaluar la calidad de las divisiones, un mínimo de tres muestras para dividir un nodo, y un total de 200 etapas de impulso. Para evaluar la eficacia del GBT en los cinco conjuntos de datos, se ajustaron dos parámetros principales: d , que representa a la profundidad máxima de los estimadores individuales de regresión con valores que van de 1 a 5, y l , que es la tasa de aprendizaje que controla cuánto contribuye cada árbol, con valores de 0.05, 0.1, 0.15, 0.2 y 0.25. En total, se probaron 25 combinaciones diferentes resultantes del GBT con las variaciones en estos parámetros. Para implementar la técnica del MLP se utilizó la función *MLPRegressor* y la biblioteca *SKlearn* 1.0.1 con seis entradas y una salida (i.e., esfuerzo estimado). La configuración consistió en tres capas ocultas, cada una con 100 neuronas, usando la función de activación Relu para las capas ocultas y el optimizador Adam para la optimización del peso. Para evaluar el desempeño del MLP en cinco conjuntos de datos, se ajustaron dos parámetros principales: t , que es el número máximo de iteraciones, con valores en el rango de 100 a 1000 en incrementos de 100, y L , que es la tasa de aprendizaje inicial, con valores en el rango de 0.002 a 0.006 en incrementos de 0.001. En total se entrenaron y probaron 50 variantes del método MLP, derivadas de la combinación de 10 valores de iteraciones y 5 valores de tasas de aprendizaje.

Las redes neuronales de LSTM son una evolución de las redes neuronales recurrentes puesto que agregan estructuras de puertas a su arquitectura. A diferencia de las redes tradicionales que tienen una sola entrada, las LSTM cuentan con dos conjuntos de entradas: la información actual y el vector de salida de la unidad anterior. Estas redes utilizan bloques de memoria en la capa oculta, en lugar de nodos simples. Estos bloques, conocidos como celdas de memoria, tienen tres puertas que regulan y modifican el flujo de información: la puerta de olvido, la puerta de entrada y la puerta de salida. La puerta de olvido decide qué información se debe descartar, la puerta de entrada controla la información que se retiene, y la puerta de salida determina el valor de salida y el estado actual que se pasará a la siguiente unidad. Para implementar la red neuronal de LSTM se utilizó la clase LSTM de *Keras.layers*, configurada con seis entradas y una salida (i.e., esfuerzo estimado). Cabe mencionar que esta técnica emplea la tangente hiperbólica como función de activación, la sigmoide como función de activación recurrente, y una probabilidad de abandono de 0.5. Por lo tanto, haciendo uso del optimizador Adam, se ajustaron dos parámetros clave para evaluar el rendimiento de la red neuronal de LSTM: e , que representa al número de épocas de entrenamiento con valores entre 100 y 1000 en incrementos de 100, y n , que es el número de neuronas en la capa oculta LSTM, con valores de 25, 50, 75 y 100. Combinando estos parámetros, se entrenaron y probaron 40 variantes de redes neuronales de LSTM.

Finalmente, PSO es una técnica inspirada en el comportamiento de las aves que se utiliza para encontrar soluciones óptimas a problemas de optimización. Cada partícula en PSO representa una posible solución y está caracterizada por su velocidad, posición y aptitud. A través de un proceso iterativo, las partículas actualizan continuamente su posición y velocidad, buscando mejorar su aptitud y acercarse al óptimo global. PSO es conocido por su rápida velocidad de búsqueda, fácil convergencia

y gran eficiencia. En este estudio se utilizó PSO para optimizar los hiperparámetros de una red neuronal de LSTM. El peso de la capa oculta LSTM fue utilizado como la entrada del enjambre de partículas, mientras que el error de salida inicial del LSTM sirvió como la aptitud del enjambre de partículas. Posteriormente, las partículas actualizaron sus parámetros basándose en su mejor rendimiento individual y el mejor rendimiento global del enjambre. Para configurar el LSTM optimizado con PSO, se comenzó con una población de 10 partículas y se realizaron 50 iteraciones. Los factores de aceleración se fijaron en 1.5. Este enfoque permitió inicializar los hiperparámetros de la red neuronal de LSTM de manera eficiente, mejorando así la precisión y eficiencia del modelo. Las actividades de este método LSTM optimizado por PSO incluyeron la configuración del conjunto de entrenamiento y el uso de PSO para determinar los valores óptimos de los hiperparámetros antes del proceso de entrenamiento

2.4.3.3. Resultados

La evaluación del rendimiento de cada uno de los métodos presentados se realizó a través de cuatro métricas de evaluación: MAE, RMSE, la Mediana de la Magnitud del Error Absoluto (MdMAE, por sus siglas en inglés) y R^2 . El método LSTM optimizado demostró ser superior, obteniendo valores más bajos para las cuatro métricas utilizadas en la evaluación (véase la Tabla 6). Sin embargo, se observaron algunas limitaciones, especialmente en conjuntos de datos pequeños como Albrecht y Kemerer, que tienen un número reducido de observaciones y baja dimensionalidad. Comparando los valores de las métricas recogidas y mostradas en la Tabla 6, el investigador concluyó que el método LSTM optimizado es una opción más eficiente para estimar el esfuerzo en los proyectos de software.

2.4.4. Conclusiones del estado del arte

La Tabla 7 muestra la comparación entre los estudios descritos anteriormente con el fin de resaltar la importancia crítica de seleccionar conjuntos de datos con una cantidad suficiente de datos para la estimación precisa del esfuerzo en el desarrollo de software. Los conjuntos de datos pequeños, como Albrecht y Kemerer, presentan limitaciones significativas que afectan la precisión de los modelos predictivos. Por lo tanto, es esencial utilizar conjuntos de datos más grandes y diversificados, como el conjunto de datos China, para mejorar la exactitud y fiabilidad de las estimaciones de esfuerzo en el desarrollo de software. Además, un factor común entre estos estudios es que carecen de detalles sobre si los proyectos de los conjuntos de datos fueron implementados en pequeñas organizaciones y si éstas utilizaron Scrum. Por lo tanto, para mejorar la relevancia y aplicabilidad práctica de este tipo de investigaciones, sería beneficioso incluir información más detallada sobre el contexto en el cual fueron desarrollados los proyectos, específicamente la metodología que se siguió y si el conjunto de datos proviene o no de una organización pequeña.

Aunado a lo anterior, los estudios incluidos en el estado del arte hacen evidente una constante: los modelos entrenados con conjuntos de datos limitados o insuficientes tienden a presentar un alto margen de error o problemas de sobreajuste (*overfitting*), lo que reduce su aplicabilidad en escenarios reales. Por el contrario, la disponibilidad de repositorios robustos y con una cantidad suficiente de datos históricos permite a los algoritmos identificar patrones complejos, relaciones ocultas y variabilidades propias de los proyectos de software con mayor precisión. En conclusión, este estado del arte resalta la importancia crítica de una selección estratégica y meticulosa de los conjuntos de datos. Sin esta base cuantitativa, incluso los modelos de aprendizaje computacional más avanzados verán limitado su potencial práctico.

Tabla 6. Comparativa de métricas de evaluación para cada técnica evaluada. Nota: Traducido de Iordan (2024)

Conjunto de datos	Técnica/método	Métricas			
		MAE	MdMAE	RMSE	R ²
Albrecht	KNN	11.228	4.556	18.202	0.437
	DT	6.975	3.825	10.213	0.561
	RF	6.015	3.579	7.726	0.593
	GBT	5.892	3.092	7.094	0.609
	MLP	7.555	3.256	8.208	0.547
	LSTM	4.870	2.911	6.560	0.631
	LSTM optimizado	3.604	2.902	4.717	0.697
Kemerer	KNN	75.519	57.933	90.933	0.351
	DT	95.853	81.802	106.226	0.309
	RF	82.739	69.174	83.726	0.327
	GBT	60.197	58.682	64.045	0.392
	MLP	48.145	47.427	58.217	0.438
	LSTM	42.094	26.206	57.560	0.493
	LSTM optimizado	35.294	13.747	41.412	0.579
COCOMO81	KNN	221.619	35.783	446.987	0.793
	DT	275.979	48.617	514.792	0.774
	RF	235.857	43.925	487.283	0.732
	GBT	209.198	39.826	463.782	0.812
	MLP	199.513	57.833	355.245	0.826
	LSTM	178.051	30.642	245.153	0.897
	LSTM optimizado	133.096	30.142	217.794	0.986
China	KNN	1,747.048	611.602	3,922.755	0.762
	DT	1,575.633	755.642	2,848.557	0.793
	RF	1,409.324	621.782	2,792.372	0.821
	GBT	1,287.382	504.672	2,563.824	0.865
	MLP	1,383.713	517.898	2,892.196	0.843
	LSTM	865.122	272.353	2,034.574	0.902
	LSTM optimizado	331.089	111.213	873.102	0.951
Desharnais	KNN	1,422.878	921.667	1,847.561	0.658
	DT	1,953.361	1,453.083	2,526.071	0.563
	RF	1,792.837	1,284.947	2,379.063	0.581
	GBT	1,549.792	901.925	2,095.827	0.639
	MLP	1,587.619	905.841	2,176.591	0.624
	LSTM	1,404.571	880.121	1,960.007	0.662
	LSTM optimizado	1,253.781	849.929	1,535.852	0.683

Tabla 7. Comparativa entre estudios presentados en el estado del arte

Título del estudio	Algoritmos utilizados	Conjunto de datos	Métricas de evaluación	
<i>“An optimized LSTM neural network for accurate estimation of software development effort”</i>	KNN, DT, RF, GBT, MLP, LSTM, y LSTM optimizada.	Albrecht, Kemerer, COCOMO81, China y Desharnais	MAE, Error Absoluto Medio Mediano (MdAE, por sus siglas en inglés), RMSE, y R ² .	COCOMO81: LSTM optimizada – MAE: 133.096, MdAE: 30.142, RMSE: 217.794 y R ² : 0.986.
<i>“Effort and cost estimation using decision tree techniques and story points in Agile software development”</i>	DT, RF, y AdaBoost.	Propuesto por Ziauddin et al., 2012	MRE, R ² , MSE, y MMRE.	RF – PRED: 96.92; AdaBoost – PRED: 96.44; DT – PRED: 96.57.
<i>“Applying soft computing techniques for software project effort estimation modelling”</i>	Regresión lineal, SVR, Regresión logística, DT, Agregación de <i>Bootstrap</i> (<i>bagging</i>) y RF.	China	MMRE, MAE, MSE, R ² y MRE.	DT – MMRE: 0.39, MAE: 0, RMSE: 0.01, MER: 0.37, MdMMRE: 0, R ² : 1 y PRED: 100.

3. Metodología: Implementación del método de estimación

A continuación se describen tanto los recursos utilizados, como los módulos principales que conforman la solución propuesta en esta tesis. En este sentido, en la sección 3.1 se detallan los recursos tecnológicos utilizados, tanto de hardware como de software, para desarrollar e implementar cada método y la aplicación web de estimación de esfuerzo. Por otro lado, la sección 3.2 describe la metodología empleada en la construcción de los conjuntos de datos sintéticos para proyectos de software y *sprints*. Además, la sección 3.3 explica la implementación enfocada en el método de estimación utilizado para la obtención de la predicción. Por último, la sección 3.4 describe el desarrollo de la aplicación web donde se integrará el modelo de estimación obtenido.

3.1. Especificaciones de hardware y software

Con relación a la solución planteada en esta tesis, a continuación, se detallan las especificaciones de hardware y software consideradas. Para el desarrollo e implementación de los diferentes algoritmos de aprendizaje computacional para la estimación del esfuerzo, se requirió de un servidor robusto capaz de manejar procesamiento intensivo debido a la complejidad computacional de tales algoritmos. Las especificaciones del servidor utilizado, Dell Precision Tower 7910, se detallan en la Tabla 8.

Tabla 8. Especificaciones de hardware.

Especificaciones	Descripción
Procesador	Intel ® Xeon ® E5-2620v4
Velocidad de procesador	2.1GHz
Núcleos	32
Memoria RAM	32 GB
GPU	Nvidia Quadro M4000
Almacenamiento	1TB HDD
Sistema operativo	Ubuntu 22.04 LTS

El servidor Dell Precision Tower 7910 fue seleccionado por su capacidad para manejar grandes volúmenes de datos y realizar cálculos en paralelo, lo que permite optimizar el rendimiento de los algoritmos. Los módulos del proyecto fueron implementados en el lenguaje de programación Python versión 3.12.7, elegido por su versatilidad y su amplio ecosistema de bibliotecas públicas como Scikit-

learn, Pandas y Numpy, fundamentales para la implementación y análisis de modelos de aprendizaje computacional. Además, se utilizó Visual Studio Code como IDE por su flexibilidad y soporte para múltiples lenguajes, mientras que Jupyter Notebook facilitó la experimentación y visualización de resultados durante el prototipado.

3.2. Construcción de los conjuntos de datos sintéticos

Antes de aplicar los algoritmos de aprendizaje computacional, resultó fundamental construir un conjunto de datos que se adaptara a la problemática planteada en esta tesis. Si bien existen conjuntos de datos creados bajo el marco de metodologías ágiles, muchos no representan las condiciones reales de las pequeñas organizaciones desarrolladoras de software, donde la mayoría de las veces los recursos son limitados, los equipos son reducidos y es imprescindible una mayor flexibilidad para responder a los cambios constantes en el proyecto. Estas características suelen quedar fuera del alcance de los conjuntos de datos públicos, que tienden a enfocarse en escenarios más genéricos (e.g., organizaciones desarrolladoras de software con procesos bien establecidos, recursos y tiempo suficientes). Por lo tanto, construir un conjunto de datos propio permite abordar estas limitaciones al incorporar factores específicos descritos en la sección 2.3.2, tales como niveles de incertidumbre, velocidad del equipo, etc. Además, esto facilita la generación de estimaciones más precisas y relevantes, ajustadas a los desafíos únicos de las pequeñas organizaciones, ofreciendo así una herramienta más confiable para apoyar la toma de decisiones en proyectos de desarrollo de software bajo un enfoque ágil.

Para desarrollar e implementar la herramienta de estimación de esfuerzo en proyectos Scrum se ha creado un conjunto de datos basado en el Método de Estimación Generalizada (GEM, por sus siglas en inglés) propuesto por Bahlerao e Ingle (2011). Este método sigue un enfoque iterativo alineado con los principios de la estimación ágil que inicia por identificar la clase de dominio del proyecto y de los factores esenciales, los cuales son definidos de acuerdo con el comportamiento proyectado y la orientación hacia el tipo de usuario destinatario. GEM tiene dos fases: Estimación Temprana (EE, por sus siglas en inglés) y Estimación Iterativa (IE, por sus siglas en inglés). La EE analiza el dominio del proyecto, identifica los factores vitales y riesgos, y es una fase sin costo, diseñada para generar confianza en el cliente. En cambio, la IE es facturable y se realiza durante la planificación de cada iteración, ajustando el tamaño del software en desarrollo al considerar la velocidad real del equipo, el impacto de los factores relevantes y los riesgos asociados a los requisitos y recursos. Este proceso permite ajustar las estimaciones en función de los nuevos requisitos y comentarios de las partes interesadas, adaptándose así a cambios en el proyecto. A continuación, la Tabla 9 muestra la terminología utilizada por los autores para construir el conjunto de datos. Para un mejor entendimiento de los factores vitales, se proporciona una descripción específica. De acuerdo con Bahlerao e Ingle (2011), estos factores se definen de la siguiente manera:

- El primer factor es el *rendimiento*, el cual está determinado por los requisitos de velocidad y eficiencia del proyecto. Se califica de acuerdo con la intensidad con una escala de alto, medio y bajo, siendo alto si el proyecto requiere un rendimiento elevado y bajo si no es así.
- La *configuración* depende de si el proyecto requiere hardware o software especializado, siendo alta si es necesario y baja si no lo es.
- El procesamiento complejo (o *complejidad*) se refiere a las necesidades de procesamiento avanzado, como cálculos científicos o reglas de negocio, con una clasificación alta en proyectos que lo requieren y baja en los que no.

- La *transferencia de datos* evalúa el volumen de intercambio de información con otros sistemas, clasificándose como alta cuando el volumen es elevado y como baja cuando es moderado.
- La *seguridad* se clasifica de acuerdo con los niveles de protección y autenticación requeridos, variando de acuerdo con la sensibilidad del proyecto, que podría ser alto, medio y bajo. Alto para un nivel de seguridad alta y baja cuando no es así.
- La *facilidad de operación* mide si el software permite la recuperación automática y una mínima intervención del usuario; se clasifica como alta si cuenta con estas características, y baja si no las tiene.
- Para proyectos distribuidos globalmente, donde los costos de comunicación son elevados, el *factor de múltiples sitios* (u operación múltiple) se clasifica como alto; sin embargo, la clasificación es baja en proyectos locales.

Tabla 9. Terminología del Método de Estimación Generalizada. Nota: Traducido de Bahlerao e Ingle (2011)

Términos utilizados	Descripción	Valores definidos por los autores
Dominio del proyecto (<i>PD</i>)	Actividades y atributos específicos involucrados en el desarrollo del proyecto.	Aplicación web ($AW=1$), Sistemas de información ($MIS=4$) y Proyecto militar ($M=9$).
Factores vitales (<i>v</i>)	Factores que afectan el desarrollo del proyecto, como clases sociológicas, tecnológicas, ergonómicas y del proyecto.	v1: Rendimiento, v2: Configuración, v3: Complejidad, v4: Transferencia de datos, v5: Seguridad, v6: Múltiples sitios, v7: Facilidad de operación.
Pesos (<i>w</i>)	Valor asignado a cada factor vital según su prioridad. La suma de todos los factores vitales debe ser 1.	Nivel 1 (v1, v2, v3, $w = 0.2$), Nivel 2 (v4, v5, $w = 0.15$), Nivel 3 (v6, v7, $w = 0.05$).
Nivel de intensidad (<i>I</i>)	Nivel de influencia de un factor vital en el proyecto.	Bajo (L), medio (M) y alto (H).
Factor de incertidumbre (<i>UF</i>)	Nivel de desconocimiento de los requisitos, tecnología o recursos del proyecto.	0.2, 0.4, 0.6 y 0.8.
Puntos de historia (<i>SP</i>)	Estimación del esfuerzo para desarrollar una historia de usuario.	25 por historia de usuario.
Factor de prioridad (<i>PF</i>)	Nivel de prioridad de cada factor vital, dependiendo del peso y nivel de intensidad.	Se obtiene usando la ecuación 6.
Valor no ajustado (<i>UV</i>)	Suma de todos los factores de prioridad de los factores vitales.	Se obtiene usando la ecuación 7.
Nuevos puntos de historia (<i>NSP</i>)	Estimación del esfuerzo para la historia de usuario considerando los factores vitales.	Se obtiene usando la ecuación 8.

Términos utilizados	Descripción	Valores definidos por los autores
Tamaño del proyecto (<i>SOP</i>)	Medido en NSP para denotar el tamaño total del proyecto.	Se obtiene usando la ecuación 9.
Duración del proyecto (<i>DOP</i>)	Tiempo necesario para desarrollar el proyecto, medido en semanas, meses o años.	Se obtiene usando la ecuación 10.
Velocidad del equipo	Numero de SP desarrollados por el equipo en un <i>sprint</i> .	8

Los factores vitales mencionados anteriormente son cuantificados en tres niveles de intensidad: bajo ($L = 1$), medio ($M = 4$) y alto ($H = 9$). A partir de estas combinaciones de factores y valores de incertidumbre se aplican las ecuaciones del método GEM para realizar los cálculos de estimación de esfuerzo. Los términos restantes de la Tabla 9 se describen y definen a continuación. El **Factor de prioridad** (PF, por sus siglas en inglés) de cada factor vital se obtiene a través de la Ecuación (6).

$$PF(v_i) = w_i * I_i \quad (6)$$

donde w_i es cada peso de cada factor vital, I_i es el nivel de intensidad (véase la Tabla 9).

Por otro lado, el **Valor no ajustado** (UV, por sus siglas en inglés), es la suma de los factores de prioridad, se calcula mediante la Ecuación (7).

$$UV = \sum_{i=1}^n PF(v_i) \quad (7)$$

Los **Nuevos puntos de historia** (NSP, por sus siglas en inglés) para cada historia de usuario, ajustado según el dominio del proyecto (PD, por sus siglas en inglés), el factor de incertidumbre (UF, por sus siglas en inglés) y UV, se obtiene a través de la Ecuación (8).

$$NSP_j = SP_j + PD + UF * UV \quad (8)$$

donde SP_j es el punto de historia base y representa el dominio del proyecto (e.g., valores como 1 para aplicaciones web, 4 para proyectos de sistemas de información, y 9 para proyectos militares).

El **Tamaño del proyecto** (SOP, por sus siglas en inglés) se calcula a través de Ecuación (9).

$$SOP = \sum_{j=1}^m NSP_j \quad (9)$$

Finalmente, la **Duración del proyecto** (DOP, por sus siglas en inglés) es obtenida mediante la Ecuación (10).

$$DOP = SOP / (\text{velocidad del equipo}) \quad (10)$$

A partir de las ecuaciones descritas se estableció una metodología estructurada que permitió replicar el conjunto de datos de manera precisa y sistemática, la cual se describe en la siguiente sección. Estas ecuaciones proporcionan los fundamentos necesarios para cuantificar y ajustar los valores de los factores vitales involucrados en el proyecto.

3.2.1. Metodología para el diseño y construcción de la base de datos

Para replicar el conjunto de datos compuesto por proyectos que incluyen historias de usuario, propuesto por Bhalerao e Ingle (2011), se adoptó una metodología que considera tanto el algoritmo descrito por los autores, como la investigación realizada por Ramessur y Nagowah (2021). En la Figura 3.1 se presenta una secuencia de los pasos detallados para obtener y replicar el conjunto de datos de manera precisa, asegurando así que cada factor relevante en la estimación de esfuerzo sea considerado en el proceso.

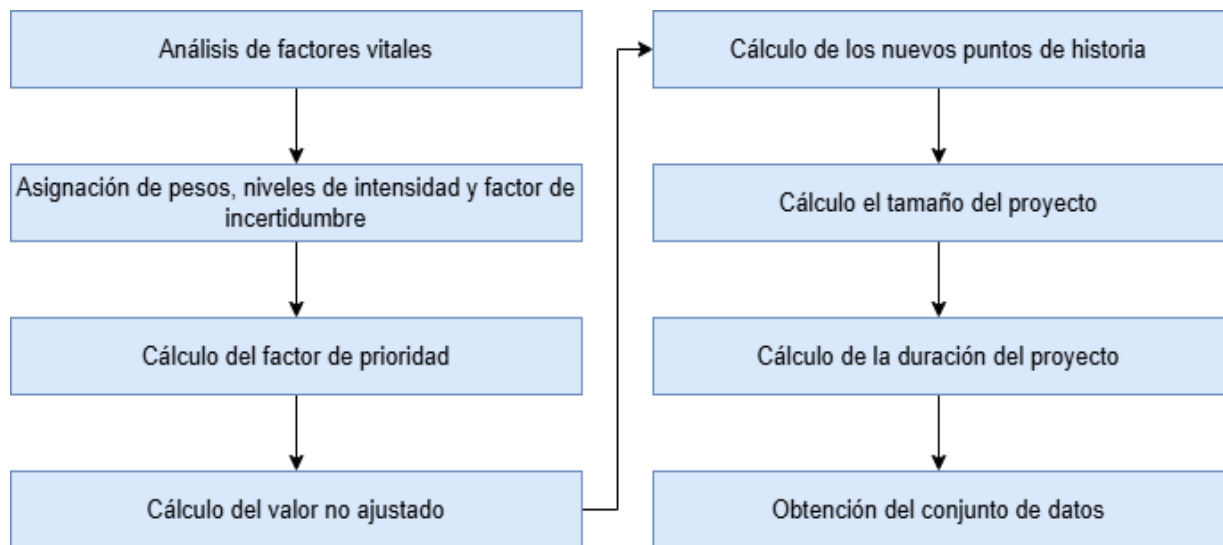


Figura 3.1. Metodología utilizada en la construcción de la base de datos. Nota: Traducida de Ramessur y Nagowah (2021).

Las actividades de la Figura 3.1 se definen de la siguiente manera:

1. **Análisis de factores vitales:** Se identifican y analizan los factores vitales que influyen en el proyecto, tales como rendimiento, complejidad y seguridad.
2. **Asignación de pesos, niveles de intensidad y factor de incertidumbre:** Se define la relevancia de cada factor mediante la asignación de pesos, niveles de intensidad (i.e., bajo, medio, alto) y el grado de incertidumbre asociado.
3. **Cálculo del factor de prioridad:** Se determina la importancia relativa de cada factor multiplicando el peso asignado por el nivel de intensidad (véase la Ecuación (6)).
4. **Cálculo del valor no ajustado:** Se suman todos los factores de prioridad para obtener un valor no ajustado que representa la influencia acumulada de los factores vitales (véase la Ecuación (7)).
5. **Cálculo de los nuevos puntos de historia:** Se ajustan los puntos de historia iniciales para reflejar el impacto de los factores vitales y la incertidumbre en el proyecto (véase la Ecuación (8)).

6. Cálculo del tamaño del proyecto: Se suman los nuevos puntos de historia para obtener una medida del tamaño total del proyecto (véase la Ecuación (9)).
7. Cálculo de la duración del proyecto: Se divide el tamaño del proyecto por la velocidad del equipo para estimar la duración del proyecto (véase la Ecuación (10)).
8. Obtención del conjunto de datos: Finalmente, se recopilan todos los valores calculados en un conjunto de datos estructurado para su análisis posterior y referencia.

Este enfoque garantiza que todos los factores relevantes sean considerados en la estimación y que el conjunto de datos resultante sea amplio y representativo para diferentes escenarios en el desarrollo ágil, proporcionando así una base sólida para la implementación de la herramienta de estimación de esfuerzo propuesta en esta tesis.

3.3. Método de estimación de esfuerzo

Una vez obtenidos los conjuntos de datos sintéticos de proyectos de software y *sprints* generados con base a Bhalariao e Ingle (2011) y Ramessur y Nagowah (2021), resultó importante implementar el método de estimación capaz de proporcionar una predicción cercana a las realizadas por la organización. Esta sección tiene como objetivo presentar los módulos que forman la obtención del modelo de estimación. En este sentido, la Figura 3.2 muestra el diagrama general de los módulos del proyecto, en ésta se observa, como primera fase, la preparación de los datos, donde se preprocesan y seleccionan las características relevantes del conjunto de datos para obtener un conjunto balanceado. En la segunda fase, se describió la obtención del modelo de estimación a través de una búsqueda de hiperparámetros utilizando *Grid Search*, o búsqueda de malla (Ma et al., 2015), y una búsqueda fina. De esta manera, se pretendió obtener la mejor configuración del rango de valores seleccionado para utilizarla en los modelos para la predicción del esfuerzo. Como última fase, se realizó la integración del “mejor modelo” obtenido de estimación en la aplicación web.

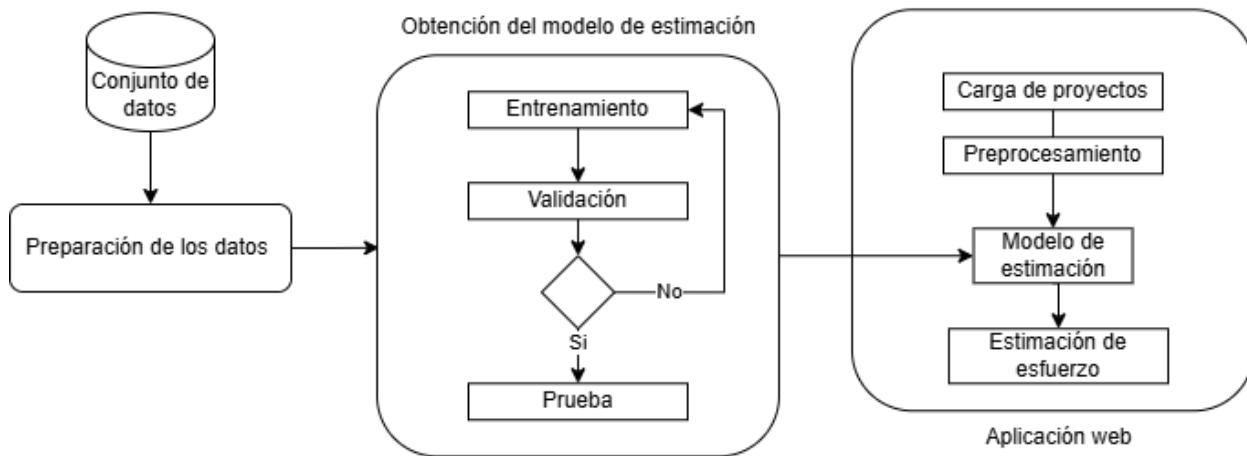


Figura 3.2. Diagrama general de los módulos para obtener el modelo de estimación.

En las siguientes secciones se proporciona una descripción de cada una de estas etapas.

3.3.1. Preparación de los datos

Con base a la metodología CRISP-DM descrita en la sección 2.2.2, la preparación de los datos es fundamental para garantizar que la siguiente fase de la metodología, en la que se realiza la selección de los métodos de estimación, se base en el “mejor modelo”. Antes de comenzar a aplicar los métodos de estimación de esfuerzo, es necesario preprocesar el conjunto de datos para asegurar que la información es relevante para el entrenamiento del modelo. El preprocesamiento se describe como una etapa importante en los proyectos de aprendizaje computacional, ya que los datos recolectados a menudo están en forma natural y pueden incluir valores faltantes, valores atípicos o características que degradan la precisión del modelo.

En este sentido, se realizó un análisis de las variables de los conjuntos de datos generados, con el fin de verificar la coherencia de los valores y detectar posibles inconsistencias. Una vez terminado el análisis, se procedió a realizar el particionamiento del conjunto de datos. Cabe resaltar que es importante determinar, qué particionamiento se utilizará para el entrenamiento de cada uno de los modelos anteriormente mencionados. Por lo tanto, se delimitó el porcentaje de los datos destinados a entrenamiento, validación y prueba. La Figura 3.3 muestra el particionamiento del conjunto de datos, el cual se subdividió en dos subconjuntos, uno que representa el 80% de todos los datos y se utilizó para el entrenamiento, y otro que se utilizó para las pruebas que comprende el 20%. Dentro del 80% para entrenamiento, este se subdividió en dos subconjuntos, uno que representa el 75% utilizado para el entrenamiento del modelo y el otro 25% para validación (para búsqueda de los mejores hiperparámetros). Esta división hizo posible entrenar el modelo en una porción de los datos y validarlo en un nuevo conjunto de datos. De esta manera, el modelo pudo ser probado para una evaluación confiable de rendimiento y generalización.

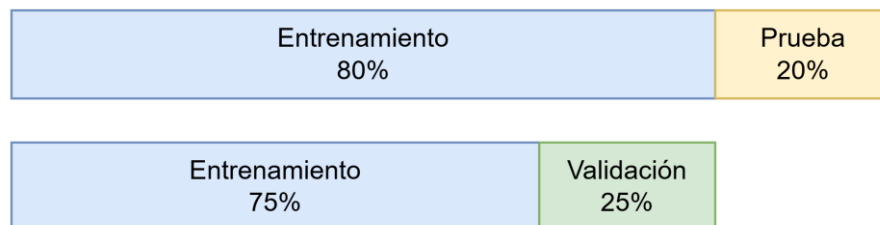


Figura 3.3. Particionamiento del conjunto de datos.

Seguidamente, se realizó el aumento de datos aplicado a los conjuntos de entrenamiento y validación, utilizando la técnica SMOGN. Sin embargo, dado que se utilizará el “mejor modelo”, se requirió seleccionar la mejor estrategia de combinación para el sobremuestreo y submuestreo. Para ello, se consideraron porcentajes de sobremuestreo del 100% y 200%, mientras que, para el submuestreo se consideraron el 10%, 20%, 30% 40% y 50%, con el fin de lograr balancear el conjunto de datos. De manera posterior, se realizó la normalización de los datos, utilizando el módulo *preprocessing* de la biblioteca de *Scikit-learn*. Este procedimiento estandarizó las características de tal manera que cada una tuviera una media de 0 y desviación estándar unitaria. Ahora bien, con el fin de evaluar si todas las variables del conjunto de datos contribuyen de manera significativa a la predicción de la estimación, se incorporó una etapa de análisis de selección de características. Este proceso tiene como objetivo identificar aquellas variables con mayor relevancia respecto a la variable objetivo y determinar si la eliminación de características menos informativas puede mejorar la capacidad de generalización de los modelos. La selección adecuada de características no solo puede reducir la complejidad del modelo, sino también contribuir a obtener predicciones más precisas y robustas ante nuevos datos.

En el contexto de esta tesis, la etapa de preparación de los datos es muy relevante, ya que se busca desarrollar una herramienta que sea útil para pequeñas organizaciones desarrolladoras de software, donde los recursos y el tiempo son limitados, y cada estimación debe reflejar fielmente las dinámicas reales de sus proyectos. Por ejemplo, la selección y por consecuencia la reducción del número de características, no solo mejora el rendimiento del modelo, sino que también garantiza que las predicciones sean prácticas y aplicables en entornos ágiles.

3.3.2. Obtención del modelo de estimación

Resultó fundamental analizar cada modelo para determinar cuál de ellos era el mejor, puesto que cada uno tiene diferentes características y es empleado en diferentes situaciones. A partir de la revisión de la literatura realizada en la sección 2.3.3, se implementaron nueve modelos individuales, los cuales se emplearon como modelos base para la construcción de siete modelos de ensamble (i.e., modelos en conjunto), considerando como criterio su frecuencia de uso y su nivel de replicabilidad presentada en las investigaciones de diversos autores. Lo anterior se consideró, para evaluar y comparar los resultados de los artículos relacionados con los obtenidos en la presente tesis. Los modelos individuales seleccionados fueron KNN, MLP, ELM, SVR, DT, RF y LR, Lasso, Ridge. Seleccionar adecuadamente los parámetros de cada modelo, es un paso necesario para su correcto funcionamiento. En este sentido, las Tablas 10 a 16 describen los principales parámetros utilizados en cada uno de los modelos implementados. La Tabla 10 presenta los parámetros utilizados para el modelo *ELMRegressor*. Estos parámetros permiten definir la estructura y configuración de una red neuronal artificial, la función de activación empleada, el algoritmo de optimización y la tasa de aprendizaje utilizada durante el entrenamiento del modelo.

Tabla 10. Parámetros ELM.

Parámetros	Descripción
<i>n_hidden</i>	Número de neuronas en la capa oculta.
<i>activation</i>	Función de activación, utilizada en la capa oculta.

De manera similar la Tabla 11, muestra la configuración para la función *MLPRegressor*.

Tabla 11. Parámetros MLP.

Parámetros	Descripción
<i>hidden_layer_sizes</i>	Número y tamaño de las capas ocultas en la red neuronal.
<i>activation</i>	Función de activación, utilizada en la capa oculta.
<i>solver</i>	Algoritmo empleado para optimizar el entrenamiento de la red.
<i>learning_rate_init</i>	Tasa de aprendizaje inicial para la optimización del modelo.

La Tabla 12, muestra la configuración para la función *KNeighborsRegressor* de la biblioteca *Scikit-learn*.

Tabla 12. Parámetros KNN.

Parámetros	Descripción
<i>n_neighbors (k)</i>	Número <i>k</i> de vecinos considerados por el modelo.

<i>metric</i>	Fija la métrica de distancia como ‘minkowski’.
---------------	--

Asimismo, la Tabla 13 muestra la descripción de cada parámetro utilizado en la función *DecisionTreeRegressor*.

Tabla 13. Parámetros DT.

Parámetros	Descripción
<i>max_depth (d)</i>	Profundidad máxima d del árbol.
<i>max_leaf_nodes (n)</i>	Número máximo n de nodos hoja.
<i>min_samples_split</i>	Número mínimo de muestras necesarias para dividir un nodo, fijado en 2.

En la Tabla 14 se describe cada parámetro utilizado en la función *RandomForestRegressor*.

Tabla 14. Parámetros RF.

Parámetros	Descripción
<i>n_estimators (t)</i>	Número de árboles t en el bosque.
<i>max_depth (d)</i>	Profundidad máxima d de los árboles.
<i>min_samples_split</i>	Número mínimo de muestras necesarias para dividir un nodo, fijado en 2.

Por otra parte, la Tabla 15 muestra la descripción de cada parámetro utilizado en la función SVR.

Tabla 15. Parámetros SVR.

Parámetros	Descripción
<i>kernel</i>	Tipo de núcleo usado en la regresión de soporte vectorial.
C	Parámetro de regularización que controla el ajuste del modelo.
<i>epsilon</i>	Define un margen dentro del cual no se penalizan errores.

En la Tabla 16 se muestra la descripción de cada parámetro utilizado en la función LR.

Tabla 16. Parámetros LR.

Parámetros	Descripción
<i>fit_intercept</i>	Le permite al modelo dibujar la línea con una “altura inicial” en lugar de obligarla a pasar por el origen. Es como darle libertad de empezar un poco arriba o abajo.
<i>copy_X</i>	Hace una copia de los datos para no modificar los originales.
<i>n_jobs</i>	Número de procesadores que usa.

En la Tabla 17 se muestra la descripción de cada parámetro utilizado en la configuración del modelo *Lasso*. Los parámetros seleccionados permiten controlar el nivel de regularización aplicado al modelo y el número máximo de iteraciones durante el proceso de entrenamiento.

Tabla 17. Parámetros Lasso.

Parámetros	Descripción
<i>alpha</i>	Puede reducir algunos coeficientes exactamente a cero, eliminando variables menos relevantes.
<i>max_iter</i>	Número máximo de iteraciones para encontrar la solución.

En la Tabla 18 se presenta la descripción de cada parámetro utilizado en la configuración del modelo *Ridge*. Estos parámetros permiten definir la intensidad de la regularización y el número máximo de iteraciones empleadas para obtener la solución del modelo.

Tabla 18. Parámetros Ridge.

Parámetros	Descripción
<i>alpha</i>	Reduce el impacto de las variables, pero normalmente no las elimina por completo.
<i>max_iter</i>	Número máximo de iteraciones permitidas.

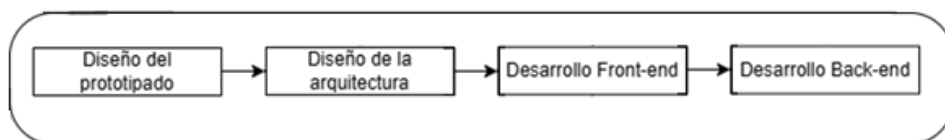
A continuación, la Tabla 19 muestra los modelos de ensamble heterogéneos (EH) obtenidos, de la combinación de los modelos individuales previamente mencionados. Cada modelo individual utilizado en cada EH, fue empleado utilizando diferentes parámetros, para poder ser entrenado con los “mejores hiperparámetros” encontrados.

Tabla 19. Ensamblados heterogéneos utilizados.

Ensamble	Modelos involucrados
Ensamble 1 (EH1)	ELM + SVR, Metamodelo RF
Ensamble 2 (EH2)	ELM + SVR + DT + RF + LR, Metamodelo SVR
Ensamble 3 (EH3)	ELM + SVR + DT + RF + LR, Metamodelo RF
Ensamble 4 (EH4)	ELM + SVR + LR, Metamodelo RF
Ensamble 5 (EH5)	ELM + SVR + Lasso, Metamodelo RF
Ensamble 6 (EH6)	ELM + SVR + Ridge, Metamodelo RF
Ensamble 7 (EH7)	KNN + MLP + SVR + DT + RF, Metamodelo RF

3.4. Desarrollo del prototipo de aplicación web

En esta sección se presenta el desarrollo del prototipo de la aplicación web. La Figura 3.4 muestra las fases requeridas para desarrollar la aplicación web que permite la ejecución del método de estimación.

**Figura 3.4.** Fases para el desarrollo de la aplicación web.

Como primer paso, se llevó a cabo el diseño del prototipo en Figma, esto permitió visualizar la estructura y funcionalidades principales. En consecuencia, con base en el prototipado, se definió la arquitectura del sistema como se muestra en la Figura 3.5.

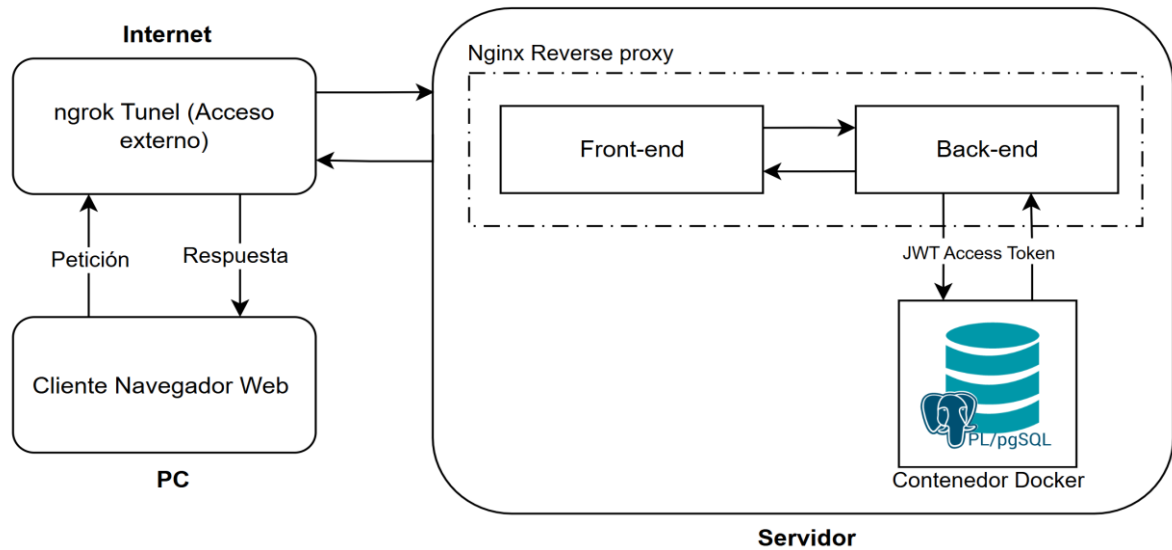


Figura 3.5. Arquitectura del sistema.

Posteriormente, se desarrolló el *Front-end*, el cual contiene todos los diseños definidos en el prototipado, y finalmente se desarrolló el *Back-end* que permite realizar las operaciones que requiere el modelo de estimación. En consonancia con lo anterior, la Figura 3.5 muestra la arquitectura general del sistema, en donde el servidor local actúa como el corazón de la aplicación, ya que aloja los principales componentes del sistema. Nginx gestiona las peticiones entrantes y actúa como un proxy inverso que hace de punto de entrada y distribuye el tráfico a los servicios. Mientras que, ngrok genera un túnel seguro mediante internet que permite el acceso externo a la aplicación. El *Front-end* responde a la petición de la ruta raíz y permite la navegación del sistema, por otro lado, el *Back-end* responde a las peticiones de la ruta a través de la interfaz de la API REST, asumiendo la lógica de negocio y gestionando el acceso y la base de datos.

Por otro lado, la Figura 3.6 muestra las dos partes principales que conforman la arquitectura Cliente-Servidor que se consideró: la interfaz de la aplicación (i.e., *Front-end*) y la parte de la lógica de negocio donde se contempla el modelo de predicción (i.e., *Back-end*). El *Front-end* está desarrollado con Next.js y Tailwind CSS; mientras que el *Back-end* está construido con Python y un marco de trabajo denominado FastAPI, al que se integra un modelo de predicción de esfuerzo basado en *Scikit-learn*. Aunado a esto, se utiliza una base de datos donde se almacenan los datos históricos de predicciones y la información de los usuarios finales.

Por lo tanto, la aplicación web está implementada de manera que resulte fácil de usar para el usuario a través de diferentes vistas, tales como: un formulario de entrada para realizar las predicciones de estimación de proyectos o *sprints*, además de un archivo .CSV en donde se almacenan varias muestras de proyectos y *sprints* que se utiliza para la predicción; la visualización de los resultados en una tabla que muestra información relevante y que, además, se puede descargar en formato .PDF junto con todos los proyectos y *sprints* registrados con su respectiva estimación del esfuerzo; y una sección de autenticación para la seguridad de acceso de los usuarios. En este sentido, la Tabla 20 resume los módulos y componentes implementados en la aplicación web. La aplicación desarrollada no es una aplicación de una sola página (SPA, por sus siglas en inglés) ya que su

estructura está diseñada para navegar en diferentes páginas, en lugar de cargar todo el contenido en una sola. Por lo tanto, se introduce una navegación tradicional entre varias páginas que es facilitada por el sistema de enrutamiento nativo de Next.js.

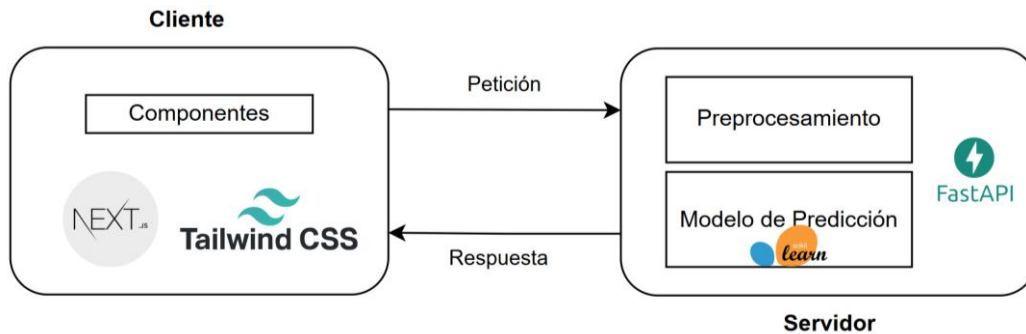


Figura 3.6. Arquitectura Cliente-Servidor propuesta para el desarrollo de la aplicación web.

Tabla 20. Módulos implementados en la aplicación web.

Rutas	Descripción
<i>Login</i>	Permite al usuario ingresar al sistema mediante un usuario y contraseña al sistema.
<i>forgot-password</i>	Permite al usuario solicitar por medio de un enlace, la recuperación de su contraseña.
<i>reset-password</i>	Permite al usuario definir una nueva contraseña.
<i>guest-estimation</i>	Permite al usuario realizar estimaciones sin iniciar sesión.
<i>(app)/page</i>	Pantalla principal del inicio de sesión utilizando usuario y contraseña.
<i>(app)/estimations</i>	Permite al usuario consultar y administrar el historial de proyectos y <i>sprints</i> ingresados.
<i>(app)/estimations/new</i>	Permite al usuario generar una nueva estimación de proyecto o <i>sprint</i> .
<i>account/change-password</i>	Permite al usuario actualizar su contraseña en el sistema.
<i>components/estimations</i>	Componente central que recopila datos de una estimación.
<i>components/projects</i>	Permite al usuario administrar y registrar los proyectos asociados a una organización.
<i>components/companies</i>	Permite al usuario registrar y administrar organizaciones en el sistema.
<i>components/main-navbar</i>	Barra de navegación que permite al usuario moverse dentro de los módulos del sistema.
<i>components/auth</i>	Verifica la sesión activa del usuario para acceder al sistema.
<i>lib</i>	Gestiona el envío y recepción de la autenticación y estimación.
<i>components/ui</i>	Componente de la interfaz.

Así, se mejora la experiencia del usuario, puesto que la navegación es fluida y organizada, sin tener que recargar la página en cada iteración. Para la comunicación del *Front-end* y *Back-end*, la aplicación utiliza una API REST desarrollada con FastAPI, con rutas definidas para manejar las

solicitudes de autenticación, predicción, historial de consultas y gestión de usuarios. La Tabla 21, muestra las rutas relevantes para la navegación de la aplicación web.

Por otra parte, el *Back-end* desarrollado recibe las solicitudes del usuario, las procesa y posteriormente da respuesta a cada una considerando dos pasos: (1) el usuario entra a la página principal de la aplicación web, realiza la carga de proyectos, *sprints* y archivo .CSV para la predicción y (2) la API recibe la solicitud, la procesa y devuelve en tipo JSON la respuesta del servidor, la cual se envía a la aplicación web para mostrar la respuesta al usuario por medio de la interfaz. Como se podrá entender, es fundamental el desarrollo óptimo del *Back-end*, ya que se deberá mejorar la gestión eficiente de las peticiones HTTP, permitiendo la descarga de predicciones anteriores de un proyecto específico, su historial de consulta y la gestión de usuarios.

Tabla 21. Rutas relevantes para la navegación.

Método	Rutas	Descripción
POST	/auth/login	Autenticación de usuario.
POST	/auth/register	Registro de nuevos usuarios.
GET	/auth/me	Permite obtener datos del usuario autenticado.
POST	/auth/forgot-password	Permite recuperar la contraseña por correo.
POST	/auth/reset-password	Permite restablecer la contraseña por medio de un enlace.
POST	/auth/change-password	Permite cambiar la contraseña con sesión activa.
GET	/companies	Obtiene la lista de empresas registradas.
GET	/companies/{company_id}/projets	Obtiene los proyectos asociados a una empresa.
DELETE	/estimations/projects/{project_id}	Elimina un proyecto y todas sus estimaciones relacionadas.
POST	/estimations/project	Genera o reestima un proyecto en específico.
GET	/estimations/projects/history/me	Consulta el historial de estimaciones de proyectos del usuario.
POST	/estimations/report/pdf	Genera un reporte (PDF) con estimaciones de proyectos y <i>sprints</i> .
POST	/estimations/sprint	Genera o reestima un <i>sprint</i> en específico.
GET	/estimations/sprints/history/me	Consulta el historial de estimaciones de <i>sprints</i> del usuario.
DELETE	/estimations/sprints/{sprint_id}	Elimina un <i>sprint</i> específico.
DELETE	/estimations/sprints	Elimina varios <i>sprints</i> en una sola ocasión.
POST	/estimations/csv	Carga un archivo .CSV para estimar varios proyectos y <i>sprints</i> .
POST	/estimations/guest/project	Genera una estimación de un proyecto sin guardar datos.
POST	/estimations/guest/sprint	Genera una estimación de un <i>sprint</i> sin guardar datos.

4. Resultados

A continuación se describen los resultados obtenidos en la implementación de los módulos mencionados en el capítulo anterior. Cabe destacar que estos resultados fueron fundamentales para la selección e integración del “mejor modelo” en la aplicación web desarrollada. La sección 4.1 describe los resultados obtenidos al implementar las estrategias de sobremuestreo y submuestreo con la técnica SMOGN en la predicción de la estimación de conjuntos de datos públicos. La sección 4.2, muestra los conjuntos de datos sintéticos de proyectos de software y *sprints*, obtenidos aplicando la metodología GEM. Además, la sección 4.3 describe los resultados obtenidos de los modelos seleccionados para la estimación con proyectos de software. Por otro lado, la sección 4.4 presenta los resultados obtenidos de la selección y evaluación de los modelos seleccionados con *sprints*. Finalmente, la sección 4.5 presenta los prototipos de la aplicación web implementada.

4.1. Pruebas de sobremuestreo y submuestreo para la estimación

Los problemas de distribución de datos surgen en distintas situaciones del mundo real, principalmente para tareas de clasificación; no obstante, el mismo problema se presenta en las tareas de regresión, donde los valores de la variable objetivo son continuos (Avelino et al., 2024). En el caso específico de la estimación del esfuerzo en el desarrollo de software, esta situación se agrava debido a que los conjuntos de datos públicos disponibles suelen ser pequeños y desbalanceados. Estas dos condiciones generan un escenario complejo: por un lado, el tamaño reducido limita la capacidad de los modelos para aprender patrones generales, mientras que, por otro, el desbalanceo de la variable objetivo sesga las predicciones hacia los rangos más frecuentes, dejando sin cobertura adecuada a los valores menos representados.

Este doble desafío repercute directamente en la eficacia y precisión de cada modelo entrenado, ya que los regresores tienden a mostrar un alto error en los casos atípicos o poco frecuentes, lo cual disminuye la confiabilidad de las estimaciones en contextos ágiles. Para aliviar parte de la problemática anterior, se pueden combinar técnicas de sobremuestreo y submuestreo sobre un conjunto de datos. El sobremuestreo permite generar instancias adicionales en las regiones poco representadas, aumentando la diversidad de los datos sin necesidad de recolectar nuevas muestras, mientras que el submuestreo elimina ejemplos de las clases más frecuentes para reducir el sesgo del modelo. Estas técnicas han mostrado un impacto positivo al ser aplicadas en conjuntos pequeños y medianos, mejorando el rendimiento de los regresores (Bhattacharyya et al., 2024; Rad et al., 2025). Primeramente, se definió la metodología a seguir para poder implementar un método de estimación, ilustrada en la Figura 4.1, donde inicialmente se realizó una revisión de la bibliografía de las técnicas de sobremuestreo para regresión, dichas técnicas están descritas en la sección 2.3.3. De ellas se

seleccionó el método SMOGN, debido a que genera un conjunto de muestras diversificado, en comparación con SMOTER (Rad et al., 2025).

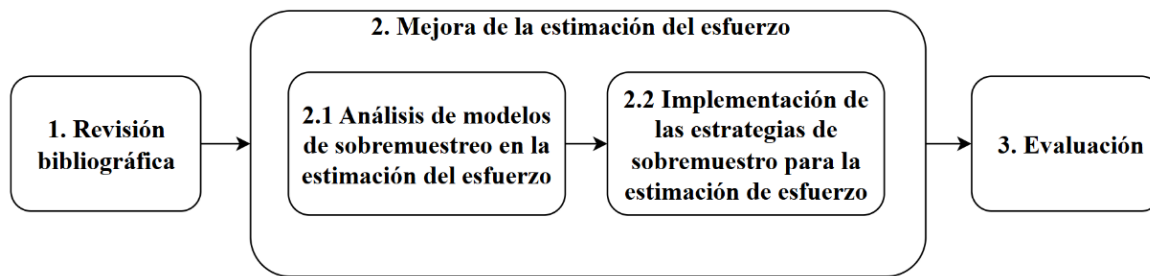


Figura 4.1. Metodología utilizada para implementar el método de estimación.

Posteriormente, se implementaron SMOGN y los métodos para la estimación. Por último, se realizó la evaluación de los métodos empleados. La Figura 4.2 muestra el flujo de trabajo que se siguió, tomando en consideración distintos porcentajes de sobremuestreo y submuestreo, así como los regresores a utilizar y los conjuntos de datos empleados. Primero se seleccionaron los conjuntos de datos: China con 499 muestras considerado como de tamaño mediano, y como conjuntos pequeños, Desharnais con 81 muestras de proyectos de software, COCOMO81 con 63 muestras, y Maxwell con 62 muestras. Más tarde, se realizaron diversas combinaciones de experimentos para cada conjunto de datos, empleando un porcentaje de 200%, 150% y 100% para sobremuestreo, y 50%, 40%, 30% y 20% para submuestreo, obteniendo un total de 12 combinaciones. Consecutivamente se implementó el método de estimación y, finalmente, se calcularon mediciones utilizando MAE, RMSE, MMRE y PRED, con el fin de establecer una base comparativa con las métricas más utilizadas en la literatura.

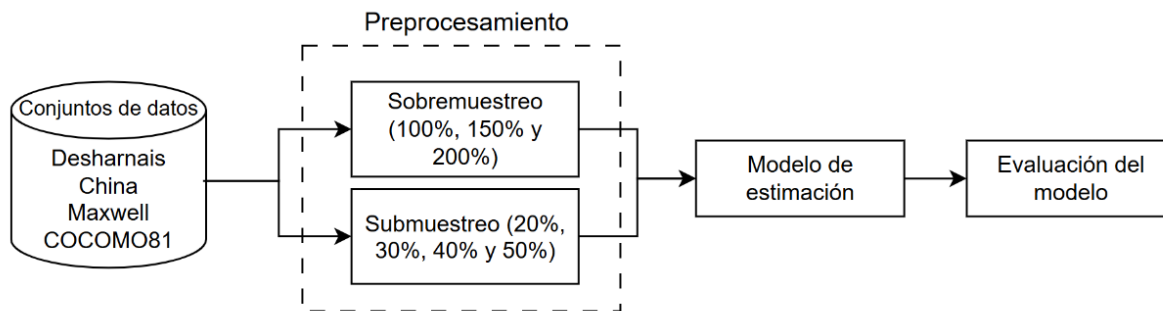


Figura 4.2. Flujo de trabajo aplicando técnicas de sobremuestreo.

Para implementar el modelo de estimación, se utilizaron siete modelos individuales (i.e., ELM, SVR, DT, RF, LR, Lasso y Ridge), y siete modelos de ensamble construidos mediante la combinación de algunos de los modelos individuales, como se muestra en la Tabla 22. Debido a que la distribución de la variable dependiendo del conjunto de datos de Desharnais se encuentra en un rango de 0 a 15,000 horas-persona (véase la Figura 4.3), fue necesario aplicar SMOGN, para permitir una mejor variabilidad de las muestras, y, en consecuencia, obtener una predicción adecuada de los modelos de estimación. Una vez aplicado SMOGN, se observó una mejor distribución de los datos como se muestra en la Figura 4.3 con color verde.

Debido a que los conjuntos seleccionados cuentan con pocas muestras, a excepción de China, se utilizó un particionamiento de 80% para entrenamiento y 20% para pruebas y para China se aplicó un particionamiento de 70% para entrenamiento y 30% para pruebas. Una vez realizado el

preprocesamiento, se procedió a evaluar los diversos modelos, aplicando las combinaciones de porcentajes de sobremuestreo y submuestreo. Además, se realizó la configuración de cada regresor utilizado mediante la técnica de *Grid Search* para búsqueda de hiperparámetros, que posteriormente se mejoró y acotó a valores más cercanos de la mejor combinación encontrada, a través de una búsqueda fina. En el caso de LR, no se llevó a cabo la búsqueda de hiperparámetros, debido a que se emplearon los parámetros determinados por default.

Tabla 22. Ensamblajes utilizados.

Ensamble	Modelos involucrados
Ensamble 1 (E1)	ELM + SVR, Metamodelo RF
Ensamble 2 (E2)	ELM + SVR + DT + RF + LR, Metamodelo SVR
Ensamble 3 (E3)	ELM + SVR + DT + RF + LR, Metamodelo RF
Ensamble 4 (E4)	ELM + SVR + LR, Metamodelo RF
Ensamble 5 (E5)	RF + RF + RF, Metamodelo RF
Ensamble 6 (E6)	ELM + SVR + Lasso, Metamodelo RF
Ensamble 7 (E7)	ELM + SVR + Ridge, Metamodelo RF

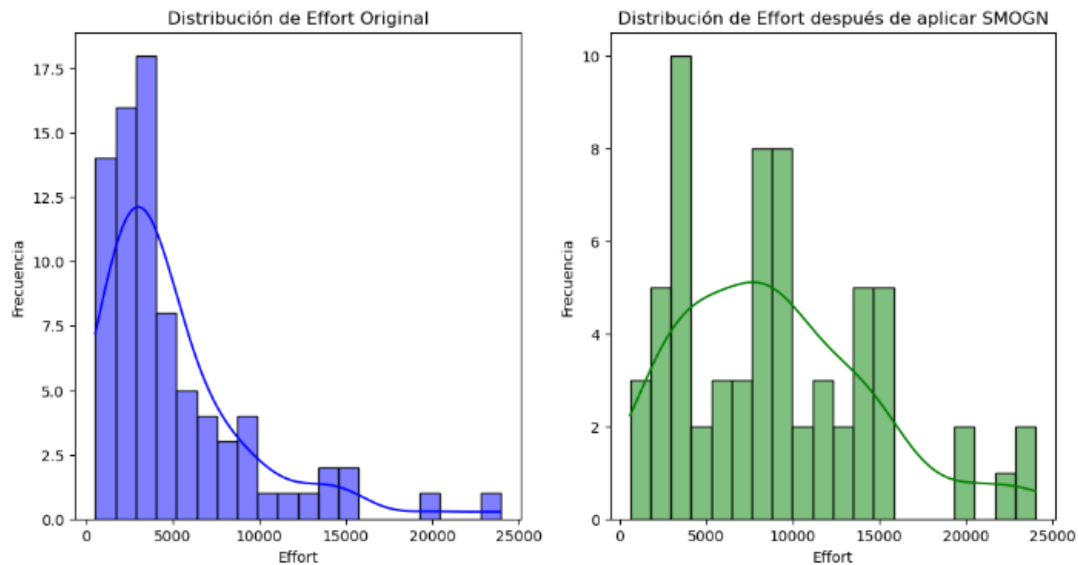


Figura 4.3. Distribución de Desharnais.

Ahora bien, en la Tabla 23 se indican los intervalos de búsqueda de cada modelo utilizado en el *Grid Search*. Una vez realizada la búsqueda de hiperparámetros, se utilizaron los valores que proporcionaron los mejores resultados en cada modelo individual y de ensambles. Finalmente, se evaluaron todos los regresores utilizando las métricas de evaluación MAE, RMSE, MMRE y PRED, asegurando una evaluación confiable del rendimiento y la generalización de los modelos. Una vez realizada la búsqueda de hiperparámetros, aplicando *Grid Search* y búsqueda fina a los intervalos presentados en la tabla anterior, se obtuvieron los valores con la mejor configuración, que obtienen un mínimo error o máximo PRED (véase la Tabla 24).

Tabla 23. Intervalos de búsqueda de hiperparámetros.

Modelo	Parámetros y valores
ELM	n_hidden: [50, 100, 150, 200, 300] activation: ['relu', 'tanh', 'sigmoid']
SVR	C: [1, 10, 100] Kernel: ['rbf', 'linear'] Gamma: ['scale', 'auto', 0.01, 0.001] Épsilon: [0.01, 0.1, 1]
DT	max_depth: None, D, [D + 5, D + 10, D + 20] i.e. si D = 6 $\rightarrow$ [6, 11, 16, 26] min_samples_split: [1, 2, 3, 4, 5]
RF	n_estimators: [10, 50, 100, 150, 200] max_depth: None = D, [D + 5, D + 10, D + 20] min_samples_split: [2, 5, 10]
Lasso	alpha: [0.0001, 0.001, 0.01, 0.1, 1, 10] max_iter: [10, 50, 100, 200]
Ridge	alpha: [0.0001, 0.001, 0.01, 0.1, 1, 10] max_iter: [10, 50, 100, 200]

Tabla 24. Hiperparámetros encontrados con la búsqueda fina.

Modelo	Maxwell	COCOMO	Desharnais	China
ELM	n_hidden: 100 activation: 'relu'	n_hidden: 150 activation: 'tanh'	n_hidden: 100 activation: 'sigmoid'	n_hidden: 50 activation: 'relu'
SVR	C: 0.5 Kernel: 'linear' Gamma: 'scale'	C: 100 Kernel: 'linear' Gamma: 'scale'	C: 20 Kernel: 'linear' Gamma: 'scale'	C: 1 Kernel: 'linear' Gamma: 'scale'
DT	max_depth: 6 min_samples_split: 2	max_depth: 11 min_samples_split: 3	max_depth: 8 min_samples_split: 2	max_depth: 11 min_samples_split: 3
RF	n_estimators: 250 max_depth: 9 min_samples_split: 3	n_estimators: 187 max_depth: 9 min_samples_split: 11	n_estimators: 250 max_depth: 8 min_samples_split: 2	n_estimators: 75 max_depth: 21 min_samples_split: 3
Lasso	alpha: 15 max_iter: 202	alpha: 15 max_iter: 52	alpha: 15 max_iter: 48	alpha: 15
Ridge	alpha: 15 max_iter: 8	alpha: 15 max_iter: 8	alpha: 15 max_iter: 48	alpha: 0.15

Utilizando la configuración para cada modelo de la anterior tabla y siguiendo el flujo de trabajo presentado en la Figura 4.2, se aplicaron estrategias de combinación de porcentajes de sobremuestreo

y submuestreo a los conjuntos de datos públicos seleccionados, integrando los siete modelos individuales y los siete modelos de ensamble, y evaluando cada modelo con la medida de evaluación RMSE. Por ejemplo, para el conjunto de datos de Desharnais, se obtuvieron los resultados presentados en la Tabla 25. En esta se pueden visualizar, en la segunda columna, los resultados expuestos por Durgesh et al. (2023), mientras que un guion indica que los autores no presentaron resultados de dichos modelos. Por otro lado, el 100%, 150% y 200% corresponden al porcentaje de sobremuestreo y el 20%, 30%, 40% y 50% corresponden al porcentaje de sobremuestreo utilizado. Asimismo, de la columna 3 a 14, se muestran los resultados de las combinaciones de sobremuestreo y submuestreo respectivamente. Los resultados resaltados en negritas representan el error mínimo obtenido de cada combinación, (i. e. resultado mínimo de cada columna), mientras que los resultados subrayados indican valores inferiores al valor mínimo expuesto por Durgesh et al. (2023). Por ejemplo, en la última columna, se muestran los resultados al utilizar un porcentaje de 200% para sobremuestreo y 20% de submuestreo, y en la columna 10, se muestran los resultados obtenidos al utilizar 150% de sobremuestreo y 20% de submuestreo.

Tabla 25. Resultados de RMSE sobre el conjunto Desharnais.

Modelo	Durgesh et al. (2023)	100				150				200			
		50	40	30	20	50	40	30	20	50	40	30	20
ELM	2846.64	3749.178	4106.164	1776.638	2886.252	<u>1747.227</u>	3767.687	<u>2231.986</u>	3749.178	2951.384	<u>2171.618</u>	<u>2332.914</u>	<u>1075.651</u>
SVR	3104.94	2613.889	6215.155	5468.670	3176.622	2866.704	<u>2137.975</u>	3402.516	2613.889	3257.932	3637.812	2835.936	<u>2276.922</u>
E1	2360.51	2487.138	4555.597	<u>1872.047</u>	2843.996	<u>1887.943</u>	3144.335	2425.891	2487.138	2579.646	2500.325	2708.159	552.118
DT	-	<u>2234.670</u>	4027.112	<u>2291.595</u>	<u>2153.035</u>	<u>1821.796</u>	876.168	4557.225	<u>2234.670</u>	<u>1926.578</u>	5181.461	3680.258	<u>764.027</u>
RF	-	1758.043	3038.255	<u>2086.285</u>	2582.859	<u>1806.617</u>	<u>1193.891</u>	2541.857	1758.043	<u>1746.124</u>	<u>2101.480</u>	<u>2280.807</u>	<u>1954.596</u>
LR	-	<u>2162.458</u>	6711.815	4229.866	<u>1733.133</u>	<u>1847.930</u>	3501.960	<u>2314.658</u>	<u>2162.458</u>	<u>2065.256</u>	3018.417	<u>2004.753</u>	<u>1007.971</u>
E2	-	2543.687	2731.890	2632.282	<u>1362.099</u>	<u>1664.622</u>	<u>1092.424</u>	1855.407	2543.687	1646.940	<u>1691.375</u>	1790.951	3118.431
E3	-	2081.205	2784.843	<u>1807.579</u>	2527.973	1564.396	<u>1406.025</u>	2870.148	<u>2081.205</u>	<u>1911.715</u>	1605.262	2702.532	<u>680.496</u>
E4	-	2390.023	4863.757	<u>2114.144</u>	2748.120	<u>1843.343</u>	3018.605	2402.198	2390.023	2397.850	2490.104	2602.652	<u>620.768</u>
E5	-	<u>1846.759</u>	2582.376	2594.789	2750.923	<u>2100.001</u>	<u>1420.908</u>	2388.627	<u>1846.759</u>	<u>1879.233</u>	<u>1946.159</u>	<u>2173.271</u>	<u>1749.702</u>
Lasso	-	<u>1901.349</u>	4557.139	3497.851	1213.073	<u>1703.147</u>	3189.451	<u>2120.623</u>	<u>1901.349</u>	<u>2078.641</u>	3007.605	2385.797	<u>1017.381</u>
Ridge	-	<u>1820.292</u>	6849.999	5002.692	2367.580	<u>1902.439</u>	3481.000	<u>2233.743</u>	<u>1820.292</u>	<u>2148.145</u>	3198.876	<u>2179.083</u>	<u>1388.292</u>
E6	-	2406.862	4392.982	<u>2139.815</u>	<u>2271.333</u>	<u>1846.279</u>	3019.035	2343.553	2406.862	2404.327	2550.375	2576.631	<u>617.636</u>
E7	-	2373.880	4878.113	<u>1993.932</u>	2793.784	<u>1835.609</u>	3045.705	2449.931	2373.880	2423.660	2489.664	2638.556	<u>581.744</u>

Como se podrá observar, mientras mayor sea el porcentaje de sobremuestreo se obtienen mejores resultados en comparación, a los presentados por Durgesh et al. (2023). A continuación, se presentan los resultados, comparando los valores obtenidos y los valores de Durgesh et al. (2023) y Jawa y Meena (2022). En la Tabla 26, se puede observar claramente el impacto de las estrategias de sobremuestreo y submuestreo en la estimación del esfuerzo. Los modelos individuales y de ensamble evaluados mostraron diferencias significativas en su rendimiento, siendo los modelos de ensamble los

que lograron una mayor reducción de los errores de predicción. Por ejemplo, para una combinación del 150% de sobremuestro y 40% de submuestreo, aplicado al conjunto de datos de COCOMO, se obtuvo para la medida MMRE un resultado muy por debajo de lo presentado por los autores; mientras que para la medida PRED se obtuvo un mejor resultado. El análisis de las métricas MAE, RMSE, MMRE y PRED evidencia que la combinación de modelos con estrategias de ajuste de datos permite mejorar la precisión y la capacidad de generalización, especialmente en conjuntos de datos pequeños y desbalanceados. Por lo tanto, se concluye que la implementación de un modelo de ensamble adecuado, junto con las estrategias de manejo de desbalance, incrementa la precisión de la predicción del esfuerzo, contribuyendo a una planificación de proyectos más efectiva y a la toma de decisiones más fundamentadas en entornos de desarrollo de software. Para obtener información más detallada al respecto, se puede consultar el estudio presentado en (Martínez et al., 2025), el cual fue publicado en la *Revista de Investigación en Tecnologías de la Información* y puede consultarse a través del siguiente enlace: <https://riti.es/index.php/riti/article/view/395>.

Tabla 26. Comparativa de métricas por cada conjunto de datos.

Mejor combinación	Conjunto de datos	Métricas de evaluación	Modelo	Resultado	Métricas de evaluación	Modelo	Resultado
150% - 50%	China	MMRE	RF (Jawa & Meena, 2022)	0.06	PRED	RF (Jawa & Meena, 2022)	98.6
			<u>SVR</u>	<u>0.06</u>		<u>RF</u>	<u>98.7</u>
150% - 40%	COCOMO		DT (Jawa & Meena, 2022)	1.500		DT (Jawa & Meena, 2022)	5.2
			<u>E6</u>	<u>0.368</u>		<u>E6</u>	<u>66.6</u>
150% - 30%	Maxwell		RF (Jawa & Meena, 2022)	0.490		RF (Jawa & Meena, 2022)	66.6
			<u>E4</u>	<u>0.304</u>		<u>E4</u>	<u>75.0</u>
200% - 20%	Desharnais	MAE	E1 (Durgesh et al., 2023)	1412.25	RMSE	E1 (Durgesh et al., 2023)	2360.51
			<u>E1</u>	<u>511.51</u>		<u>E1</u>	<u>552.11</u>

4.2. Creación de conjuntos de datos sintéticos

La disponibilidad de conjuntos de datos constituye un elemento fundamental para el desarrollo y evaluación de modelos de aprendizaje computacional orientados a la estimación del esfuerzo en proyectos de software. Sin embargo, los conjuntos de datos públicos existentes, como Desharnais, Albrecht, China o ISBSG, fueron desarrollados a partir de proyectos de organizaciones específicas y,

en la mayoría de los casos, no consideran las características propias de pequeñas empresas que emplean Scrum, ni incorporan variables como los factores vitales del proyecto, los niveles de incertidumbre, los puntos de historia y la velocidad del equipo. Esta limitación dificulta el entrenamiento de modelos capaces de representar escenarios reales en este contexto.

Por lo descrito anteriormente, fue necesario crear conjuntos de datos sintéticos que permitieran representar diferentes escenarios de proyectos y *sprints* mediante la combinación sistemática de las variables definidas en esta investigación. Estos conjuntos de datos proporcionan la base empírica necesaria para realizar estimaciones precisas del esfuerzo en proyectos Scrum. A partir de la combinación de factores vitales, niveles de incertidumbre, puntos de historia y velocidad del equipo, y mediante la aplicación sistemática de las ecuaciones del método GEM (Bhalerao e Ingle, 2011), fue posible generar registros representativos de distintos contextos de desarrollo.

4.2.1. Conjunto de datos sintéticos obtenido para la estimación de proyectos de software

El método GEM calcula el esfuerzo mediante la combinación de factores que influyen en el desarrollo de un proyecto. Su funcionamiento consiste en asignar valores a los factores vitales y a los niveles de incertidumbre para integrarlos con métricas ágiles, como los puntos de historia y la velocidad del equipo, mediante un conjunto de ecuaciones. Como resultado, se obtiene una estimación del esfuerzo que representa las características y condiciones particulares de cada escenario de desarrollo.

A partir de este procedimiento, descrito en la sección 3.2, se generó un conjunto de datos sintéticos conformado por diferentes combinaciones de factores vitales, niveles de incertidumbre, puntos de historia y velocidad del equipo. Cada registro representa un escenario potencial de desarrollo de software y su correspondiente estimación de esfuerzo calculada mediante el método GEM, proporcionando una base para el entrenamiento y evaluación de los modelos propuestos. Para visualizar de manera clara y detallada el uso de cada factor, a continuación, se describe un ejemplo con los pasos a seguir para la obtención de una muestra que representa un proyecto de software del conjunto de datos obtenido. Cabe recordar que las siglas utilizadas en el ejemplo fueron definidas en la sección 3.2.

Ejemplo:

1. Se asignan niveles de Intensidad (1, 4 o 9) a cada uno de los 7 factores, por ejemplo: Rendimiento: 4, Complejidad: 4, Configuración: 4, Datos: 9, Seguridad: 9, Múltiple: 1 y Operación: 1.
2. Se definen los pesos para cada factor vital, sin que la suma de ellos sea mayor que 1, Rendimiento: 0.2, Complejidad: 0.2, Configuración: 0.2, Datos: 0.15, Seguridad: 0.15, Múltiple: 0.05 y Operación: 0.05.
3. Se calcula el PF multiplicando el peso por el nivel de intensidad, por lo que se obtendría: Rendimiento: 0.8 (4×0.2), Complejidad: 0.8, Configuración: 0.8, Datos: 1.35, Seguridad: 1.35, Múltiple: 0.05 y Operación: 0.05.
4. Se calcula el UV, sumando los factores de prioridad de cada factor, por lo que se obtiene $0.8 + 0.8 + 0.8 + 1.35 + 1.35 + 0.05 + 0.05 = 5.2$.
5. Después, se calculan los $NSP = SP + PD + UF * UV$, teniendo un PD de 4, SP de 7 y UF de 0.8, por lo tanto, se obtiene $7 + 4 + 0.8 * 5.2 = 15.16$.

6. Se obtiene el tamaño del proyecto sumando todos los NSP, de cada historia de usuario. Si son 25 historias, y cada una tiene 15.16 NSP, entonces se obtendría 379 NSP.
7. Finalmente, la duración del proyecto se obtiene al dividir el tamaño del proyecto sobre la velocidad del equipo. Por lo tanto, dado que el tamaño del proyecto es 379 NSP y la velocidad del equipo es 8 SP/persona, entonces se obtendría una duración de 47.375 semanas.

Este ejemplo solamente representa una muestra del conjunto de datos obtenido, por lo que, para poder generar nuevas muestras, es necesario utilizar diferentes valores para niveles de intensidad, pesos para cada factor, PD, SP, UF y velocidad del equipo.

Para ampliar la aplicabilidad del modelo se ha incrementado el rango de valores para el factor de incertidumbre de 0.1 a 1, con intervalos de 0.1, permitiendo modelar distintos niveles de riesgo en los proyectos, a diferencia de la escala propuesta por Bahlerao e Ingle (2011) que va de 0.2 a 0.8, con intervalos de 0.2. El conjunto de datos obtenido contiene un total de 480 registros sintéticos y 15 atributos. La Tabla 27 muestra la descripción de las variables y valores específicos utilizados para la obtención del conjunto de datos sintéticos. El conjunto de datos obtenido se encuentra disponible en el siguiente enlace [GEM.csv](#).

Tabla 27. Variables del conjunto de datos sintéticos obtenido para proyectos de software.

Variables	Descripción	Valores
UF	Factor de incertidumbre del proyecto, para un nivel bajo (0.1) cuando se tiene un 10% de incertidumbre y nivel alto (1) cuando se tiene uno 100% de incertidumbre.	0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9 y 1.
v1, v2, v3, v4, v5, v6 y v7	Factores vitales involucrados en el proyecto. v1: Rendimiento, v2: Configuración, v3: Complejidad, v4: Transferencia de datos, v5: Seguridad, v6: Múltiples sitios, v7: Facilidad de operación.	Nivel bajo (1), medio (4) y alto (9).
UV	Valor no ajustado	Obtenido con la ecuación 7.
HU	Historia de usuario	25 por cada proyecto.
SP	Puntos por historia de usuario	7 por cada historia de usuario.
PD	Dominio del proyecto	AW (1), MIS (2) y M (3).
Velocidad del equipo	Velocidad del equipo del proyecto	8 puntos de historia por persona.
NSP	Nuevos puntos de historia	Obtenido con la ecuación 8.
SOP	Tamaño del proyecto	Obtenido con la ecuación 9.
DOP	Duración del proyecto	Obtenido con la ecuación 10.

A continuación, la Figura 4.4 presenta algunas muestras del conjunto de datos obtenido, aplicando la metodología descrita en la sección 3.2.

Este conjunto de datos constituyó la base inicial de la presente tesis. No obstante, con el objetivo de analizar a mayor detalle los factores de los proyectos de software bajo la metodología Scrum, se construyó un segundo conjunto de datos que incorpora factores de dos pequeñas

organizaciones desarrolladoras de software. De esta manera, se buscó enriquecer la diversidad de las muestras. La calidad y precisión del conjunto de datos ayudó a implementar una herramienta eficaz de estimación de esfuerzo, permitiendo una gestión más eficiente de éste y de los recursos en los proyectos.

No.	UF	v1	v2	v3	v4	v5	v6	v7	UV	SP	PD	velocidad	NSP	SOP	DOP	
1	0.1	1	1	1	1	1	1	1	1	1	7	1	8	8.1	202.5	25.31
2	0.1	4	4	4	4	4	4	4	4	4	7	1	8	8.4	210	26.25
3	0.1	9	9	9	9	9	9	9	9	9	7	1	8	8.9	222.5	27.81
4	0.2	1	1	1	1	1	1	1	1	1	7	1	8	8.2	205	25.62
5	0.2	4	4	4	4	4	4	4	4	4	7	1	8	8.8	220	27.5
6	0.2	9	9	9	9	9	9	9	9	9	7	1	8	9.8	245	30.63
7	0.3	1	1	1	1	1	1	1	1	1	7	1	8	8.3	207.5	25.94
8	0.3	4	4	4	4	4	4	4	4	4	7	1	8	9.2	230	28.75
9	0.3	9	9	9	9	9	9	9	9	9	7	1	8	10.7	267.5	33.44
10	0.4	1	1	1	1	1	1	1	1	1	7	1	8	8.4	210	26.25
11	0.4	4	4	4	4	4	4	4	4	4	7	1	8	9.6	240	30
12	0.4	9	9	9	9	9	9	9	9	9	7	1	8	11.6	290	36.25
13	0.5	1	1	1	1	1	1	1	1	1	7	1	8	8.5	212.5	26.56
14	0.5	4	4	4	4	4	4	4	4	4	7	1	8	10	250	31.25
15	0.5	9	9	9	9	9	9	9	9	9	7	1	8	12.5	312.5	39.06
16	0.6	1	1	1	1	1	1	1	1	1	7	1	8	8.6	215	26.87
17	0.6	4	4	4	4	4	4	4	4	4	7	1	8	10.4	260	32.5
18	0.6	9	9	9	9	9	9	9	9	9	7	1	8	13.4	335	41.87
19	0.7	1	1	1	1	1	1	1	1	1	7	1	8	8.7	217.5	27.19
20	0.7	4	4	4	4	4	4	4	4	4	7	1	8	10.8	270	33.75
21	0.7	9	9	9	9	9	9	9	9	9	7	1	8	14.3	357.5	44.69
22	0.8	1	1	1	1	1	1	1	1	1	7	1	8	8.8	220	27.5
23	0.8	4	4	4	4	4	4	4	4	4	7	1	8	11.2	280	35
24	0.8	9	9	9	9	9	9	9	9	9	7	1	8	15.2	380	47.5
25	0.9	1	1	1	1	1	1	1	1	1	7	1	8	8.9	222.5	27.81
26	0.9	4	4	4	4	4	4	4	4	4	7	1	8	11.6	290	36.25
27	0.9	9	9	9	9	9	9	9	9	9	7	1	8	16.1	402.5	50.31

Figura 4.4. Conjunto de datos sintéticos obtenido para proyectos de software.

De acuerdo con lo establecido anteriormente, la investigación siguió un enfoque mixto, por lo que los datos fueron recolectados utilizando tanto herramientas cualitativas como cuantitativas. Considerando un enfoque cualitativo, se diseñó un cuestionario de evaluación siguiendo las recomendaciones de Engström y Runeson (2010) y Hynninen y Jantunen (2022) con el fin de determinar si, de acuerdo con las percepciones de las y los participantes del grupo experimental, la estrategia propuesta contribuía a mejorar el proceso relacionado con la gestión de las pruebas de regresión (véase Tabla 34). Dicho cuestionario busca evaluar la opinión de estas y estos participantes sobre la efectividad de las pruebas de regresión a través de preguntas que se dividen en cuatro categorías: la automatización de la estrategia, la cobertura de las pruebas, la detección de los fallos, y el tiempo de ejecución. Las preguntas se relacionan directamente con la estrategia propuesta y las métricas clave para asegurar que los cambios en el código no afecten negativamente la funcionalidad existente del software. Dicho instrumento se basó en una escala Likert de 5 puntos con las siguientes alternativas de respuesta: “Totalmente de acuerdo” (TDA=2), “De acuerdo” (DA=1), “Neutral” (N=0), “En desacuerdo” (ED=-1) y “Totalmente en desacuerdo” (TED=-2), que se convirtieron a un valor numérico para obtener una medida cuantitativa. Aunado a esto, se calcularon la mediana (M) y la desviación estándar (DE) de las respuestas proporcionadas por todas y todos los participantes en la evaluación.

4.2.2. Conjunto de datos sintéticos obtenido para la estimación de *sprints*

Para la construcción del conjunto de datos sintéticos para *sprints*, se utilizó el mismo el método GEM, pero tomando como referencia los datos propuestos por Ramessur y Nagowah (2021), a excepción de la cantidad de factores vitales y factores de incertidumbre los cuales fueron definidos por dos pequeñas organizaciones desarrolladoras de software, dando como resultado la terminología descrita en la Tabla 28.

En este sentido, el conjunto de datos obtenido permitió integrar factores vitales, que las pequeñas organizaciones toman en cuenta al momento de desarrollar algún proyecto, puesto que,

debido a su tamaño, capacidad y experiencia, se imposibilita la obtención clara de la estimación del esfuerzo para sus proyectos de software. Una vez integradas las variables del algoritmo GEM junto con los escenarios propuestos por Ramessur y Nagowah (2021), se propició la generación de datos relevantes que reflejan la complejidad, las capacidades y los riesgos asociados a cada proyecto de software.

Tabla 28. Terminología de GEM. Nota: Traducido de Ramessur y Nagowah (2021).

Términos utilizados	Valores definidos por los autores	Valores definidos por las organizaciones
Dominio del proyecto (<i>PD</i>)	Aplicación web (AW=1), Sistemas de información (MIS=4) y Proyecto militar (M=9).	Aplicación web (AW=1), Sistemas de información (MIS=4) y Proyecto militar (M=9).
Factores vitales (<i>v</i>)	v1: Partes interesadas, v2: Dinámica y compromiso del equipo, v3: Comunicación, v4: Experiencia y capacidad técnica del personal, v5: Experiencia en proyectos anteriores, v6: Responsabilidades fuera del proyecto, v7: Configuración, v8: Seguridad, v9: Defectos y cambios de implementaciones anteriores, v10: Calidad de los requisitos, v11: Complejidad de los requisitos y v12: Volatilidad de los requisitos.	v1: Partes interesadas, v2: Experiencia técnica, v3: Calidad de las historias, v4: Experiencia previa, v5: Comunicación, v6: Capacidad del equipo, v7: Configuraciones, v8: Cambio de requisitos, v9: Defectos y cambios previos, v10: Limitaciones técnicas, v11: Compromiso del equipo y v12: Seguridad, v13: Responsabilidades fuera del proyecto y v14: Rotación del personal.
Pesos (<i>w</i>)	Escenario 1: (v1, v2, v3, v5, v6, v7, v8, v9, v10, v12 $w = 0.08$) y (v4, v11, $w = 0.1$). Escenario 2: (v1, v2, v3, v5, v6, v7, v8, v9, v10, v12 $w = 0.06$) y (v4, v11, $w = 0.2$). Escenario 3: (v1, v2, v3, v5, v6, v7, v8, v9, v10, v12 $w = 0.04$) y (v4, v11, $w = 0.3$). Escenario 4: (v1, v2, v3, v5, v6, v7, v8, v9, v10, v12 $w = 0.02$) y (v4, v11, $w = 0.4$).	Escenario 1: (v1, v2, v3, v5, v6, v7, v8, v9, v10, v12 $w = 0.08$) y (v4, v11, $w = 0.1$). Escenario 2: (v1, v2, v3, v5, v6, v7, v8, v9, v10, v12 $w = 0.06$) y (v4, v11, $w = 0.2$). Escenario 3: (v1, v2, v3, v5, v6, v7, v8, v9, v10, v12 $w = 0.04$) y (v4, v11, $w = 0.3$). Escenario 4: (v1, v2, v3, v5, v6, v7, v8, v9, v10, v12 $w = 0.02$) y (v4, v11, $w = 0.4$).
Nivel de intensidad (<i>I</i>)	Bajo (L), medio (M) y alto (H).	Bajo (L), medio (M) y alto (H).
Factor de incertidumbre (<i>UF</i>)	0.2, 0.4, 0.6 y 0.8.	0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8 y 0.9.
HU	5, 10 o 15 por cada <i>sprint</i> .	5, 10 o 15 por cada <i>sprint</i> .
Puntos de historia (<i>SP</i>)	7 por historia de usuario.	7 por historia de usuario.
Factor de Prioridad (<i>PF</i>)	Se obtiene usando la Ecuación (6).	Se obtiene usando la ecuación 6.
Valor No Ajustado (<i>UV</i>)	Se obtiene usando la ecuación 7.	Se obtiene usando la ecuación 7.
Nuevos puntos de historia (<i>NSP</i>)	Se obtiene usando la ecuación 8.	Se obtiene usando la ecuación 8.
Velocidad del equipo	8.	8.

Esta combinación permitió representar diferentes contextos de desarrollo y proporcionar información valiosa para apoyar la toma oportuna de decisiones durante la ejecución del proyecto, contribuyendo a minimizar desviaciones en el cronograma y en la asignación de recursos durante cada *sprint*. Al seguir la metodología descrita anteriormente se obtuvo un conjunto de datos de 576 muestras, pero al aplicar los valores de los pesos para cada escenario, se obtuvo un conjunto de datos con 2,304 muestras.

El conjunto de datos obtenido, se encuentra disponible en el siguiente enlace [GEM Sprint](#). La Figura 4.5 muestra algunos registros del conjunto de datos obtenido, el cual está conformado por 21 características. De estas, 20 corresponden a variables predictoras y una a la variable objetivo denominada NSP. No se considera la característica No., ya que únicamente se utiliza para identificar y enumerar cada muestra.

No.	UF	v1	v2	v3	v4	v5	v6	v7	v8	v9	v10	v11	v12	v13	v14	UV	HU	SP	PD	velocidad	NSP
1	0.1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	5	7	1	8
2	0.1	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	5	7	1	8
3	0.1	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	5	7	1	8
4	0.2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	5	7	1	8
5	0.2	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	5	7	1	8
6	0.2	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	5	7	1	8
7	0.3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	5	7	1	8
8	0.3	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	5	7	1	8
9	0.3	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	5	7	1	8
10	0.4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	5	7	1	8
11	0.4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	5	7	1	8
12	0.4	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	5	7	1	8
13	0.5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	5	7	1	8
14	0.5	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	5	7	1	8
15	0.5	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	5	7	1	8
16	0.6	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	5	7	1	8
17	0.6	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	5	7	1	8
18	0.6	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	5	7	1	8
19	0.7	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	5	7	1	8
20	0.7	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	5	7	1	8
21	0.7	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	5	7	1	8
22	0.8	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	5	7	1	8
23	0.8	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	5	7	1	8
24	0.8	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	5	7	1	8
25	0.9	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	5	7	1	8
26	0.9	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	5	7	1	8
27	0.9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	5	7	1	8
28	0.1	1	1	1	9	9	1	1	1	1	1	9	9	1	1	3.24	5	7	1	8	8
29	0.1	4	4	4	9	9	1	1	4	4	4	9	9	1	1	4.68	5	7	1	8	8

Figura 4.5. Conjunto de datos sintéticos obtenido para *sprints*.

Con base en lo anterior, se dispone de un conjunto de datos estructurado, consistente y representativo de diversos escenarios en proyectos desarrollados bajo el marco de trabajo Scrum, lo que proporciona una base adecuada para la aplicación y evaluación de técnicas de aprendizaje computacional orientadas a la estimación. Ahora bien, la distribución de las muestras en el conjunto de datos, de acuerdo con el dominio del proyecto, sería de 50% por PD. Por ejemplo para un PD=1, serían 1,152 muestras correspondientes.

4.3. Resultados de estimación con proyectos de software

En esta sección se presentan los resultados obtenidos al aplicar la metodología descrita en la sección 3.3, en conjunto con las estrategias de sobremuestreo basadas en SMOGN, con el objetivo de seleccionar el “mejor modelo” que sería integrado a la aplicación web implementada. Para el conjunto de datos sintéticos obtenido de proyectos de software, se aplicaron las técnicas de preprocesamiento y búsqueda de hiperparámetros previamente definidas en la metodología. Estas técnicas permitieron mejorar tanto la calidad de los datos, así como el desempeño de los modelos de estimación evaluados.

4.3.1. Preprocesamiento

En la etapa de preprocesamiento, una vez obtenido el conjunto de datos normalizado, se analizó e identificó si al seleccionar las características más relevantes, estas contribuían en el desempeño de los modelos seleccionados. Para ello se empleó la selección de características mediante la matriz de correlación, donde las características con mayor correlación con la variable objetivo fueron seleccionadas, obteniendo así “PD”, “velocidad”, “NSP” y “SOP” como las características con mayor correlación. Al evaluar los modelos entrenados con esta selección de características y comparar los resultados obtenidos, se observó que ninguno presentó mejoras en las métricas de rendimiento.

Por el contrario, todos los modelos registraron un incremento en el MAE al aplicar selección de características, en comparación con los resultados obtenidos al utilizar el conjunto completo de variables. Entre ellos, SVR fue el modelo más afectado, mostrando un aumento considerable en el error, llegando incluso a triplicar su valor respecto al obtenido utilizando todas las características. Posteriormente, se aplicó la técnica de sobremuestreo SMOGN con el objetivo de abordar el desbalance presente en la variable objetivo. Este enfoque permitió generar nuevas instancias sintéticas en aquellas regiones con menor representación, favoreciendo una distribución más equilibrada de los datos. La Figura 4.6 muestra la distribución del conjunto de datos, antes y después del proceso de sobremuestreo, donde se puede observar una mejora, en la representación de valores extremos.

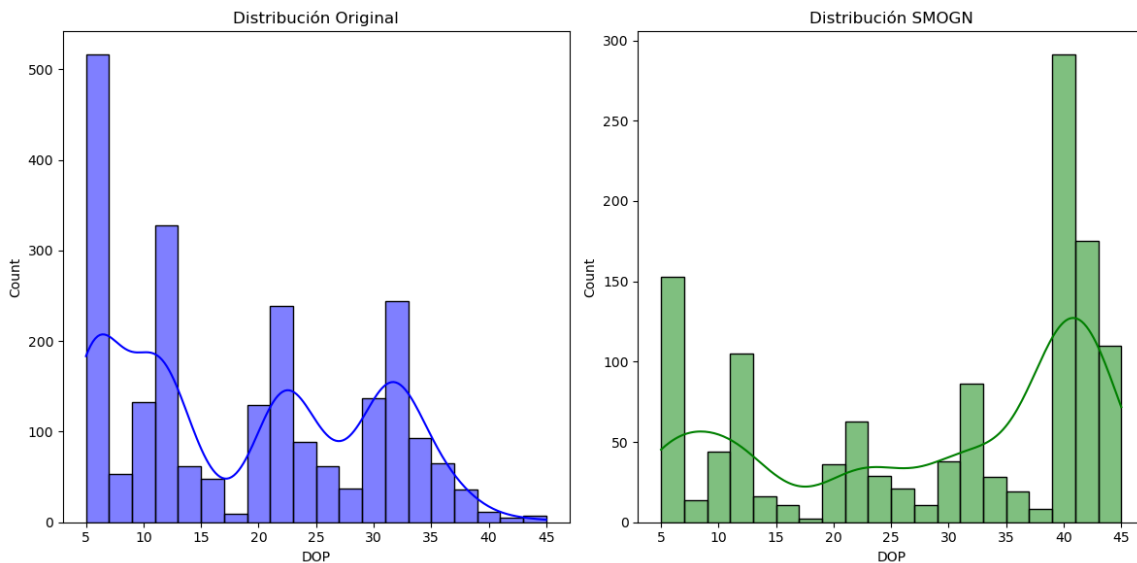


Figura 4.6. Distribución del conjunto de datos para proyectos de software.

Como resultado se obtuvo un conjunto de datos más equilibrado y representativo, lo que permitió establecer una base adecuada para el entrenamiento de los modelos. En consecuencia, se procedió a la etapa de búsqueda de hiperparámetros, en la cual se ajustaron los hiperparámetros de los modelos con el propósito de mejorar su desempeño predictivo y su capacidad de generalización. El conjunto de datos obtenido tendrá un número de muestras de acuerdo con el porcentaje de sobremuestreo y submuestreo aplicado.

4.3.2. Búsqueda de hiperparámetros

Posteriormente, se realizó la búsqueda de hiperparámetros, con el fin de mejorar la capacidad de predicción de los modelos. Para ello, se implementó una estrategia de búsqueda basada en *Grid*

Search, la cual permitió explorar distintas combinaciones de parámetros para cada uno de los nueve modelos utilizados. Los intervalos de búsqueda de hiperparámetros fueron establecidos de acuerdo con las características de cada modelo, considerando valores que permitieran capturar tanto configuraciones simples como complejas.

En síntesis, la Tabla 29 presenta los rangos definidos para los modelos individuales seleccionados, y que serán parte de los modelos de ensamble. Estos datos, son presentados en conjunto para poder delimitar el rango de valores utilizados para la selección de hiperparámetros. Sin embargo, cabe señalar que LR no fue incluido en este proceso, ya que se utilizaron los valores predeterminados de la implementación de *Scikit-learn*, debido a la naturaleza limitada de sus parámetros configurables.

Tabla 29. Intervalos de búsqueda de hiperparámetros.

Modelo	Parámetros y valores
KNN	n_neighbors: [3, 5, 7, 9, 11, 15] weights: ['uniform', 'distance']
MLP	hidden_layer_sizes: [(3,), (5,), (9,), (16,), (100, 50), (100, 100), (150, 100, 50)] activation: ['relu', 'tanh'] learning_rate_init: [0.001, 0.01, 0.02, 0.03, 0.04, 0.05] solver: ['adam']
ELM	n_hidden: [50, 100, 300, 600, 1000] activation: ['relu', 'tanh', 'sigmoid']
SVR	C: [0.1, 1, 10, 100, 500, 1000] Kernel: ['rbf', 'linear'] Gamma: ['scale', 'auto', 0.01, 0.001] Épsilon: [0.01, 0.05, 0.1, 0.5]
DT	D = tree.get_depth() max_depth: [D, D + 5, D + 10, D + 20] i.e. si $D = 6 \rightarrow [6, 11, 16, 26]$ min_samples_split: [1, 2, 3, 4, 5]
RF	n_estimators: [50, 100, 200, 300, 500] max_depth: None = [D, D + 10, D + 20] min_samples_split: [2, 5, 10]
Lasso	alpha: [0.0001, 0.001, 0.01, 0.1, 1, 10] max_iter: [10, 50, 100, 200]
Ridge	alpha: [0.0001, 0.001, 0.01, 0.1, 1, 10] max_iter: [10, 50, 100, 200]

Una vez realizada la búsqueda inicial de hiperparámetros mediante *Grid Search*, se seleccionaron aquellas configuraciones que presentaron un mejor desempeño de acuerdo con las métricas de evaluación. Después, se llevó a cabo una búsqueda fina sobre los intervalos previamente definidos, con el objetivo de ajustar de manera más precisa los hiperparámetros y obtener configuraciones que minimizaran el error de predicción o maximizaran el PRED. La Tabla 30 muestra los parámetros que fueron utilizados para la búsqueda fina, a partir de los hiperparámetros obtenidos en Grid Search. Como resultado de este proceso, se obtuvieron los valores de hiperparámetros para

cada uno de los modelos individuales y de ensamble (véase la Tabla 17). En el caso de los ensambles, estos fueron construidos mediante la combinación de varios modelos base, cuyas predicciones son utilizadas como entradas para un metamodelo encargado de generar la estimación final. Debido a ello, fue necesario realizar la búsqueda de los hiperparámetros tanto de los modelos individuales que conforman el ensamble como del metamodelo, con el propósito de maximizar su capacidad predictiva.

Tabla 30. Intervalos de búsqueda de hiperparámetros para búsqueda fina.

Modelo	Parámetros y valores
KNN	n_neighbors: [n_neighbors - 2, n_neighbors, n_neighbors + 2]
MLP	learning_rate_init: [learning_rate_init * 0.5, learning_rate_init * 2]
ELM	n_hidden: [n_hidden, n_hidden + 10]
SVR	C: [C * 0.5, C, C * 2], epsilon: [epsilon * 0.5, epsilon * 2]
DT	max_depth: [D, D - 2, D + 2] min_samples_split: [min_samples_split - 1, min_samples_split, min_samples_split + 1]
RF	n_estimators: [n_estimators * 0.75, n_estimators, n_estimators * 1.25] max_depth: None = [D, D - 2, D + 2] min_samples_split: [min_samples_split - 1, min_samples_split, min_samples_split + 1]
Lasso	alpha: [alpha * 0.1, alpha * 0.5, alpha, alpha * 2, alpha * 10]
Ridge	alpha: [alpha * 0.1, alpha * 0.5, alpha, alpha * 2, alpha * 10]

Los valores finales seleccionados para cada componente se presentan en la Tabla 31. Es importante mencionar que el modelo LR no fue sometido al proceso de búsqueda de hiperparámetros, por lo que se utilizaron los valores predeterminados proporcionados por la implementación de *Scikit-learn*. Utilizando las configuraciones mostradas en la tabla anterior, y siguiendo el flujo de trabajo definido en la metodología, se procedió a entrenar y evaluar los modelos bajo diferentes combinaciones de sobremuestreo y submuestreo aplicadas previamente en la etapa de preprocesamiento.

Tabla 31. Hiperparámetros encontrados con la búsqueda fina.

Modelo	Valores	Modelo	Valores
KNN	n_neighbors: 2 weights: distance	DT	max_depth: 9 min_samples_split: 2
MLP	hidden_layer_sizes: 150, 100, 50 activation: 'relu' learning_rate_init: 0.001 max_iter: 1000	RF	n_estimators: 100 max_depth: 7 min_samples_split: 2
ELM	activation: 'relu' n_hidden: 150	Lasso	alpha: 1.5 max_iter: 100
SVR	C: 1, kernel: 'linear' gamma: 'scale', Épsilon: 0.01	Ridge	alpha: 0.15 max_iter: 200

4.3.3. Evaluación de modelos

Como siguiente etapa, se evaluaron los dieciséis modelos conformados por nueve modelos individuales (ELM, MLP, KNN, DT, RF, SVR, Lasso, Ridge y LR) y siete modelos de ensamble previamente descritos en la Tabla 17, evaluados utilizando las métricas de rendimiento previamente definidas: MAE, RMSE, MMRE y PRED, con el objetivo de analizar su precisión y capacidad de generalización. Los resultados obtenidos para MAE sobre el conjunto de datos se presentan en la Tabla 32, donde se comparan los distintos modelos bajo diferentes configuraciones de sobremuestreo y submuestreo. Los mejores resultados obtenidos en cada columna, se han resaltado en negritas, correspondiendo a los menores valores de MAE. Se observó que el modelo SVR con estrategia de combinación de 100% de sobremuestreo y 50% de submuestreo obtuvo el mejor desempeño, alcanzando un MAE de 0.070.

Tabla 32. Resultados de MAE sobre el conjunto de datos para proyectos de software.

Modelo	100%				200%			
	50%	40%	30%	20%	50%	40%	30%	20%
KNN	1.243	1.368	1.368	2.119	1.072	1.447	1.206	1.504
MLP	1.648	1.102	0.927	0.558	0.620	0.681	0.661	7.407
ELM	2.119	1.745	2.849	1.868	1.863	1.373	1.253	1.566
SVR	0.070	1.961	1.612	1.281	1.973	1.390	1.269	0.894
DT	0.340	2.182	2.699	3.888	0.877	3.218	2.072	2.607
RF	0.155	2.034	2.514	3.665	0.663	3.329	2.399	2.889
LR	2.806	2.288	15.417	6.554	2.379	1.848	1.694	1.539
Lasso	2.738	2.076	2.012	1.619	2.364	1.617	1.513	1.235
Ridge	2.805	2.204	2.119	1.764	2.355	1.789	1.669	1.256
EH1	0.213	1.331	1.781	1.650	1.247	1.153	0.982	1.324
EH2	33.433	1.496	5.765	2.877	1.025	2.497	1.631	1.844
EH3	0.190	1.216	2.253	2.243	0.632	1.810	1.193	1.536
EH4	0.213	1.048	2.532	1.730	0.874	1.118	0.895	1.121
EH5	0.270	2.197	3.326	6.514	1.208	4.505	3.632	3.645
EH6	0.200	1.120	1.429	1.483	0.872	1.045	0.907	1.153
EH7	0.201	1.055	1.361	1.486	0.877	1.099	0.892	1.136

De manera complementaria, la Tabla 33 presenta los resultados obtenidos para RMSE, permitiendo evaluar el comportamiento de los modelos considerando penalizaciones mayores a errores grandes. Los valores resaltados en negritas, forman parte de los mejores valores de cada columna, correspondiendo a los menores valores obtenidos entre las diferentes estrategias de combinación evaluadas. El mejor resultado obtenido fue el modelo SVR en la combinación del 100%

de sobremuestreo y 50% de submuestreo, con un RMSE de 0.093. Asimismo, la Tabla 34 muestra los valores correspondientes de MMRE, la cual permite analizar el error relativo promedio de los modelos. Los valores resaltados en negritas, indican los menores valores de error relativo obtenidos y mejores valores obtenidos por cada combinación.

Tabla 33. Resultados de RMSE sobre el conjunto de datos para proyectos de software.

Modelo	100%				200%			
	50%	40%	30%	20%	50%	40%	30%	20%
KNN	2.398	2.780	2.336	4.006	2.010	3.102	3.335	3.347
MLP	2.568	2.139	2.760	1.933	2.372	2.216	2.832	8.971
ELM	2.459	2.737	5.185	3.110	3.104	2.328	2.444	0.094
SVR	0.093	2.738	3.587	3.053	3.052	3.341	3.274	2.514
DT	1.567	6.353	7.095	7.739	1.607	8.443	6.323	6.368
RF	1.748	4.625	4.628	6.266	1.165	8.599	6.041	5.968
LR	3.516	3.053	41.912	17.158	3.107	3.429	3.005	2.811
Lasso	3.205	2.966	2.958	2.671	3.098	3.178	3.069	2.214
Ridge	3.049	3.024	2.995	2.684	3.096	3.416	2.998	2.139
EH1	1.874	2.387	3.294	3.072	1.967	2.232	2.341	2.911
EH2	1.254	3.555	12.903	4.840	1.726	5.582	4.503	3.774
EH3	1.114	2.920	4.491	3.933	0.962	4.473	3.304	3.118
EH4	1.534	1.987	5.132	3.109	1.344	2.246	2.190	2.381
EH5	2.398	5.200	5.497	10.321	2.080	9.405	7.960	6.836
EH6	1.418	2.025	2.566	2.800	1.345	2.194	2.247	2.542
EH7	1.410	2.044	2.489	2.751	1.350	2.260	2.189	2.475

El mejor desempeño correspondió al modelo SVR bajo la combinación 100% y 50%, de sobremuestreo y submuestreo respectivamente, obteniendo un MMRE de 0.002. Finalmente, la Tabla 35 presenta los resultados de PRED, la cual indica el porcentaje de predicciones cuyo error se encuentra dentro del 25% del valor real. En este caso, los resultados resaltados en negritas corresponden a los mejores y mayores valores obtenidos, ya que representan un mejor desempeño predictivo.

El valor máximo de 100% fue alcanzado por varios modelos, entre ellos KNN, MLP, SVR, DT, RF, EH1, EH2, EH3, EH4, EH6 y EH7, en la estrategia de combinación del 100% de sobremuestreo y 50% de submuestreo. Los resultados obtenidos permitieron determinar el impacto de las diferentes combinaciones de balanceo de datos en el desempeño de los modelos. En general, se observó que las combinaciones con mayores porcentajes de sobremuestreo tendieron a mejorar la precisión de los modelos, particularmente en aquellos casos donde la variable objetivo presentó una distribución desbalanceada. Asimismo, los modelos de ensamble mostraron un comportamiento

competitivo frente a los modelos individuales, logrando en varios casos reducir el error de predicción. Este comportamiento sugiere que la combinación de múltiples modelos permite capturar de manera más efectiva la complejidad de los datos. En conjunto, el proceso de búsqueda de hiperparámetros permitió obtener modelos mejor ajustados, evidenciando la importancia de la selección adecuada de hiperparámetros y su impacto directo en el desempeño de los modelos de estimación.

Tabla 34. Resultados de MMRE sobre el conjunto de datos para proyectos de software.

Modelo	100%				200%			
	50%	40%	30%	20%	50%	40%	30%	20%
KNN	0.040	0.042	0.041	0.061	0.031	0.042	0.036	0.043
MLP	0.022	0.032	0.025	0.015	0.018	0.019	0.019	0.206
ELM	0.053	0.053	0.089	0.059	0.052	0.045	0.040	0.044
SVR	0.002	0.082	0.069	0.054	0.082	0.058	0.053	0.037
DT	0.031	0.061	0.071	0.116	0.025	0.107	0.056	0.078
RF	0.032	0.056	0.074	0.112	0.021	0.109	0.080	0.090
LR	0.095	0.082	0.582	0.263	0.082	0.068	0.061	0.051
Lasso	0.090	0.075	0.073	0.057	0.081	0.061	0.055	0.041
Ridge	0.092	0.081	0.079	0.061	0.081	0.067	0.061	0.043
EH1	0.038	0.039	0.055	0.048	0.036	0.037	0.031	0.036
EH2	0.024	0.037	0.192	0.079	0.026	0.075	0.041	0.046
EH3	0.022	0.033	0.080	0.063	0.019	0.059	0.036	0.045
EH4	0.031	0.030	0.090	0.048	0.026	0.035	0.028	0.031
EH5	0.041	0.061	0.101	0.180	0.034	0.136	0.113	0.100
EH6	0.030	0.032	0.044	0.042	0.026	0.033	0.028	0.031
EH7	0.029	0.030	0.041	0.026	0.035	0.028	0.031	0.026

4.3.4. Obtención del mejor modelo de estimación

Resultó fundamental analizar el desempeño de cada modelo para identificar la alternativa más adecuada, ya que cada técnica posee características particulares que pueden influir en su capacidad predictiva dependiendo de las condiciones del conjunto de datos. A partir de los resultados obtenidos, se observó que los modelos no lineales, especialmente SVR, así como algunos modelos de ensamble como EH3, EH4, EH6 y EH7, presentaron un desempeño superior al de otros enfoques, alcanzando menores valores de error y una mayor estabilidad en las distintas combinaciones evaluadas.

En particular, SVR destacó como el mejor modelo, al obtener los mínimos valores para MAE, RMSE, MMRE y un máximo valor para PRED. La Tabla 36 presenta los resultados del mejor modelo seleccionado, considerando la configuración del 100% de sobremuestreo y 50% de submuestreo.

En este sentido, el modelo seleccionado corresponde a aquel que logró un equilibrio entre precisión y robustez, destacando por mantener un buen desempeño en múltiples mediciones de rendimiento y combinaciones de sobremuestreo y submuestreo. Por tanto, este modelo es integrado en la aplicación web desarrollada, con el objetivo de proporcionar estimaciones confiables y consistentes en entornos reales.

Tabla 35. Resultados de PRED sobre el conjunto de datos para proyectos de software.

Modelo	100%				200%			
	50%	40%	30%	20%	50%	40%	30%	20%
KNN	100	99.537	100	96.429	99.060	97.857	99.177	97.561
MLP	100	99.074	97.191	97.857	97.806	98.571	97.531	81.463
ELM	90.625	98.611	88.764	92.143	97.492	97.857	97.531	97.073
SVR	100	84.259	85.955	89.286	91.536	87.500	88.889	92.683
DT	100	96.296	93.258	82.143	100	91.429	93.827	87.805
RF	100	97.685	96.067	85.000	100	90.714	92.181	87.317
LR	90.625	91.667	83.708	83.571	91.850	87.857	91.770	94.176
Lasso	90.625	91.204	90.449	92.857	91.850	88.571	90.947	94.634
Ridge	90.625	92.130	91.011	92.857	91.850	87.857	91.770	95.122
EH1	100	98.611	95.506	97.143	100	96.786	97.119	97.561
EH2	100	96.296	84.831	93.571	100	91.786	97.119	95.610
EH3	100	97.685	92.135	95.000	100	91.786	97.531	95.610
EH4	100	99.074	91.011	99.286	100	97.143	97.942	99.024
EH5	98.819	96.296	89.326	64.286	100	84.286	86.831	81.463
EH6	100	99.074	96.629	100	100	97.143	97.531	98.049
EH7	100	99.074	97.191	100	97.143	97.942	98.049	97.143

Tabla 36. Resultados del mejor modelo.

Mejor modelo	MAE	RMSE	MMRE	PRED
SVR	0.070	0.093	0.002	100

4.4. Resultados de estimación con *sprints*

De manera similar, en esta sección se presentan los resultados obtenidos evaluando los modelos seleccionados con el conjunto de datos sintéticos obtenido para *sprints*, en conjunto con las estrategias de sobremuestreo previamente definidas. El objetivo fue seleccionar el “mejor modelo”

que sería integrado a la aplicación web implementada. Como fue descrito anteriormente, se aplicaron las mismas técnicas de preprocesamiento y búsqueda de hiperparámetros.

Posteriormente, se aplicó la técnica de sobremuestreo SMOGN con el propósito de mitigar el desbalance presente en la variable objetivo NSP. Esta estrategia permitió incrementar la cantidad de ejemplos en las regiones menos representadas mediante la generación de instancias sintéticas, favoreciendo una distribución más equilibrada de los datos. La Figura 4.7 muestra la distribución del conjunto de datos antes y después del proceso de sobremuestreo, donde puede observarse una mayor representación de los valores extremos y una reducción de la concentración de observaciones en las regiones más densamente pobladas, contribuyendo a mejorar la capacidad de generalización de los modelos evaluados.

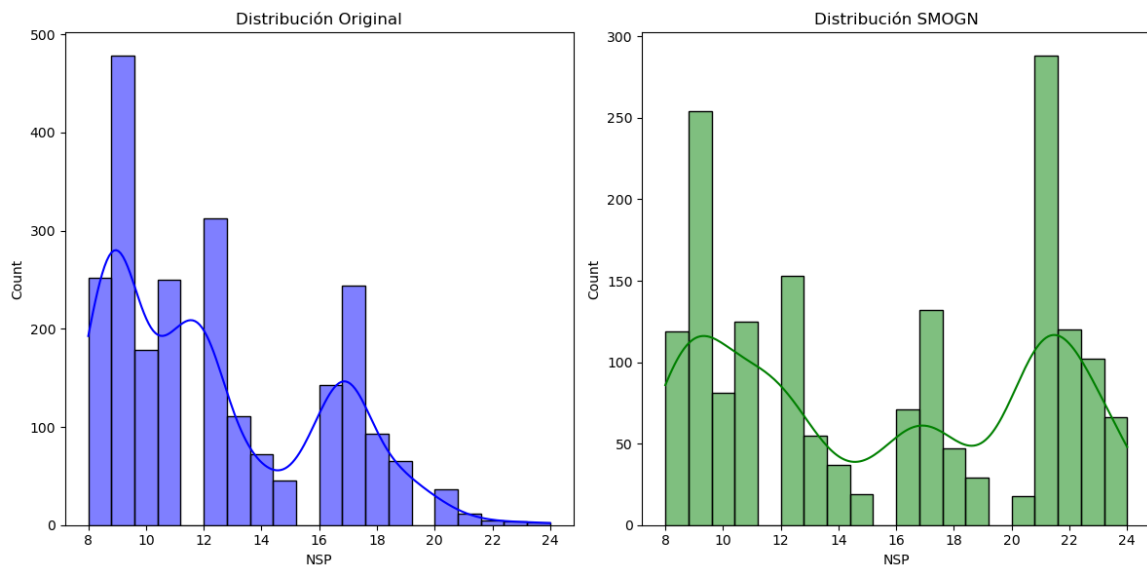


Figura 4.7. Distribución del conjunto de datos para *sprints*.

4.4.1. Preprocesamiento

Para el caso del conjunto de datos a nivel *sprint*, fue necesario aplicar un proceso de preparación de datos que permitiera reducir el impacto del desbalance y la variabilidad inherente a este tipo de información. Una vez aplicado el balanceo, se procedió a la normalización de las variables, con el objetivo de estandarizar las escalas de los datos y favorecer el desempeño de los modelos. Aunado a esto, se realizó el análisis para comparar si la selección de características proporcionaba mejores resultados a los modelos seleccionados o no, obteniendo como resultado que usando todas las características proporcionaban mejores resultados.

4.4.2. Búsqueda de hiperparámetros

Con los datos previamente preparados, se llevó a cabo el ajuste de los modelos mediante la búsqueda de sus hiperparámetros. Para este propósito, se empleó una estrategia de búsqueda basada en *Grid Search*. Los rangos de búsqueda considerados para los distintos modelos fueron equivalentes a los utilizados en la sección anterior, asegurando así una base común para la comparación de resultados.

Los rangos de valores de hiperparámetros utilizados para los modelos individuales, fueron presentados en la Tabla 29 y que posteriormente serán integrados a los ensambles heterogéneos presentados en la Tabla 17. Posteriormente, se realizó una etapa de ajuste fino sobre las configuraciones encontradas en la primera búsqueda, tomando como punto de partida los valores que mostraron el mejor desempeño durante la exploración inicial, con el propósito de refinar la selección de hiperparámetros y maximizar la capacidad predictiva de los modelos. Lo anterior, permite explorar regiones más específicas del espacio de búsqueda y obtener configuraciones más precisas que contribuyeran a mejorar los resultados de estimación (véanse las Tablas 37 y 38).

Tabla 37. Intervalos de búsqueda de hiperparámetros para búsqueda fina.

Modelo	Parámetros y valores
KNN	n_neighbors: [n_neighbors - 2, n_neighbors, n_neighbors + 2]
MLP	learning_rate_init: [learning_rate_init * 0.5, learning_rate_init * 2]
ELM	n_hidden: [n_hidden, n_hidden + 10]
SVR	C: [C * 0.5, C, C * 2], epsilon: [epsilon * 0.5, epsilon * 2]
DT	max_depth: [D, D - 2, D + 2] min_samples_split: [min_samples_split - 1, min_samples_split, min_samples_split + 1]
RF	n_estimators: [n_estimators * 0.75, n_estimators, n_estimators * 1.25] max_depth: None = [D, D - 2, D + 2] min_samples_split: [min_samples_split - 1, min_samples_split, min_samples_split + 1]
Lasso	alpha: [alpha * 0.1, alpha * 0.5, alpha, alpha * 2, alpha * 10]
Ridge	alpha: [alpha * 0.1, alpha * 0.5, alpha, alpha * 2, alpha * 10]

Tabla 38. Hiperparámetros encontrados con la búsqueda fina.

Modelo	Valores	Modelo	Valores
KNN	n_neighbors: 2, weights: distance	DT	max_depth: 9 min_samples_split: 2
MLP	hidden_layer_sizes: 150, 100, 50 activation: 'relu', learning_rate_init: 0.001 max_iter: 1000	RF	n_estimators: 100, max_depth: 7 min_samples_split: 2
ELM	activation: 'relu', n_hidden: 150	Lasso	alpha: 1.5, max_iter: 100
SVR	C: 1, kernel: 'linear', gamma: 'scale', Épsilon: 0.01	Ridge	alpha: 0.15, max_iter: 200

4.4.3. Evaluación de modelos

Una vez entrenados los modelos con las mejores configuraciones, se procedió a evaluar su desempeño utilizando las métricas de rendimiento definidas: MAE, RMSE, MMRE y PRED. La Tabla 39 muestra los valores del error MAE para todos los modelos evaluados. Además, el 100%, 150% y 200% corresponden al porcentaje de sobremuestreo y el 20%, 30%, 40%, y 50% corresponden al porcentaje de submuestreo utilizado. Los mejores resultados se encuentran resaltados en negritas,

correspondiendo a los menores valores de error. El mejor desempeño fue obtenido por el ensamble EH7 bajo la combinación del 200% de sobremuestreo y 50% de submuestreo, alcanzando un MAE de 0.340.

Por otro lado, la Tabla 40 presenta los resultados obtenidos para RMSE donde, de manera similar, el 100%, 150% y 200% corresponden al porcentaje de sobremuestreo y el 20%, 30%, 40%, y 50% corresponden al porcentaje de submuestreo utilizado, permitiendo así evaluar el comportamiento de los modelos considerando una mayor penalización ante errores de gran magnitud. Los mejores resultados de cada combinación por columna se encuentran resaltados en negritas, correspondiendo a los menores valores de error obtenidos para cada modelo.

El mejor resultado fue alcanzado por el ensamble EH7 en la combinación de 200% de sobremuestreo y 50% de submuestreo, obteniendo un RMSE de 0.623. Este resultado muestra la capacidad de los modelos de ensamble para reducir los errores de predicción más significativos, superando a los modelos individuales. Asimismo, se observa que las combinaciones con mayores niveles de sobremuestreo tendieron a producir valores de RMSE más bajos, lo que sugiere que una mayor variabilidad y representatividad de los datos favorece el desempeño de los modelos en la estimación a nivel *sprint*.

De manera similar, la Tabla 41 muestra los valores correspondientes de MMRE, la cual permite analizar el error relativo promedio de los modelos. Asimismo, el 100%, 150% y 200% corresponden al porcentaje de sobremuestreo y el 20%, 30%, 40%, y 50% corresponden al porcentaje de submuestreo utilizado. Los valores resaltados en negritas, indican los menores valores de error relativo obtenidos por cada columna. El menor MMRE fue obtenido por los ensambles EH3, EH6 y EH7, bajo la combinación 200% de sobremuestreo y 50% de submuestreo, alcanzando un valor de 0.021.

Tabla 39. Resultados de MAE sobre el conjunto de datos para *sprints*.

Modelo	100%				200%			
	50%	40%	30%	20%	50%	40%	30%	20%
KNN	0.384	0.430	0.518	0.629	0.388	0.434	0.522	0.633
MLP	0.562	0.548	0.593	0.596	0.563	0.571	0.638	0.720
ELM	0.978	0.929	0.888	0.884	1.022	1.032	0.953	1.034
SVR	0.455	0.484	0.536	0.572	0.446	0.471	0.530	0.584
DT	0.401	0.430	0.447	0.481	0.384	0.422	0.439	0.465
RF	0.385	0.393	0.427	0.474	0.372	0.388	0.419	0.470
LR	0.521	0.522	0.533	0.592	0.533	0.538	0.563	0.651
Lasso	0.523	0.523	0.532	0.550	0.532	0.538	0.566	0.596
Ridge	0.534	0.542	0.551	0.566	0.533	0.538	0.552	0.576
EH1	0.447	0.479	0.505	0.639	0.411	0.485	0.521	0.581
EH2	0.394	0.414	0.447	0.498	0.390	0.444	0.443	0.478
EH3	0.349	0.365	0.395	0.432	0.357	0.405	0.441	0.428

EH4	0.375	0.398	0.419	0.481	0.372	0.416	0.453	0.470
EH5	0.374	0.412	0.422	0.452	0.383	0.417	0.456	0.473
EH6	0.365	0.408	0.419	0.434	0.348	0.411	0.429	0.433
EH7	0.354	0.396	0.397	0.428	0.340	0.399	0.413	0.413

En términos generales, los modelos de ensamble presentaron los mejores resultados en esta medida, seguidos por RF y DT, los cuales también mostraron errores relativos bajos y consistentes. Asimismo, se observa que el incremento en el porcentaje de sobremuestreo favoreció la reducción del MMRE en varios modelos, evidenciando que una mayor variabilidad y representatividad de los datos contribuye a mejorar la capacidad de aprendizaje y generalización. Los resultados obtenidos sugieren que los ensambles heterogéneos son capaces de capturar de manera más efectiva las relaciones entre las variables y la estimación objetivo, reduciendo el error relativo y proporcionando estimaciones más precisas para el esfuerzo requerido a nivel *sprint*.

Finalmente, la Tabla 42 presenta los resultados de PRED, la cual indica el porcentaje de predicciones cuyo error se encuentra dentro del 25% del valor real. En este caso, el 100%, 150% y 200% corresponden al porcentaje de sobremuestreo y el 20%, 30%, 40%, y 50% corresponden al porcentaje de submuestreo utilizado. Los resultados resaltados en negritas corresponden a los mayores valores obtenidos. El valor máximo de 100% fue alcanzado por diversos modelos y combinaciones, destacando los ensambles EH2 y EH3, que mantuvieron este resultado en todas las combinaciones de sobremuestreo y submuestreo evaluadas.

Tabla 40. Resultados de RMSE sobre el conjunto de datos para *sprints*.

Modelo	100%				200%			
	50%	40%	30%	20%	50%	40%	30%	20%
KNN	0.829	0.879	0.982	1.112	0.836	0.885	0.988	1.116
MLP	0.754	0.743	0.784	0.808	0.777	0.777	0.861	0.980
ELM	1.268	1.243	1.220	1.161	1.327	1.322	1.218	1.403
SVR	0.692	0.724	0.802	0.819	0.669	0.706	0.805	0.867
DT	0.819	0.861	0.902	0.996	0.782	0.850	0.885	0.981
RF	0.698	0.692	0.727	0.800	0.665	0.685	0.710	0.808
LR	0.692	0.691	0.711	0.782	0.711	0.718	0.758	0.884
Lasso	0.696	0.697	0.724	0.750	0.719	0.729	0.766	0.808
Ridge	0.710	0.718	0.733	0.751	0.710	0.719	0.744	0.778
EH1	0.758	0.777	0.831	1.005	0.703	0.802	0.838	0.881
EH2	0.656	0.677	0.705	0.816	0.625	0.714	0.727	0.761
EH3	0.647	0.662	0.678	0.745	0.639	0.702	0.776	0.786
EH4	0.638	0.667	0.703	0.753	0.644	0.691	0.734	0.749

EH5	0.673	0.716	0.723	0.755	0.686	0.709	0.763	0.767
EH6	0.644	0.696	0.720	0.746	0.642	0.706	0.740	0.756
EH7	0.631	0.694	0.682	0.725	0.623	0.698	0.698	0.712

En conclusión, los resultados evidencian un elevado nivel de precisión en las estimaciones realizadas a nivel *sprint*, ya que la mayoría de los modelos alcanzó valores superiores al 99%. Asimismo, los ensambles heterogéneos mostraron un comportamiento más consistente frente a las distintas configuraciones de preprocesamiento, manteniendo porcentajes cercanos o iguales al 100%. Estos resultados refuerzan la capacidad de los ensambles para generar estimaciones confiables y robustas.

4.4.4. Obtención del mejor modelo de estimación

Con base en los resultados obtenidos, se llevó a cabo un análisis comparativo entre los modelos evaluados con el fin de identificar aquel que presenta el mejor desempeño en el contexto de estimación a nivel *sprint*. A partir de este análisis, se observó que, al igual que en el caso de proyectos de software, los modelos no lineales y los ensambles presentan un mejor comportamiento general.

Asimismo, la mayor variabilidad presente en los datos a nivel *sprint* permitió disponer de una mayor diversidad de escenarios de entrenamiento y evaluación, lo que favoreció la capacidad de los modelos para aprender patrones más representativos del proceso de desarrollo. Esta característica contribuyó a obtener modelos más robustos y con una mejor capacidad de generalización, reflejándose en un desempeño competitivo entre las distintas técnicas evaluadas. Por ello, además de considerar los valores de error mínimo obtenidos, se analizó la estabilidad de los modelos a través de las distintas métricas de rendimiento.

Tabla 41. Resultados de MMRE sobre el conjunto de datos para *sprints*.

Modelo	100%				200%			
	50%	40%	30%	20%	50%	40%	30%	20%
KNN	0.023	0.026	0.032	0.039	0.023	0.026	0.033	0.039
MLP	0.038	0.036	0.040	0.037	0.034	0.035	0.041	0.049
ELM	0.081	0.066	0.068	0.064	0.072	0.076	0.068	0.072
SVR	0.028	0.030	0.035	0.038	0.028	0.030	0.036	0.039
DT	0.023	0.024	0.026	0.028	0.023	0.024	0.026	0.027
RF	0.023	0.023	0.026	0.029	0.022	0.023	0.026	0.028
LR	0.035	0.034	0.035	0.040	0.035	0.035	0.037	0.045
Lasso	0.034	0.033	0.034	0.035	0.034	0.034	0.037	0.040
Ridge	0.033	0.034	0.035	0.036	0.034	0.034	0.036	0.038
EH1	0.027	0.031	0.034	0.043	0.025	0.030	0.034	0.039
EH2	0.024	0.025	0.027	0.029	0.024	0.026	0.027	0.029

EH3	100	100	100	100	100	100	100	100
EH4	100	100	99.8	99.8	100	100	99.8	99.8
EH5	100	100	99.3	99.6	100	100	99.8	99.3
EH6	99.8	100	99.3	99.3	100	99.8	99.1	99.1
EH7	100	100	99.8	99.8	100	100	99.8	99.8

Tabla 43. Resultados del mejor modelo.

Mejor modelo	MAE	RMSE	MMRE	PRED
EH7	0.340	0.623	0.021	100

Para acceder el usuario debe ingresar al siguiente enlace <https://mis-lab.com/>, donde podrá visualizar la pantalla de inicio, que se muestra en la Figura 4.8. La aplicación contempla tres tipos de usuario: invitado, estimador y administrador. Los usuarios con rol de estimador y administrador podrán ingresar a la aplicación web, al crear su cuenta e iniciar sesión con sus credenciales previamente creadas como se muestra en la Figura 4.9.

Predicción de esfuerzo

Impulsa tus estimaciones con modelos avanzados que analizan datos históricos, esfuerzo, complejidad técnica e historias de usuario para proyectar la duración real de tus proyectos.



Iniciar sesión

EMAIL

correo.ejemplo@example.com

PASSWORD

Tu contraseña

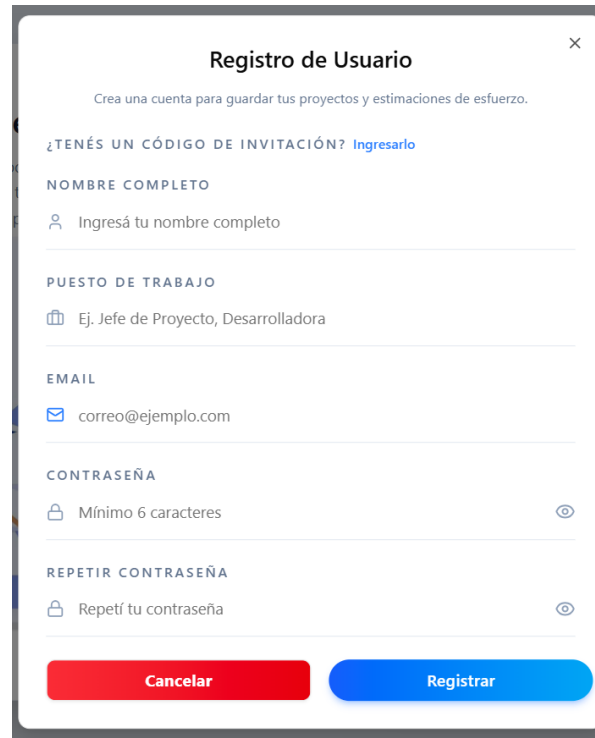
☐ Recordarme [¿Olvidaste tu contraseña?](#)

LOGIN

[¿No eres miembro? Crear cuenta](#)

[Entrar como invitado](#)

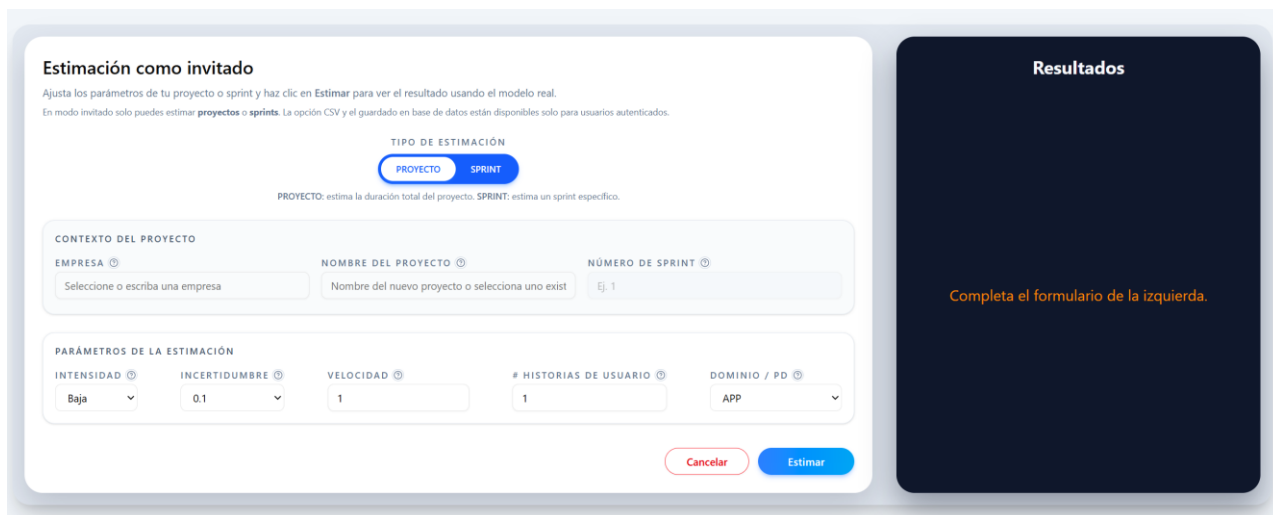
Figura 4.8. Inicio de sesión.



The image shows a 'Registro de Usuario' (User Registration) form. At the top, it says 'Crea una cuenta para guardar tus proyectos y estimaciones de esfuerzo.' Below this is a link '¿TENÉS UN CÓDIGO DE INVITACIÓN? Ingresarlo'. The form has several input fields: 'NOMBRE COMPLETO' with a person icon and placeholder 'Ingresá tu nombre completo'; 'PUESTO DE TRABAJO' with a briefcase icon and placeholder 'Ej. Jefe de Proyecto, Desarrolladora'; 'EMAIL' with an envelope icon and placeholder 'correo@ejemplo.com'; 'CONTRASEÑA' with a lock icon, placeholder 'Mínimo 6 caracteres', and an eye icon to toggle visibility; and 'REPETIR CONTRASEÑA' with a lock icon, placeholder 'Repetí tu contraseña', and an eye icon to toggle visibility. At the bottom are two buttons: a red 'Cancelar' button and a blue 'Registrar' button.

Figura 4.9. Registro de usuario.

Ahora bien, el usuario invitado podrá ingresar y estimar proyectos de software en duración de semanas y el esfuerzo con puntos de historia para *sprints*, como se muestra en la Figura 4.10. Los resultados obtenidos al momento de generar la estimación, se presentará del lado derecho de la interfaz.



The image shows a 'Estimación como invitado' (Estimation as guest) interface. It has a title 'Estimación como invitado' and a subtitle 'Ajusta los parámetros de tu proyecto o sprint y haz clic en Estimar para ver el resultado usando el modelo real. En modo invitado solo puedes estimar proyectos o sprints. La opción CSV y el guardado en base de datos están disponibles solo para usuarios autenticados.' Below this is a 'TIPO DE ESTIMACIÓN' section with two buttons: 'PROYECTO' (selected) and 'SPRINT'. A note below says 'PROYECTO: estima la duración total del proyecto. SPRINT: estima un sprint específico.' The main form is divided into two sections: 'CONTEXTO DEL PROYECTO' and 'PARÁMETROS DE LA ESTIMACIÓN'. The 'CONTEXTO DEL PROYECTO' section has three input fields: 'EMPRESA' (placeholder 'Seleccione o escriba una empresa'), 'NOMBRE DEL PROYECTO' (placeholder 'Nombre del nuevo proyecto o selecciona uno exist'), and 'NÚMERO DE SPRINT' (placeholder 'Ej. 1'). The 'PARÁMETROS DE LA ESTIMACIÓN' section has five input fields: 'INTENSIDAD' (dropdown menu with 'Baja' selected), 'INCERTIDUMBRE' (dropdown menu with '0.1' selected), 'VELOCIDAD' (input field with '1' entered), '# HISTORIAS DE USUARIO' (input field with '1' entered), and 'DOMINIO / PD' (dropdown menu with 'APP' selected). At the bottom are two buttons: a red 'Cancelar' button and a blue 'Estimar' button. To the right of the form is a dark blue sidebar with the title 'Resultados' and a message 'Completa el formulario de la izquierda.'

Figura 4.10. Estimación como invitado.

De manera similar, el estimador y administrador, puede estimar proyectos y *sprints* ingresando datos como Intensidad, Incertidumbre, Velocidad, Historias de Usuario y Dominio del proyecto. Además, podrá subir un archivo .CSV, con filas que representan muestras de proyectos y *sprints* y que cumplan con la información de las siguientes columnas: Tipo, Empresa, Proyecto, Sprint,

Incertidumbre, Intensidad, Velocidad, Historias de usuario y Dominio, para poder generar la estimación de sus correspondientes proyectos y *sprints* (véase Figura 4.11).

Figura 4.11. Subir archivo a la aplicación web.

En caso de que la estimación seleccionada se realice de manera exitosa, los resultados se presentarán como lo muestra la Figura 4.12. En ella, se muestra los datos ingresados y el resultado de la estimación realizada, tales como Fecha y hora de la estimación, Empresa, Proyecto, UF, Intensidad, UV, Velocidad, PD, semanas, la duración en meses y semanas para los proyectos de software.

FECHA	HORA	EMPRESA	PROYECTO	UF	INTENSIDAD	HU	VELOCIDAD	PD	SEMANAS	DURACIÓN	SPRINTS	ACCIONES
1/7/2026	02:09:56 a.m.	Empresa2	Proyecto2	0.1	4	25	8	4	35.96	9 meses	Ver 2 sprints	
1/7/2026	02:09:55 a.m.	Empresa1	Proyecto1	0.6	1	25	8	1	27.12	6 meses y 3 semanas	Ver 3 sprints	
12/6/2026	03:37:18 p.m.	Empresa2	Proyecto 3	0.5	4	10	5	1	18.11	4 meses y 2 semanas	Ver 5 sprints	

Figura 4.12. Estimaciones realizadas.

Para la obtención de estimaciones de proyectos o *sprints*, se tomaron en cuenta tres casos para generarlas:

1. Estimar solo proyectos de software.
2. Estimar un proyecto de software y posteriormente sus *sprints* asociados, para ello los usuarios deberán seleccionar el botón “Agregar *sprint*”.
3. Estimar solo *sprints*, sin estimar un proyecto de software.

Una vez obtenidas las estimaciones la aplicación permite descargarlas en archivos .PDF, dando clic en el botón “Generar reporte PDF”. Para el caso del archivo .CSV, se podrán agregar diversas estimaciones en conjunto, tomando en cuenta que se debe seguir la estructura de datos para poder subir el archivo en cuestión, como se muestra en la Figura 4.13. Nótese, que la siguiente imagen muestra ejemplos de los casos anteriormente descritos.

Tipo	Empresa	Proyecto	Sprint	Incertidumbre	Intensidad	Velocidad	Historias de usuario	Dominio
Proyecto	Empresa1	Proyecto1		0.6	1	8	25	1
Proyecto	Empresa2	Proyecto2		0.1	4	8	25	4
Sprint	Empresa1	Proyecto1	1	0.1	1	8	15	4
Sprint	Empresa2	Proyecto2	2	0.1	4	8	15	4
Sprint	Empresa1	Proyecto1	3	0.1	9	8	15	4
Sprint	Empresa2	Proyecto2	1	0.3	4	8	15	4
Sprint	Empresa1	Proyecto1	2	0.4	4	8	15	4

Figura 4.13. Muestras para el archivo .CSV.

En caso de que, al momento de cargar el archivo, o al estimar, se presente un error, se mostrará una pantalla con el botón de “Ir a inicio”, cuya funcionalidad dependerá del tipo de usuario, si es un usuario estimador, será redirigido a la página de sus proyectos guardados (véase Figura 4.12), o si es un usuario invitado, será redirigido a la página principal (véase Figura 4.8).

Asimismo, el administrador cuenta con más funcionalidades, podrá analizar y gestionar las estimaciones generadas por varios estimadores tales como el Product Owner o miembros del equipo Scrum (véase Figura 4.14).

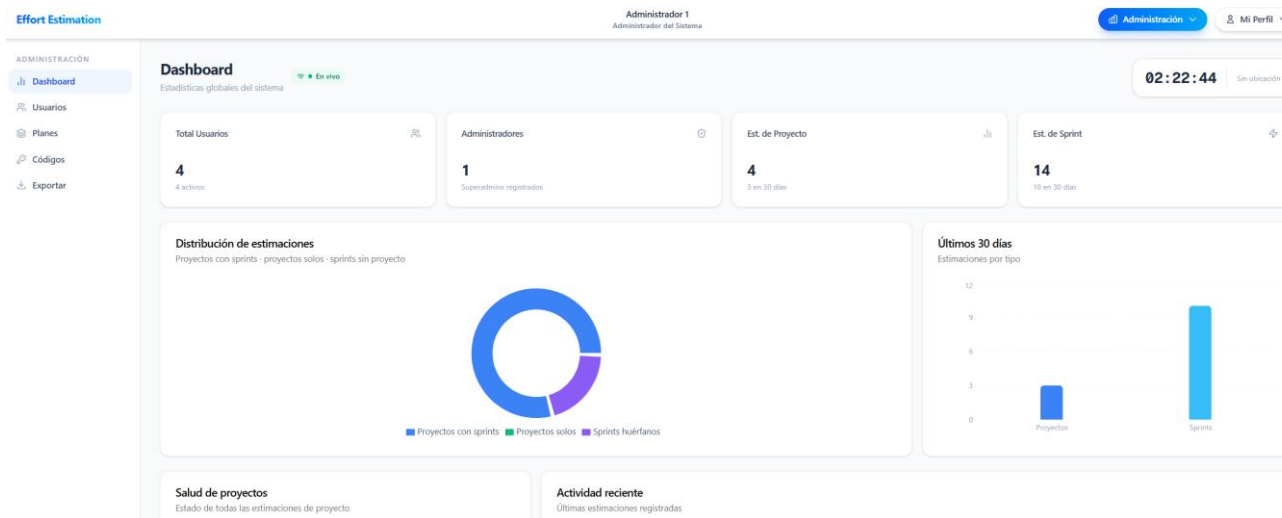


Figura 4.14. Página para administrador.

A continuación, se describen cada uno de los módulos que integran las funcionalidades del administrador:

- **Dashboard:** En este módulo se muestra información para la gestión, monitoreo y seguimiento de todos los proyectos de la empresa. Se podrá visualizar indicadores relevantes, como el número total de estimadores registrados, total de administradores secundarios, la distribución de estimaciones generadas y actividad reciente dentro de la aplicación.

- Usuario: En este módulo se listan y administran los usuarios registrados, mostrando información asociada a sus roles, los cuales podrán ser activados o desactivados por el administrador.
- Códigos: En este módulo permite la generación y administración de códigos de acceso creados por estimadores o administradores secundarios al ingresar a la plataforma. Su propósito es fortalecer la seguridad y confidencialidad de la información relacionada con los proyectos de la empresa, controlando el acceso a funcionalidades y recursos específicos.
- Exportar: En este módulo se ofrece la funcionalidad de exportación de información almacenada en la aplicación web, incluyendo el listado de usuarios registrados, datos de proyectos y empresas, estimaciones generadas e historial de códigos habilitados para el acceso a la aplicación.

4.6. Resultados del caso de estudio

La experimentación en la Ingeniería de Software debe promover el avance del área a través de un proceso iterativo de aprendizaje (Erthal et al., 2023). Bajo esta premisa, el enfoque experimental se ha aplicado durante las últimas décadas en diversos entornos para estudiar y validar las tecnologías de software. Al respecto, Vegas y Juristo (2024) señalaron que experimentar en Ingeniería de Software implica demostrar con hechos y rigor científico las suposiciones, hipótesis y creencias que abundan alrededor de la construcción y uso del software. Por lo tanto, la adopción de métodos de investigación formales resulta indispensable para trasladar el paradigma experimental clásico a este campo de estudio y validar científicamente cualquier propuesta.

En este contexto, Wohlin et al. (2024) identificaron tres estrategias empíricas principales para el desarrollo de investigaciones en Ingeniería de Software: las encuestas, los experimentos y los casos de estudio. Estos últimos se utilizan formalmente para investigar un fenómeno o entidad única dentro de un espacio de tiempo específico, recopilando información detallada de un proyecto real en su entorno natural. Cuando el objetivo es evaluar el impacto de un nuevo método, Kitchenham y Madeyski (2024) sugirieron el uso de proyectos piloto para contrastar los efectos del cambio respecto a una línea base. Si bien los casos de estudio ofrecen la ventaja de ser más viables de planificar en entornos de desarrollo reales, poseen la limitación de que sus resultados son difíciles de generalizar y complejos de interpretar de manera universal (Schoch, 2020).

A partir de estas consideraciones, en esta tesis se seleccionó el caso de estudio como la estrategia empírica para la validación de la hipótesis planteada en el Capítulo 1. De acuerdo con las directrices de Wohlin et al. (2024), existen tres variantes metodológicas para diseñar un caso de estudio:

1. Comparación directa: Contrastar los resultados obtenidos mediante la aplicación de una nueva propuesta frente a una línea base preexistente.
2. Proyectos hermanos: Desarrollar dos proyectos en paralelo, utilizando uno de ellos como grupo de control (i.e., línea base) y el otro como grupo experimental.
3. Aislamiento de componentes: Aplicar la nueva propuesta únicamente sobre componentes seleccionados y comparar su desempeño con aquellos que mantuvieron el enfoque tradicional.

Para los fines de la presente tesis, se optó por la primera estrategia (comparación directa). De esta manera, se pretendió contrastar una nueva propuesta (i.e., el modelo de aprendizaje computacional) contra una línea base con el fin de demostrar validez científica. En este sentido, la

nueva propuesta estuvo representada por el modelo EH7, diseñado para predecir el esfuerzo de *sprints* a partir de variables históricas de los proyectos; mientras que la línea base representó al statu quo, es decir, el enfoque tradicional con el que una organización desarrolladora de software estima actualmente. En el caso particular de este caso de estudio, el personal de la organización calcula el esfuerzo basándose en su experiencia usando, principalmente, la técnica de póker de planificación. Las siguientes secciones muestran el diseño y desarrollo del caso de estudio.

4.6.1. Participantes

Se contó con la participación de una pequeña organización desarrolladora de software del Estado de Oaxaca, que por razones de confidencialidad será denominada O1 a partir de ahora, cuyas características principales se resumen en la Tabla 44. O1 se ha dedicado al desarrollo de software a la medida, consultoría en TI, identidad gráfica y consultoría integral empresarial. Además, ha sido proveedora de servicios de desarrollo de software y soporte en TI para otras organizaciones y entidades educativas. Por otro lado, entre los productos que tiene en venta destacan un sistema de punto de venta, sistema administrador de agua potable y alcantarillado, y un gestor de eventos académicos y profesionales, los cuales se pueden adecuar a las necesidades y requerimientos de múltiples clientes.

Tabla 44. Características de la organización participante.

Denominación	Tamaño de personal	Metodología usada en los proyectos	Actividad principal
O1	21	Scrum	Desarrollo de software a la medida, consultoría.

Con relación a su experiencia previa, O1 afirmó lo siguiente:

- Como ya se indicó, se ha enfocado principalmente en desarrollar software a la medida con el objetivo de proporcionar soluciones tecnológicas al mercado estatal. Sin embargo, es demasiado frecuente que los problemas en sus estimaciones les impidan tomar y concluir exitosamente una buena proporción de los proyectos aceptados.
- La organización utiliza Scrum para la gestión de los proyectos.
- Además de seguir la metodología ágil, la organización ha incorporado, con cierto esfuerzo, algunas técnicas recomendadas por las metodologías ágiles. En este sentido, los argumentos proporcionados por el personal muestran que hace falta experiencia para entender y aplicar con éxito dichas herramientas en el trabajo cotidiano.

Por otro lado, O1 está conformada por dos áreas principales:

- Área técnica (Desarrollo de software): Se encarga de desarrollar todos los proyectos de desarrollo de software que son solicitados por los clientes. Esta área gestiona y controla todos los aspectos técnicos que el personal realiza para concluir exitosamente todos los proyectos aprobados, incluyendo las siguientes subáreas:
 - Diseño (UX/UI): Responsable del diseño de interfaces y de la experiencia de usuario, asegurando que los productos desarrollados sean funcionales, intuitivos y alineados con las necesidades del cliente.

- Equipo de desarrollo: Responsable de realizar el levantamiento de requerimientos, análisis, codificación, construcción de pruebas y entregas del desarrollo de acuerdo con la metodología y las buenas prácticas.
- Administrador de sistemas: Responsable de monitorear y dar solución a problemáticas que se presenten en configuración, soporte, conexiones de red y funcionamiento de los equipos de cómputo.
- Aseguramiento de la calidad: Encargado de verificar que los productos desarrollados cumplan con los estándares de calidad establecidos mediante actividades de validación, pruebas funcionales y seguimiento de incidencias detectadas durante el proceso de desarrollo.
- Procesos BPMN: Responsable del análisis, diseño y definición de modelos de negocios (e.g., financieros, recursos humanos, administrativos). Además, proporciona seguimiento y acompañamiento a clientes y proyectos a través de medios digitales.
- Área administrativa: Se encarga de la dirección de la organización a nivel administrativo. Esta área es controlada por una directora relacionada con el área de Administración de Empresas, quien establece los lineamientos organizacionales a seguir.

De los 21 empleados con los que cuenta O1, el caso de estudio contó con la participación directa de 12, cuyos roles Scrum y años de experiencia son mostrados en la Tabla 45.

Tabla 45. Perfiles del personal participante.

Cantidad	Rol	Tiempo promedio de experiencia en la industria	Tiempo promedio de experiencia en la organización
1	Scrum Master (SM)	9 años	7 años
1	Product Owner (PO)	2 años y seis meses	2 meses
2	Líderes de Desarrollo (LD)	10 años	8 años
6	Equipo de Desarrollo (ED)	3 años y seis meses	1 año y tres meses
2	Aseguramiento de la calidad (AC)	3 años	1 año

Aunado a lo anterior, la tesista participó en el diseño, ejecución y recolección de información en el caso de estudio. Esto incluyó la capacitación al personal sobre los factores que influyen en la estimación de los proyectos de software, el uso de la aplicación web, y la recogida e interpretación de los datos resultantes. Cabe destacar que esta labor fue importante en el caso de estudio, puesto que la tesista tuvo que analizar los datos históricos de los proyectos de software ya finalizados por la organización, los cuales condujeron a la obtención de una estimación de esfuerzo (i.e., línea base), para posteriormente introducirlos a la aplicación web con el fin de generar sus propias predicciones de esfuerzo y, como consiguiente, facilitar la comparación directa.

Finalmente, el director y codirector de tesis participaron en el caso de estudio asegurándose en todo momento que el aspecto metodológico de la puesta en marcha fuera el apropiado.

4.6.2. Instrumentos

La evaluación empírica consistió en recoger y analizar información proveniente de tres fuentes: (1) la comparación directa entre las estimaciones realizadas de forma tradicional contra las estimaciones obtenidas a través del uso de la aplicación web, (2) la obtención de métricas cuantitativas que permiten determinar la eficiencia del nuevo método de estimación de esfuerzo, y (3) la valoración subjetiva tanto de Scrum Masters como de Líderes de Desarrollo sobre los beneficios del método alternativo de estimación y las estimaciones obtenidas con la aplicación web.

En este sentido, se utilizaron las siguientes métricas durante la experimentación:

- MAE (Error absoluto medio): Es el promedio de los errores absolutos entre la estimación y el valor real. Trata a todos los errores por igual, sin importar si fueron por sobreestimación o subestimación. Mientras menor sea el MAE, mejor.
- RMSE (Raíz del error cuadrático medio): Es la raíz cuadrada del promedio de los errores al cuadrado. A diferencia del MAE, el RMSE penaliza severamente los errores grandes. Mientras menor sea el RMSE, mejor. Además, si el RMSE es mucho más alto que el MAE, significa que el método comete errores puntuales muy graves.
- MMRE: Es una métrica porcentual que calcula el tamaño del error en proporción al tamaño real de la tarea. En la industria de software, un $MMRE \leq 0.25$ (un error promedio igual o menor al 25%) se considera un modelo de estimación altamente aceptable y competitivo.
- PRED: Es el porcentaje de tareas o proyectos cuyas estimaciones caen dentro de un rango aceptable de error determinado por el valor q . Generalmente en software se utiliza PRED (0.25). Mientras mayor sea el PRED, mejor. Un estándar bastante aceptado en la Ingeniería de Software es buscar que el $PRED(0.25) \geq 0.75$ (es decir, que al menos el 75% de las estimaciones tengan un error menor al 25%).

Por otro lado, se diseñó un cuestionario de evaluación con el objetivo de conocer las percepciones y experiencia del Scrum Master y los Líderes de Desarrollo respecto al uso del nuevo método de estimación basado en aprendizaje computacional en comparación con la técnica tradicional de póker de planificación, evaluando el impacto en el tiempo, la dedicación del equipo y la reducción de errores. Dicho cuestionario utiliza una escala tipo Likert con cinco opciones y se compuso de cuatro categorías que enmarcan un total de 12 ítems (véase Tabla 46). Dicha escala tuvo las siguientes alternativas de respuesta: “Totalmente de acuerdo” (TA = 2), “De acuerdo” (DA = 1), “Neutral” (N = 0), “En desacuerdo” (ED = -1) y “Totalmente en desacuerdo” (TD = -2). Con el fin de convertir la valoración subjetiva a un valor numérico, se calcularon la media (M) y la desviación estándar (Std) para cada ítem del cuestionario. Las categorías del cuestionario se definieron de la siguiente manera:

- Eficiencia en el tiempo: Mide el impacto del nuevo método de estimación en la agenda y el cronograma del equipo. Evalúa si la automatización reduce la duración de las ceremonias de la metodología Scrum. Los ítems relacionados con esta categoría validan si la tecnología realmente está devolviendo tiempo productivo al equipo.
- Dedicación y carga de trabajo: Evalúa el desgaste cognitivo, la energía mental y el tipo de esfuerzo que el equipo invierte durante el proceso de estimación. No se busca eliminar el criterio humano, sino cambiar el foco: que el equipo dedique su energía a entender el problema técnico y los riesgos, delegando el cálculo matemático del esfuerzo al modelo de aprendizaje computacional.

- Precisión y reducción de errores: Analiza la calidad de las estimaciones desde una perspectiva de control de daños. Mide la capacidad del modelo para neutralizar los errores humanos y los vicios de la dinámica grupal. Básicamente, los ítems de esta categoría miden si los Scrum Master perciben que los *sprints* ahora se planifican de forma más realista, reduciendo las historias que se quedan “a medias”.
- Adopción y confianza en general: Mide el nivel de aceptación cultural, la credibilidad del algoritmo y la resistencia al cambio dentro del equipo. En resumen, los ítems de esta categoría detectan si los Scrum Master y sus equipos ven al método de estimación como un aliado legítimo o si lo ven con escepticismo.

En aquellos casos donde surgieron discrepancias o fue necesario ampliar las respuestas de las o los participantes, la tesista realizó entrevistas individuales no estructuradas.

4.6.3. Diseño experimental y proceso

La evaluación empírica de este caso de estudio siguió un diseño cuasi-experimental, puesto que fue necesario que las y los participantes tuvieran perfiles específicos, dejando a un lado la selección aleatoria de personas. Asimismo, se recogió tanto información cuantitativa como cualitativa, por lo que el caso de estudio siguió un enfoque mixto. El proceso establecido por la estrategia puede resumirse a través de los siguientes pasos:

Tabla 46. Cuestionario para recoger las percepciones de las y los participantes en el caso de estudio.

Categoría	Ítem	TD	ED	N	DA	TA	M	Std
Eficiencia en tiempo	I1. La aplicación web reduce significativamente el tiempo total que se dedica a las reuniones de estimación en comparación con el póker de planificación.							
	I2. La velocidad para obtener una estimación inicial con la aplicación web permite agilizar la planificación del <i>sprint</i> .							
	I3. Considero que el tiempo invertido en configurar o revisar los datos de la aplicación web es menor que el tiempo que se suele gastar discutiendo cartas en el póker de planificación.							
Dedicación y carga de trabajo	I4. El uso de la aplicación web disminuye el agotamiento o la fatiga mental del equipo durante el proceso de estimación.							
	I5. El nuevo método de estimación permite que el equipo de desarrollo concentre sus esfuerzos en analizar la arquitectura y los criterios de aceptación, en lugar de discutir el “puntaje” numérico.							
	I6. La dedicación requerida por mi parte para facilitar el proceso de estimación es menor o más fluida con la aplicación web.							
	I7. Percibo que las estimaciones sugeridas por la aplicación web son realistas y ajustadas a la							

Categoría	Ítem	TD	ED	N	DA	TA	M	Std
Precisión y reducción de errores	capacidad real del equipo que las obtenidas mediante el consenso del póker de planificación.							
	I8. El método de estimación ayuda a mitigar sesgos humanos (como que el equipo vote lo mismo que el desarrollador más Senior) que suelen ocurrir en el póker de planificación.							
	I9. Noté una reducción en los errores de estimación (e.g., historias que se quedan sin terminar en el <i>sprint</i> por estar subestimadas) cuando se usó la aplicación web.							
Adopción y confianza en general	I10. Siento confianza en los criterios históricos y variables que utiliza el método de estimación para calcular el esfuerzo.							
	I11. Creo que el equipo de desarrollo se siente cómodo adoptando esta nueva técnica impulsada por inteligencia artificial en lugar del debate tradicional.							
	I12. Recomendaría mantener la aplicación web de forma permanente para la estimación de esfuerzo en los proyectos de la organización.							

1. Se inició con la capacitación presencial sobre los factores involucrados en la estimación de esfuerzo en los proyectos de software. En este sentido, una vez que los factores fueron explicados, la tesista pidió que las y los participantes ordenaran la lista de factores principales por orden de prioridad, considerando su experiencia y los proyectos realizados hasta ese momento.
2. Posteriormente, se le solicitó al Product Owner y al Scrum Master que proporcionaran la información relacionada con los *sprints* llevados a cabo en un proyecto (i.e., datos históricos), así como la información relacionada con la velocidad del equipo.
3. Con la intención de evitar contratiempos con la nueva tecnología, se procedió a realizar sesiones de uso con la aplicación web diseñada para generar las predicciones de esfuerzo. En este sentido, las y los participantes se relacionaron con el uso de la herramienta a través de ejercicios hipotéticos que buscaron relacionar rápidamente al Scrum Master con sus principales funcionalidades.
4. Una vez que las funcionalidades fueron comprendidas, se procedió a una segunda capacitación para que las y los participantes entendieran, principalmente, la importancia de configurar correctamente los parámetros de la estimación, así como la interpretación de las estimaciones generadas.
5. La capacitación terminó con la realización de ejercicios prácticos en los cuales las y los participantes realizaron estimaciones bajo la supervisión de la tesista, con el objetivo de corregir errores de interpretación. Por ejemplo, algunos participantes intentaban estimar por tareas, otros introducían información incorrecta, o incluso realizaban una interpretación incorrecta de la predicción resultante.
6. Con la capacitación terminada, se desplegó la aplicación web para que la organización realizara pruebas e iniciara con las estimaciones de esfuerzo considerando los *sprints* y los proyectos. De hecho, como parte de las pruebas realizadas, el Scrum Master realizó 10 *sprints*

a través de la aplicación web y generó, con la ayuda del Equipo de Desarrollo, estimaciones manuales a través de la técnica de póker de planificación. Una vez terminadas, introdujo la información a la aplicación web y generó las estimaciones a través de la predicción realizada por el modelo EH7. A través de este paso se generaron las condiciones para realizar una comparación directa entre las estimaciones de esfuerzo.

7. Las pruebas permitieron detectar pequeños errores en el despliegue en pantalla de la información resultante y la generación de los reportes, principalmente, los cuales fueron corregidos en la versión final de la aplicación web.
8. Aunado a esto, con la información generada se determinaron pequeñas variaciones entre las estimaciones, derivadas de errores cometidos al introducir historias de usuario diferentes, para el caso de la aplicación web. No obstante, este resultado proporcionó evidencia objetiva de que la estimación, además de realizarse por *sprints* y proyectos, también podría hacerse por historias de usuario. Una funcionalidad que se considera como trabajo futuro.
9. Finalmente, se procedió con la estimación de dos proyectos: un proyecto de complejidad media que ya había sido concluido y otro proyecto corto que estaba siendo realizado en ese momento por la organización. Los resultados obtenidos se muestran en la siguiente sección.

4.6.4. Hallazgos

Como ya se ha indicado anteriormente, en el contexto de una tesis de posgrado, la evaluación de un método es el núcleo que demuestra el rigor científico y la validez de la propuesta que se hace. En este contexto, la evaluación empírica siguió un enfoque mixto para analizar el método de estimación desde dos dimensiones indispensables en la Ingeniería de Software: la precisión del modelo escogido y la utilidad en el mundo real. En consecuencia, se presentan a continuación los hallazgos tanto cuantitativos como cualitativos.

4.6.4.1. Hallazgos cuantitativos

En esta sección se presentan las estimaciones generadas para dos proyectos: el Proyecto 1 (P1) que fue desarrollado en 11 *sprints* de 4 semanas cada uno, esto debido a que dentro del ciclo de implementación se contemplan diversas actividades para el equipo Scrum; y para el Proyecto 2 (P2) que se desarrolló en 8 *sprints*, también de 4 semanas cada uno. Como se mencionó, P1 estuvo conformado por 11 *sprints* ejecutados entre mayo de 2025 y mayo de 2026 (véase Tabla 47). Para cada *sprint* se registró el número de participantes, las incidencias iniciales, incidencias finales, el compromiso inicial y el compromiso concluido, expresados en SP, así como la estimación generada por la aplicación web (AW) con el modelo EH7. Estos datos constituyeron la base para analizar el comportamiento del método de estimación durante el desarrollo del proyecto. En términos generales, se observa que el tamaño del equipo, el número de incidencias y el compromiso asumido presentaron variaciones a lo largo de los distintos *sprints*, dando lugar a escenarios con diferentes niveles de complejidad. Además, debido a la importancia de las incidencias, estas cambiaban conforme se desarrollaban algunos *sprints*, haciendo que fuera necesario estimar una segunda vez durante el *sprint*. En particular, el número de participantes osciló entre dos y cinco desarrolladores, mientras que el compromiso inicial varió entre 9 y 50 SP. Asimismo, las incidencias finales mostraron fluctuaciones considerables, alcanzando un máximo de 28 incidencias en el Sprint 5.

Tabla 47. Estimación de *sprints* de P1 utilizando el modelo EH7.

	P1 2025 Sprint 1 (20/mayo/2025 – 16/junio/2025)					
Participantes	Incidencias iniciales	Incidencias finales	Compromiso inicial (SP)	SP Inicial (AW)	Compromiso terminado (SP)	SP Final (AW)
D1	3	0	17	-	0	-
D2	4	8	9	-	15	-
Total	7	8	26	23	15	17
	P1 2025 Sprint 2 (24/junio/2025 – 22/julio/2025)					
Participantes	Incidencias iniciales	Incidencias finales	Compromiso inicial (SP)	SP Inicial (AW)	Compromiso terminado (SP)	SP Final (AW)
D1	3	4	15	-	15	-
D2	4	6	11	-	15	-
Total	7	10	26	27	30	33
	P1 2025 Sprint 3 (11/agosto/2025 – 09/septiembre/2025)					
Participantes	Incidencias iniciales	Incidencias finales	Compromiso inicial (SP)	SP Inicial (AW)	Compromiso terminado (SP)	SP Final (AW)
D1	6	8	15	-	17	-
D2	4	7	10	-	13	-
Total	10	15	25	28	30	32
	P1 2025 Sprint 4 (12/septiembre/2025 – 10/octubre/2025)					
Participantes	Incidencias iniciales	Incidencias finales	Compromiso inicial (SP)	SP Inicial (AW)	Compromiso terminado (SP)	SP Final (AW)
D1	6	8	18	-	20	-
D2	4	4	7	-	7	-
Total	10	12	25	23	27	28
	P1 2025 Sprint 5 (16/octubre/2025 – 13/noviembre/2025)					
Participantes	Incidencias iniciales	Incidencias finales	Compromiso inicial (SP)	SP Inicial (AW)	Compromiso terminado (SP)	SP Final (AW)
D1	5	5	15	-	15	-
D2	3	6	6	-	11	-
D3	2	4	6	-	11	-
D4	5	13	14	-	31	-
Total	15	28	41	43	68	65
	P1 2026 Sprint 6 (08/enero/2026 – 24/enero/2026)					

Participantes	Incidencias iniciales	Incidencias finales	Compromiso inicial (SP)	SP Inicial (AW)	Compromiso terminado (SP)	SP Final (AW)
D1	1	2	3	-	5	-
D2	0	1	0	-	5	-
D3	1	2	3	-	3	-
D4	0	0	0	-	0	-
D5	1	2	3	-	5	-
Total	3	7	9	7	18	21
P1 2026 Sprint 7 (27/enero/2026 – 09/febrero/2026)						
Participantes	Incidencias iniciales	Incidencias finales	Compromiso inicial (SP)	SP Inicial (AW)	Compromiso terminado (SP)	SP Final (AW)
D1	2	3	5	-	11	-
D2	1	2	3	-	8	-
D3	2	3	5	-	11	-
D4	1	1	3	-	3	-
Total	6	9	16	18	33	35
P1 2026 Sprint 8 (19/febrero/2026 – 13/marzo/2026)						
Participantes	Incidencias iniciales	Incidencias finales	Compromiso inicial (SP)	SP Inicial (AW)	Compromiso terminado (SP)	SP Final (AW)
D1	6	4	18	-	14	-
D2	4	2	13	-	5	-
D3	6	4	15	-	11	-
D4	7	3	18	-	5	-
Total	23	13	64	64	35	37
P1 2026 Sprint 9 (13/marzo/2026 – 07/abril/2026)						
Participantes	Incidencias iniciales	Incidencias finales	Compromiso inicial (SP)	SP Inicial (AW)	Compromiso terminado (SP)	SP Final (AW)
D1	6	6	16	-	16	-
D2	1	1	3	-	3	-
D3	0	0	0	-	0	-
Total	7	7	19	21	19	21
P1 2026 Sprint 10 (09/abril/2026 – 21/abril/2026)						
Participantes	Incidencias iniciales	Incidencias finales	Compromiso inicial (SP)	SP Inicial (AW)	Compromiso terminado (SP)	SP Final (AW)
D1	2	3	6	-	9	-

D2	2	2	6	-	6	-
Total	4	5	12	10	15	16
P1 2026 Sprint 11 (24/abril/2026 – 19/mayo/2026)						
Participantes	Incidencias iniciales	Incidencias finales	Compromiso inicial (SP)	SP Inicial (AW)	Compromiso terminado (SP)	SP Final (AW)
D1	2	2	6	-	6	-
D2	1	1	5	-	5	-
D3	5	7	12	-	17	-
Total	8	10	23	25	28	31

La comparación entre las estimaciones efectivamente obtenidas para P1 (véase columna de Compromiso Terminado (SP) en la Tabla 47) y aquellas producidas por la aplicación web (véase columna SP Final (AW) en la misma tabla) arrojó resultados que muestran un buen funcionamiento del modelo en relación con las tendencias reales de los diferentes periodos de *sprint*. En general, las estimaciones de la aplicación estuvieron acorde con lo finalmente logrado, lo que demuestra la eficacia del modelo para absorber las fluctuaciones del proyecto. Además, resulta evidente que los cambios más pronunciados entre las estimaciones iniciales y las reales se vinculan al aumento en la incidencia de problemas y las diferencias en la naturaleza del trabajo. Es decir, durante la implementación del *sprint*, en comparación con los periodos que experimentaron menor variabilidad, donde las predicciones se mantuvieron muy cercanas al nivel comprometido. Por lo tanto, esta información muestra que la aplicación desarrollada ofrece un soporte útil para la organización y seguimiento del esfuerzo en contextos Scrum.

Otro punto que resaltar es que los resultados muestran que, en varios sprints, la columna SP Inicial (AW) presentó diferencias respecto a la columna Compromiso terminado (SP); sin embargo, la columna SP Final (AW) mostró una mayor aproximación al esfuerzo realmente completado en la mayoría de las iteraciones. Este comportamiento es particularmente evidente en los *sprints* 1, 4, 5, 7, 8 y 10, donde la estimación final se aproximó al valor registrado en la columna de Compromiso terminado (SP). En contraste, en los *sprints* 2, 3, 9 y 11 la estimación inicial ya se encontraba cercana al valor real, por lo que las diferencias entre las columnas SP Inicial (AW) y SP Final (AW) fueron reducidas.

De manera similar, en la Tabla 48 se presentan las estimaciones obtenidas para los ocho *sprints* correspondientes a P2 utilizando el modelo EH7. En esta se muestran las incidencias registradas por cada integrante, el compromiso inicial expresado en puntos de historia, las estimaciones iniciales generadas por la aplicación web, el compromiso realmente completado al finalizar cada *sprint* y la estimación final proporcionada por el modelo. La comparación entre las estimaciones generadas por la aplicación y el esfuerzo realmente desarrollado evidencia un comportamiento consistente del modelo EH7 a lo largo de los ocho *sprints*. En términos generales, las estimaciones iniciales producidas por la aplicación se mantuvieron próximas al compromiso alcanzado por el equipo y, conforme avanzó la ejecución de cada *sprint*, las estimaciones finales lograron aproximarse aún más a los puntos de historia efectivamente completados.

Tabla 48. Estimación de *sprints* de P2 utilizando el modelo EH7.

P2 2025 <i>Sprint</i> 1 (21/abril/2025 – 19/mayo/2025)						
Participantes	Incidencias iniciales	Incidencias finales	Compromiso inicial (SP)	SP Inicial (AW)	Compromiso terminado (SP)	SP Final (AW)
D1	3	3	5	-	5	-
D2	7	8	9	-	14	-
D3	5	5	6	-	6	-
Total	15	16	20	17	25	25
P2 2025 <i>Sprint</i> 2 (20/mayo/2025 – 17/junio/2025)						
Participantes	Incidencias iniciales	Incidencias finales	Compromiso inicial (SP)	SP Inicial (AW)	Compromiso terminado (SP)	SP Final (AW)
D1	3	3	8	-	8	-
D2	3	4	7	-	12	-
D3	2	4	6	-	12	-
Total	8	11	21	23	32	29
P2 2025 <i>Sprint</i> 3 (18/junio/2025 – 11/julio/2025)						
Participantes	Incidencias iniciales	Incidencias finales	Compromiso inicial (SP)	SP Inicial (AW)	Compromiso terminado (SP)	SP Final (AW)
D1	4	5	11	-	14	-
D2	5	5	15	-	15	-
Total	9	10	26	25	29	30
P2 2025 <i>Sprint</i> 4 (30/julio/2025 – 27/agosto/2025)						
Participantes	Incidencias iniciales	Incidencias finales	Compromiso inicial (SP)	SP Inicial (AW)	Compromiso terminado (SP)	SP Final (AW)
D1	5	7	15	-	18	-
D2	4	4	7	-	7	-
Total	9	11	22	23	25	26
P2 2025 <i>Sprint</i> 5 (28/agosto/2025 – 25/septiembre/2025)						
Participantes	Incidencias iniciales	Incidencias finales	Compromiso inicial (SP)	SP Inicial (AW)	Compromiso terminado (SP)	SP Final (AW)
D1	5	7	11	-	17	-
D2	4	5	8	-	13	-
D3	5	5	13	-	13	-
D4	5	8	11	-	20	-
Total	19	25	43	43	63	65

P2 2026 <i>Sprint</i> 6 (26/septiembre/2025 – 24/octubre/2025)						
Participantes	Incidencias iniciales	Incidencias finales	Compromiso inicial (SP)	SP Inicial (AW)	Compromiso terminado (SP)	SP Final (AW)
D1	3	2	5	-	4	-
D2	2	3	6	-	9	-
D3	1	2	3	-	6	-
D4	1	3	3	-	9	-
Total	7	10	17	18	28	28
P2 2026 <i>Sprint</i> 7 (27/octubre/2025 – 18/diciembre/2025)						
Participantes	Incidencias iniciales	Incidencias finales	Compromiso inicial (SP)	SP Inicial (AW)	Compromiso terminado (SP)	SP Final (AW)
D1	2	3	5	-	8	-
D2	1	2	1	-	4	-
D3	2	3	4	-	9	-
D4	1	1	3	-	3	-
Total	6	9	13	14	24	25
P2 2026 <i>Sprint</i> 8 (8/enero/2026 – 5/febrero/2026)						
Participantes	Incidencias iniciales	Incidencias finales	Compromiso inicial (SP)	SP Inicial (AW)	Compromiso terminado (SP)	SP Final (AW)
D1	5	8	10	-	17	-
D2	3	5	8	-	15	-
D3	5	6	11	-	14	-
D4	7	8	17	-	20	-
Total	20	27	46	49	66	68

Los resultados para P2 mostraron que, en la mayoría de los *sprints*, la diferencia entre la estimación final de la aplicación y el esfuerzo real fue reducida. En los *sprints* 1 y 6 la estimación final coincidió exactamente con el compromiso terminado, mientras que en los *sprints* 3, 4, 7 y 8 la diferencia fue de únicamente un punto de historia. Asimismo, en el *sprint* 2 la estimación final presentó una diferencia de tres puntos de historia respecto al valor real, mientras que el *sprint* 5 registró la mayor diferencia, con dos puntos de historia adicionales sobre el compromiso concluido, comportamiento que puede atribuirse al incremento en el número de incidencias y al aumento de la carga de trabajo durante dicho periodo.

En términos generales, los resultados obtenidos en P1 y P2 evidenciaron que el modelo EH7 fue capaz de generar estimaciones consistentes y cercanas al esfuerzo realmente desarrollado durante los *sprints*. Aunque en algunas iteraciones se presentaron diferencias importantes entre el compromiso inicial y el compromiso concluido, principalmente debido al incremento de incidencias, cambios en

el alcance o variaciones en la productividad del equipo, las estimaciones finales producidas por la aplicación lograron reducir dichas diferencias y aproximarse con mayor precisión al comportamiento real de cada *sprint*.

Aunado a esto, la comparación entre los proyectos P1 y P2 mostró que el modelo mantiene un desempeño estable, aun cuando las condiciones de ejecución son diferentes. En P2, por ejemplo, se observaron estimaciones finales con menor variabilidad y una mayor cercanía al compromiso realmente completado, mientras que en P1, el modelo presentó estimaciones confiables, pero con una ligera mayor diferencia con el compromiso terminado, lo cual se puede atribuir a *sprints* con mayor incertidumbre. Estos resultados demuestran que el modelo EH7 aprendió de los diferentes escenarios de desarrollo con los que fue entrenado, por lo que contiene información útil para apoyar la planificación y el seguimiento de esfuerzo en proyectos de Scrum.

4.6.4.2. Hallazgos cualitativos

Aunado a lo anterior, la aplicación del cuestionario proporcionó información importante que reflejó la percepción de las y los participantes en el caso de estudio sobre el método de estimación y, en consecuencia, de la aplicación web (véase Tabla 49). En dicha tabla TD corresponde a “Totalmente en Desacuerdo”, ED a “En Desacuerdo”, N a “Neutral”, DA a “De Acuerdo” y TA a “Totalmente de Acuerdo”. Estos valores representan la frecuencia de respuestas obtenidas de los doce participantes para cada ítem evaluado. Asimismo, M corresponde a la media de las respuestas y Std a la desviación estándar, la cual permite medir el grado de consenso entre las y los participantes.

Como se podrá notar, las percepciones de las y los participantes fue buena en general, puesto que la gran mayoría de valoraciones se mantienen del lado positivo de la escala de Likert. En este sentido, la categoría mejor evaluada fue la de “Eficiencia en tiempo” puesto que las y los participantes consideraron que el método de estimación propuesto reduce considerablemente el tiempo dedicado a realizar las ceremonias de estimación en Scrum, contribuyendo de esta manera a “agilizar” el desarrollo del proyecto. De hecho, expresaron que la aplicación web puede ayudar a la organización a potenciar el factor humano. Esta percepción general es correcta, puesto que, en una pequeña organización desarrolladora de software, donde los recursos son limitados y cada error de estimación cuesta caro, el método basado en aprendizaje computacional podría ser un salto significativo de calidad.

Tabla 49. Percepciones de las y los participantes en el caso de estudio.

Categoría	Ítem	TD	ED	N	DA	TA	M	Std
Eficiencia en tiempo	I1. La aplicación web reduce significativamente el tiempo total que se dedica a las reuniones de estimación en comparación con el póker de planificación.	0.0	0.0	0.0	4.0	8.0	1.7	0.5
	I2. La velocidad para obtener una estimación inicial con la aplicación web permite agilizar la planificación del <i>sprint</i> .	0.0	0.0	0.0	0.0	12.0	2.0	0.0
	I3. Considero que el tiempo invertido en configurar o revisar los datos de la aplicación web es menor que el tiempo que se suele gastar discutiendo cartas en el póker de planificación.	0.0	0.0	0.0	0.0	12.0	2.0	0.0

Categoría	Ítem	TD	ED	N	DA	TA	M	Std
Dedicación y carga de trabajo	I4. El uso de la aplicación web disminuye el agotamiento o la fatiga mental del equipo durante el proceso de estimación.	0.0	0.0	1.0	4.0	7.0	1.5	0.6
	I5. El nuevo método de estimación permite que el equipo de desarrollo concentre sus esfuerzos en analizar la arquitectura y los criterios de aceptación, en lugar de discutir el “puntaje” numérico.	0.0	0.0	2.0	7.0	3.0	1.1	0.6
	I6. La dedicación requerida por mi parte para facilitar el proceso de estimación es menor o más fluida con la aplicación web.	0.0	0.0	0.0	1.0	11.0	1.9	0.3
Precisión y reducción de errores	I7. Percibo que las estimaciones sugeridas por la aplicación web son realistas y ajustadas a la capacidad real del equipo que las obtenidas mediante el consenso del póker de planificación.	0.0	0.0	0.0	10.0	2.0	1.2	0.4
	I8. El método de estimación ayuda a mitigar sesgos humanos (como que el equipo vote lo mismo que el desarrollador más Senior) que suelen ocurrir en el póker de planificación.	0.0	0.0	0.0	0.0	12.0	2.0	0.0
	I9. Noté una reducción en los errores de estimación (e.g., historias que se quedan sin terminar en el <i>sprint</i> por estar subestimadas) cuando se usó la aplicación web.	0.0	0.0	0.0	9.0	3.0	1.3	0.4
Adopción y confianza en general	I10. Siento confianza en los criterios históricos y variables que utiliza el método de estimación para calcular el esfuerzo.	0.0	0.0	0.0	10.0	2.0	1.2	0.4
	I11. Creo que el equipo de desarrollo se siente cómodo adoptando esta nueva técnica impulsada por inteligencia artificial en lugar del debate tradicional.	0.0	0.0	0.0	6.0	6.0	1.5	0.5
	I12. Recomendaría mantener la aplicación web de forma permanente para la estimación de esfuerzo en los proyectos de la organización.	0.0	0.0	0.0	0.0	12.0	2.0	0.0

Si bien el póker de planificación, técnica utilizada por la organización participante, fomenta la discusión en equipo, se sufren sesgos cognitivos como el anclaje (i.e., dejarse llevar por la primera opinión) o el sesgo de optimismo (i.e., pensar que todo saldrá más rápido de lo que realmente toma). Por lo contrario, el método de estimación presentado a la organización puede analizar objetivamente los datos de los proyectos pasados (tiempo real tomado, complejidad de las tareas, experiencia de los desarrolladores y demás factores relacionados) y ofrece una estimación basada en la realidad histórica de la organización. Aunado a esto, a las y los participantes en el caso de estudio les pareció excelente la idea de que la aplicación actúe como un repositorio del “conocimiento histórico” de la organización. Es decir, aunque el equipo cambie, los datos de lo que costó construir el software en el pasado siguen ahí, entrenando al algoritmo y suavizando la curva de aprendizaje de las y los nuevos integrantes.

Por otro lado, la categoría menos valorada fue la de “Dedicación y carga de trabajo” puesto que uno de los participantes permaneció neutral al responder el ítem “I4. El uso de la aplicación web disminuye el agotamiento o la fatiga mental del equipo durante el proceso de estimación”. Con la intención de determinar el origen de esta percepción, la tesista procedió a recoger información

adicional mediante una entrevista y se pudo determinar que el Product Owner participante no estaba del todo convencido sobre la afirmación que se hace en dicho ítem, puesto que solamente tiene cuatro meses trabajando para la organización.

Si bien reconoce que, de acuerdo con su experiencia pasada, en este tipo de organizaciones las y los integrantes del Equipo de desarrollo suelen a menudo asumir múltiples roles e, incluso, pasar horas en sesiones de póker de planificación que resultan agotadoras y costosas, no está seguro que el método de estimación, por sí solo, sea capaz de estimar el esfuerzo de forma automatizada y sugiere seguir utilizando la técnica de póker de planificación para validar en aquellos casos donde el modelo de aprendizaje tenga un bajo nivel de confianza o donde haya una incertidumbre técnica considerable.

Aunado a esto, dos participantes no estuvieron convencidos al responder el ítem “I5. El nuevo método de estimación permite que el equipo de desarrollo concentre sus esfuerzos en analizar la arquitectura y los criterios de aceptación, en lugar de discutir el “puntaje” numérico”. Al indagar sobre esta percepción, se pudo comprobar que se trataba del mismo Product Owner y un integrante del Equipo de desarrollo que no tiene experiencia previa en la industria y que ha desempeñado el rol de programador únicamente por tres meses en la organización. De hecho, este participante argumentó no estar relacionado con la técnica de póker de planificación.

4.6.5. Cierre del caso de estudio: Comprobación de la hipótesis de la tesis

Considerando la evidencia mostrada en las secciones 4.7.1 y 4.7.2, principalmente los resultados cuantitativos, se concluye que la hipótesis establecida al inicio de esta tesis de la siguiente forma:

Hi: “A través de un método de estimación que haga uso del aprendizaje computacional, se podrá determinar el esfuerzo requerido para desarrollar proyectos con Scrum en una pequeña organización desarrolladora de software”.

puede ser aceptada considerando los siguientes argumentos:

1. Los bajos valores obtenidos en el MAE mostraron que el modelo de aprendizaje computacional es capaz de modelar el comportamiento del equipo y predecir el esfuerzo con desviaciones mínimas respecto a la realidad. En una pequeña organización desarrolladora de software, un error de estimación drástico puede desestabilizar financieramente un proyecto. El hecho de que el RMSE se mantenga controlado demuestra que el modelo no solo es preciso en promedio, sino que elimina los errores catastróficos (i.e., desviaciones extremas) a los que el póker de planificación es vulnerable debido a la falta de datos históricos centralizados.
2. Al evaluar el modelo bajo la métrica de MMRE y el nivel de predicción PRED(0.25), los resultados se alinean o superan los umbrales aceptados por la Ingeniería de Software (como el criterio de Conte de $MMRE \leq 0.25$ y $PRED(0.25) \geq 0.75$). En consecuencia, esto demuestra que el aprendizaje computacional no es solo una alternativa teórica, sino un método empíricamente confiable.
3. Scrum depende fuertemente de la velocidad del equipo, la cual suele fluctuar si las estimaciones iniciales son inestables. Al introducir la aplicación web, el proceso de estimación deja de depender de la intuición o del estado de ánimo del equipo en una sesión de póker de planificación. En pequeñas organizaciones, la salida de un desarrollador Senior o la presión del cliente suelen sesgar por completo el póker de planificación, provocando subestimación

por complacencia o sobreestimación por miedo. El modelo de estimación presentado en esta tesis actúa como un “ancla de objetividad” basado en hechos empíricos anteriores, logrando estabilizar las métricas de Scrum (como el *burndown chart* y la velocidad).

4. La hipótesis exigía demostrar si el método funcionaba en una pequeña organización. Los resultados empíricos demuestran que el modelo no necesitó de macrodatos inalcanzables para ser efectivo; los datos históricos de la propia organización fueron suficientes para entrenar un modelo preciso.

En resumen, los datos demuestran de manera concluyente que un método de estimación basado en aprendizaje computacional es capaz de determinar el esfuerzo requerido en proyectos Scrum dentro del contexto de una pequeña organización. El modelo demostró no solo precisión, sino también la capacidad de mitigar los errores y sesgos cognitivos propios de las técnicas tradicionales como el póker de planificación, para fomentar el mejor uso de los recursos limitados de la organización. Sin embargo, es de suma importancia aclarar que los resultados mostrados en este capítulo únicamente permiten validar la hipótesis establecida para el contexto de esta tesis de grado. Es decir, de ninguna manera debe entenderse que la efectividad que demostró el método de estimación y la aplicación web generada puede generalizarse a organizaciones más grandes, o incluso del mismo tamaño, que desarrollen cualquier tipo de proyectos con equipos de cualquier cantidad de integrantes. Es evidente que afirmaciones de este tipo deben sustentarse con más evaluaciones empíricas en diferentes organizaciones participantes y, por ende, quedan fuera de la aportación que se hace con esta tesis de grado.

5. Conclusiones y trabajo futuro

En este proyecto de tesis, se implementaron métodos de estimación del esfuerzo, integrando técnicas de aprendizaje computacional con variables propias del contexto para proyectos Scrum. Como resultado de esta investigación se diseñó e implementó una aplicación web denominada *Effortestimation*, la cual calcula la estimación del esfuerzo y constituye una alternativa a los métodos tradicionales de estimación, como el póker de planificación, proporcionando apoyo durante la planificación de proyectos y *sprints*.

Debido a la falta de conjuntos de datos públicos que integren factores vitales del proyecto, niveles de incertidumbre, puntos de historia y velocidad del equipo, específicamente en pequeñas empresas de desarrollo de software, fue necesario generar conjuntos de datos sintéticos utilizando el Método Generalizado de Estimación Ágil (GEM). El método permitió la aplicación sistemática para capturar escenarios diversos del desarrollo de software a nivel de proyecto y de *sprint*, ofreciendo así una base sólida para apoyar el entrenamiento, validación y prueba de los modelos computacionales de aprendizaje propuestos.

A partir de estos conjuntos de datos, se evaluaron distintos modelos de regresión individuales y ensambles heterogéneos con el propósito de identificar el modelo con mayor capacidad predictiva. Los resultados obtenidos demostraron que el desempeño de los modelos depende de las características del conjunto de datos y del nivel de estimación considerado. Para la estimación de proyectos, el modelo SVR presentó el mejor desempeño, al obtener un MAE de 0.070, un RMSE de 0.093, un MMRE de 0.002 y un PRED de 100%, mostrando un equilibrio entre precisión y robustez en las diferentes configuraciones evaluadas, por lo que fue integrado en la aplicación web para la obtención de las estimaciones de proyectos de software. Por otra parte, para la estimación de *sprints*, los resultados evidenciaron que los modelos de ensamble superan a los modelos individuales. En particular, el ensamble EH7 fue seleccionado como el mejor modelo al alcanzar un MAE de 0.340, un RMSE de 0.623, un MMRE de 0.021 y un PRED de 100%, reflejando una elevada precisión y una adecuada capacidad de generalización en los diferentes escenarios evaluados.

La validación realizada mediante el caso de estudio confirmó que las estimaciones generadas mantienen un comportamiento consistente con la evolución real de los proyectos y *sprints* analizados. En particular, el seguimiento del Proyecto 1 permitió comprobar la influencia que ejercen variables como el tamaño del equipo, las incidencias surgidas durante el desarrollo, las modificaciones al alcance y las variaciones en el compromiso asumido sobre la estimación final del esfuerzo, evidenciando la importancia de considerar a futuro este tipo de factores dentro del proceso de estimación.

Asimismo, los resultados mostraron que el método mantiene un desempeño satisfactorio en escenarios con un comportamiento relativamente estable; sin embargo, su precisión disminuye cuando se presentan cambios importantes en el alcance del *sprint* o cuando se incorporan nuevas historias de

usuario durante su ejecución. Estos escenarios incrementan la incertidumbre y la variabilidad del proyecto, por lo que representan un área de oportunidad para incorporar mecanismos dinámicos de actualización o variables adicionales que permitan mejorar la capacidad predictiva del modelo.

Por otra parte, la evaluación cualitativa evidenció una percepción ampliamente favorable por parte del Scrum Master y del equipo Scrum respecto al método propuesto y a la aplicación web desarrollada. Las respuestas obtenidas mediante el cuestionario indicaron que la aplicación contribuye a reducir el tiempo destinado a las reuniones de estimación, disminuye la carga cognitiva del equipo, ayuda a mitigar los sesgos presentes en técnicas tradicionales como el póker de planificación y genera estimaciones que son percibidas como confiables y acordes con la capacidad real del equipo. Además, la mayoría de las y los participantes manifestó su interés en incorporar permanentemente la herramienta dentro del proceso de estimación de proyectos de la organización.

De igual manera, la experimentación permitió comprobar que la estimación basada en aprendizaje computacional no busca sustituir el conocimiento ni la experiencia del equipo de desarrollo, sino complementarlos mediante el aprovechamiento del conocimiento histórico generado por la organización. En este sentido, la aplicación web constituye una herramienta de apoyo para la toma de decisiones, proporcionando una referencia objetiva que favorece estimaciones más consistentes.

En conjunto, los resultados obtenidos permiten concluir que la propuesta desarrollada constituye una alternativa viable para apoyar la estimación del esfuerzo en proyectos ágiles dentro de pequeñas organizaciones desarrolladoras de software. La integración de GEM con técnicas de aprendizaje computacional permitió generar estimaciones precisas, objetivas y consistentes, además de ofrecer una herramienta tecnológica con potencial para mejorar los procesos de planificación, optimizar la toma de decisiones y contribuir al desarrollo exitoso de proyectos de software bajo el marco de trabajo Scrum.

Como trabajo futuro se propone ampliar la investigación mediante la incorporación de nuevas variables relacionadas con el contexto organizacional, el comportamiento del equipo y las características técnicas de los proyectos. Asimismo, resulta de interés extender la propuesta hacia la estimación automática de historias de usuario, tareas y esfuerzo expresado en horas, además de evaluar nuevas técnicas de aprendizaje computacional y arquitecturas de ensamble que permitan incrementar aún más la precisión de las estimaciones.

6. Bibliografia

Abdelali, Z., Hicham, M., & Abdelwahed, N. (2018, October). An ensemble of optimal trees for software development effort estimation. In *International Conference on Advanced Information Technology, Services and Systems* (pp. 55-68). Cham: Springer International Publishing.

Abdolrasol, M. G. M., Hussain, S. M. S., Ustun, T. S., Sarker, M. R., Hannan, M. A., Mohamed, R., Ali, J. A., Mekhilef, S., & Milad, A. (2021). Artificial neural networks based optimization techniques: A review. *Electronics*, 10(21), 2689.

Agbejule, A., & Lehtineva, L. (2022). The relationship between traditional project management, agile project management and teamwork quality on project success. *International Journal of Organizational Analysis*, 30(7), 124-136.

Agrawal, A., Singh, S., Tripathi, M., & Maurya, L. S. (2014). Implementing agile in small organizations. *International Journal of Information & Computation Technology*, 687-692.

Al Kharusi, A. A. (2020). A systematic mapping study on software effort estimation. *Journal of Theoretical and Applied Information Technology*, 98(17).

Ali, A., & Gravino, C. (2019). A systematic literature review of software effort prediction using machine learning methods. *Journal of Software: Evolution and Process*, 31(10), e2211.

Ali, S. S., Ren, J., Zhang, K., Wu, J., & Liu, C. (2023). Heterogeneous ensemble model to optimize software effort estimation accuracy. *IEEE Access*, 11, 27759-27792.

Almeida, F., Simões, J., & Lopes, S. (2022). Exploring the benefits of combining DevOps and agile. *Future Internet*, 14(2), 63.

Almutlaq, A. J. H., & Jawawi, D. N. (2020). Missing data imputation techniques for software effort estimation: A study of recent issues and challenges. In: *Emerging Trends in Intelligent Computing and Informatics: Data Science, Intelligent Information Systems and Smart Computing 4* (pp. 1144-1158). Springer International Publishing.

Alostad, J. M., Abdullah, L. R., & Aali, L. S. (2017). A fuzzy based model for effort estimation in Scrum projects. *International Journal of Advanced Computer Science and Applications*, 8(9), 270-277.

Alsaadi, B., & Saeedi, K. (2022). Data-driven effort estimation techniques of agile user stories: a systematic literature review. *Artificial Intelligence Review*, 55(7), 5485-5516.

Amirian, E., Abdollahzadeh, A., & Sulaiman, N. (2024, March). Synergizing hybrid agile-Scrum and CRISP-DM approaches in data science project management. In: *SPE Canadian Energy Technology Conference* (p. D021S017R001). SPE.

Arora, M., Verma, S., Kavita, & Chopra, S. (2020). A systematic literature review of machine learning estimation approaches in Scrum projects. In: Mallick, P., Balas, V., Bhoi, A., Chae, G. S. (Eds.), *Cognitive Informatics and Soft Computing. Advances in Intelligent Systems and Computing* (pp. 573-586). Cham: Springer, Singapore.

Arora, M., Verma, S., Kavita, Wozniak, M., Shafi, J., & Ijaz, M. F. (2022). An efficient ANFIS-EEBAT approach to estimate effort of Scrum projects. *Scientific Reports*, 12(1), 7974.

Attar, M., & Abdul-Kareem, A. (2020). The role of agile leadership in organisational agility. In: B. Akkaya (Ed.), *Agile business leadership methods for Industry 4.0* (pp. 171–191). Emerald Publishing Limited.

Avelino, J. G., Cavalcanti, G. D., & Cruz, R. M. (2024). Resampling strategies for imbalanced regression: a survey and empirical analysis. *Artificial Intelligence Review*, 57(4), 82.

Azevedo, A., & Santos, M. F. (2008). KDD, SEMMA and CRISP-DM: A parallel overview. *IADS-DM*.

Azzeh, M., Nassif, A. B., Elsheikh, Y., & Angelis, L. (2022). On the value of project productivity for early effort estimation. *Science of Computer Programming*, 219, 102819.

Babaei, M., Rashidi-baqhi, A., & Rashidi, M. (2022). Estimating project cost under uncertainty using universal generating function method. *Journal of Construction Engineering and Management*, 148(2), 04021194.

Baham, C., & Hirschheim, R. (2022). Issues, challenges, and a proposed theoretical core of agile software development research. *Information Systems Journal*, 32(1), 103-129.

Bahlerao, S., & Ingle, M. (2011). Generalized agile estimation method. *International Journal on Advanced Science, Engineering and Information Technology*, 1(3), 262-267.

Barghoth, M. E., Salah, A., & Ismail, M. A. (2020, November). Software Project Management Effectiveness Evaluation Tool. In: *Proceedings of the 9th International Conference on Software and Information Engineering* (pp. 18-24).

Behrens, A., Ofori, M., Noteboom, C., & Bishop, D. (2021). A systematic literature review: How agile is agile project management? *Issues in Information Systems*, 22(3).

Bhardwaj, V., & Arora, S. (2021). A practical approach for software project management. *The Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, 12(1), 913-918.

Bhattacharyya, A., Srijith, K., Behera, R. P., Dasgupta, A., & Chakraborty, R. S. (2023, September). A study on effects of synthetic data for predicting the remaining useful life of aluminium electrolytic capacitors using bagging-based ensemble learning. In: *International Conference on Advances in Data-driven Computing and Intelligent Systems* (pp. 549-564). Singapore: Springer Nature Singapore.

Bonner, N. A., Kulangara, N., Nerur, S., & Teng, J. T. C. (2021). An empirical investigation of the perceived benefits of agile methodologies using an innovation-theoretical model. In: M. Khosrow-Pour, S. Clarke, M. E. Jennex, & A.-V. Anttiroiko (Eds.), *Research anthology on recent trends, tools, and implications of computer programming* (pp. 202–231). IGI Global.

Branco, P., Torgo, L., & Ribeiro, R. P. (2017, October). SMOGN: a pre-processing approach for imbalanced regression. In *First international workshop on learning with imbalanced domains: Theory and applications* (pp. 36-50). PMLR.

Butt, S. A., Misra, S., Luis, D. M. J., Emiro, D. L. H. F. (2021a). Efficient approaches to agile cost estimation in software industries: a project-based case study. In: *Information and Communication*

Technology and Applications: 3rd International Conference, ICTA 2020, Minna, Nigeria, November 24–27, 2020, Revised Selected Papers 3 (pp. 645-659). Springer International Publishing.

Butt, S. A., Misra, S., Piñeres-Espitia, G., Ariza-Colpas, P., & Sharma, M. M. (2021b). A cost estimating method for agile software development. In: Computational Science and Its Applications—ICCSA 2021: 21st International Conference, Cagliari, Italy, September 13–16, 2021, Proceedings, Part VII 21 (pp. 231-245). Springer International Publishing.

Camacho Sánchez, N. (2023). La importancia de las metodologías ágiles en los proyectos de desarrollo de software [Trabajo de fin de grado, Universidad de Sevilla, Escuela Técnica Superior de Ingeniería]. Repositorio Institucional de la Universidad de Sevilla. <https://idus.us.es/handle/11441/150887>

Canedo, E. D., Aranha, D. P., Cardoso, M. D. O., Da Costa, R.P., & Leite, L. L. (2018, Julio). Métodos para estimar proyectos de software ágiles: Revisión sistemática de la literatura. En: Actas de la Conferencia Internacional sobre Ingeniería de Software e Ingeniería del Conocimiento, SEKE (págs. 34-39).

Cao, L. (2022). Estimating efforts for various activities in agile software development: An empirical study. IEEE Access, 10, 83311-83321.

Carbonera, C. E., Farias, K., & Bischoff, V. (2020). Software development effort estimation: A systematic mapping study. IET Software, 14(4), 328-344.

Cardozzo, D. R. (2016). Desarrollo de software: Requisitos, estimaciones y análisis. IT Campus Academy.

Carvalho, H. D. P., Lima, M. N. C. A., Santos, W. B., & Fagunde, R. A. de A. (2020). Ensemble regression models for software development effort estimation: A comparative study. International Journal of Software Engineering & Applications, 11(3), 71-86.

Charbuty, B., & Abdulazeez, A. (2021). Classification based on decision tree algorithm for machine learning. Journal of Applied Science and Technology Trends, 2(01), 20-28.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. Journal of artificial intelligence research, 16, 321-357.

Choraś, M., Springer, T., Kozik, R., López, L., Martínez-Fernández, S., Ram, P., Rodríguez, P., & Franch, X. (2020). Measuring and improving agile processes in a small-size software development company. IEEE Access, 8, 78452–78466.

Ciancarini, P., Farina, M., Masyagin, S., Succi, G., Yermolaieva, S., & Zagvozkina, N. (2021). Root causes of interaction issues in agile software development teams: Status and perspectives. In: K. Arai (Ed.), Advances in information and communication: Proceedings of the 2021 Future of Information and Communication Conference (FICC), Volume 2 (pp. 1017–1036). Springer International Publishing.

Ciric, D., Lalic, B., Gracanin, D., Tasic, N., Delic, M., & Medic, N. (2019). Agile vs. traditional approach in project management: Strategies, challenges and reasons to introduce agile. Procedia Manufacturing, 39, 1407-1414.

Coelho, E., & Basu, A. (2012). Effort estimation in agile software development using story points. International Journal of Applied Information Systems (IJ AIS), 3(7).

Conoscenti, M., Besner, V., Vetrò, A., & Fernández, D. M. (2019). Combining data analytics and developers feedback for identifying reasons of inaccurate estimations in agile software development. Journal of Systems and Software, 156, 126-135.

Dantas, E., Perkusich, M., Dilenzo, E., Santos, D. F., Almeida, H., & Perkusich, A. (2018). Effort estimation in agile software development: An updated review. *International Journal of Software Engineering and Knowledge Engineering*, 28(11, 12), 1811-1831.

Department of Defense. (2023). Software modernization implementation plan summary. Washington, DC: DoD.

Dunning, D. (2011). The Dunning–Kruger effect: On being ignorant of one's own ignorance. In *Advances in experimental social psychology* (Vol. 44, pp. 247-296). Academic Press.

Durgesh, D. V. S., Saket, M. V. S., & Ramana Reddy, B. (2023, June). Improving software effort estimation with heterogeneous stacked ensemble using SMOTER over ELM and SVR base learners. In: *International Conference on Multi-disciplinary Trends in Artificial Intelligence* (pp. 442-448). Cham: Springer Nature Switzerland.

Dybå, T., Dingsøyr, T., & Moe, N. B. (2014). Agile project management. *Software project management in a changing world*, 277-300.

El Beggar, O. (2024). IFEJM: New intuitionistic fuzzy expert judgment method for effort estimation in agile software development. *Arabian Journal for Science and Engineering*, 49(3), 2887-2908.

Engström, E., & Runeson, P. (2010, June). A qualitative survey of regression testing practices. In *International conference on product focused software process improvement* (pp. 3-16). Berlin, Heidelberg: Springer Berlin Heidelberg.

Erthal, V. M., de Souza, B. P., dos Santos, P. S. M., & Travassos, G. H. (2023). Characterization of continuous experimentation in software engineering: Expressions, models, and strategies. *Science of Computer Programming*, 229, 102961.

Esquirol-Caussa, J., Aldeguez, J. S., & Santamaria, I. D. (2017). La revisión bibliográfica: Base de la investigación. *Actualizaciones en Fisioterapia*, 2(12), 34-37.

Fang, Y. (2021, October). The theory and application in agile project management (APM) with Scrum. In: *Proceedings of the 2021 6th International Conference on Modern Management and Education Technology (MMET 2021)* (pp. 652–657). Atlantis Press.

Farley, D. (2021). *Modern software engineering: Doing what works to build better software faster*. Addison-Wesley Professional.

Fávero, E. M. D. B., Pereira, R., Pimentel, A. R., & Casanova, D. (2018). Analogy-based effort estimation: A systematic mapping of literature. *INFOCOMP Journal of Computer Science*, 17(2), 07-22.

Fernández-Diego, M., Méndez, E. R., González-Ladrón-de-Guevara, F., Abrahão, S., & Insfran, E. (2020). An update on effort estimation in agile software development: A systematic literature review. *IEEE Access*, 8, 166768–166800.

Ferreira Lorenzo, G. L., Gálvez Lío, D., Quintero Domínguez, L. A., & Antón Vargas, J. (2014). Estimación del esfuerzo en proyectos de software utilizando técnicas de Inteligencia Artificial. *Revista Cubana de Ciencias Informáticas*, 8(4), 1-20.

Figueiredo, J. (2023). PMBOK 7th Edition in the Good Direction but how Effective. *Global Journals of Research in Engineering*, 23(J3), 69-75.

Filippetto, A. S., Lima, R., & Barbosa, J. L. V. (2021). A risk prediction model for software project management based on similarity analysis of context histories. *Information and Software Technology*, 131, 106497.

Fitzgerald, B. (2012). Software crisis 2.0. *Computer*, 45(4), 89-91.

- Fürst, L., Hovelja, T., Poženel, M., & Vavpotič, D. (2024). Impact of competence on agile effort estimation in academic setting. *Software: Practice and Experience*, 54(11), 2144-2159.
- Fuentes-Del-Burgo, J., Pérez, S., & Ángel, M. (2022). Comparative analysis of the board tool in the agile methodologies Scrum, Kanban and Scrumban in software projects. In: 26th International Congress on Project Management and Engineering Terrassa.
- Gandomani, T. J., Wei, K. T., & Binhamid, A. K. (2014). A case study research on software cost estimation using experts' estimates, wideband Delphi, and planning poker technique. *International Journal of Software Engineering and its applications*, 8(11), 173-182.
- Govil, N., & Sharma, A. (2022). Estimation of cost and development effort in Scrum-based software projects considering dimensional success factors. *Advances in Engineering Software*, 172, 103209.
- Grebić, B., & Stojanović, A. (2021). Application of the Scrum framework on projects in IT sector. *European Project Management Journal*, 11(2), 37-46.
- Hameed, S., Elsheikh, Y., & Azzeh, M. (2023). An optimized case-based software project effort estimation using genetic algorithm. *Information and Software Technology*, 153, 107088.
- Hamid, M., Zeshan, F., & Ahmad, A. (2021, December). Fuzzy logic-based expert system for effort estimation in scrum projects. In: 2021 International Conference on Decision Aid Sciences and Application (DASA) (pp. 761-765). IEEE.
- Hill, P. R. (2010). International Software Benchmarking Standards Group, Practical Software Project Estimation: A Toolkit for Estimating Software Development Effort & Duration. McGraw-Hill Osborne, 81-82.
- Holgeid, K. K., & Jørgensen, M. (2020). Benefits management and agile practices in software projects: How perceived benefits are impacted. In: 2020 IEEE 22nd Conference on Business Informatics (CBI) (Vol. 2, pp. 48-56). IEEE.
- Hosni, M., Idri, A., Nassif, A. B., & Abran, A. (2016). Heterogeneous ensembles for software development effort estimation. In: 2016 3rd International Conference on Soft Computing & Machine Intelligence (ISCMI), (pp. 174-178). IEEE.
- Hosni, M., Idri, A., Abran, A., & Nassif, A. B. (2018). On the value of parameter tuning in heterogeneous ensembles effort estimation. *Soft Computing*, 22(18), 5977-6010.
- Hosni, M. (2023). On the value of combiners in heterogeneous ensemble effort estimation. In: Proceedings of the 15th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2023), (pp. 153-163). SCITEPRESS.
- Huynh Thai, H., Vo Van, H., & Ho, L. T. K. N. (2021). An approach to adjust effort estimation of function point analysis. *Lecture Notes in Networks and Systems*, 230, 522-537.
- Hynninen, T., & Jantunen, S. (2022, May). Questionnaire approach for assessing software engineering and quality assurance practices. In 2022 45th Jubilee International Convention on Information, Communication and Electronic Technology (MIPRO) (pp. 1301-1306). IEEE.
- Idri, A., Hosni, M. & Abran, A. (2016). Revisión sistemática de la literatura sobre estimación del esfuerzo conjunto. *Revista de Sistemas y Software*, 118, 151-175.
- Jordan, A. E. (2024). An optimized LSTM neural network for accurate estimation of software development effort. *Mathematics*, 12(2), 200
- Iqbal, M., Ijaz, M., Mazhar, T., Shahzad, T., Abbas, Q., Ghadi, Y., Ahmad, W., & Hamam, H. (2024). Exploring issues of story-based effort estimation in agile software development (ASD). *Science of Computer Programming*, 103114.

Jadhav, A., Kaur, M., & Akter, F. (2022). Evolution of software development effort and cost estimation techniques: Five decades study using automated text mining approach. *Mathematical Problems in Engineering*, 2022, 1-17.

Jawa, M., & Meena, S. (2022). Software effort estimation using synthetic minority over-sampling technique for regression (SMOTER). In: 2022 3rd International Conference for Emerging Technology (INCET), (pp. 1-6). IEEE.

Jodpimai, P., Sophatsathit, P., & Lursinsap, C. (2018a). Ensemble effort estimation using selection and genetic algorithms. *International Journal of Computer Applications in Technology*, 58(1), 17-28.

Jodpimai, P., Sophatsathit, P., & Lursinsap, C. (2018b). Re-estimating software effort using prior phase efforts and data mining techniques. *Innovations in Systems and Software Engineering*, 14, 209-228.

Johnson, J. (2020). *CHAOS 2020: Beyond infinity*. The Standish Group.

Jørgensen, M. (2004). Top-down and bottom-up expert estimation of software development effort. *Information and Software Technology*, 46(1), 3-16.

Jørgensen, M. (2005). Practical guidelines for expert-judgment-based software effort estimation. *IEEE software*, 22(3), 57-63.

Karna, H., Vicković, L., & Gotovac, S. (2019). Application of data mining methods for effort estimation of software projects. *Software: Practice and Experience*, 49(2), 171-191.

Karna, H., Gotovac, S., & Vicković, L. (2020). Data mining approach to effort modeling on agile software projects. *Informatica*, 44(2).

Khan, J. A., Khan, S. U. R., Iqbal, J., & Rehman, I. U. (2021). Empirical investigation about the factors affecting the cost estimation in global software development context. *IEEE Access*, 9, 22274-22294.

Kitchenham, B., & Madeyski, L. (2024). Recommendations for analysing and meta-analysing small sample size software engineering experiments. *Empirical Software Engineering*, 29(6), 137.

Kraut, N., & Transchel, F. (2022, March). On the application of SCRUM in data science projects. In: 2022 7th International Conference on Big Data Analytics (ICBDA) (pp. 1-9). IEEE.

Kuan, S. (2017). Factors on software effort estimation. *International Journal of Software Engineering & Applications*, 8(1), 23-32.

Kuhrmann, M., Tell, P., Hebig, R., Klünder, J., Münch, J., Linssen, O., Pfahl, D., Felderer, M., Prause, C. R., MacDonell, S. G., Nakatumba-Nabende, J., Raffo, D., Beecham, S., Tüzün, E., López, G., Paez, N., Fontdevila, D., Licorish, S. A., Küpper, S., Ruhe, G., Knauss, E., Özcan-Top, Ö., Clarke, P., McCaffery, F., Genero, M., Piattini, M., Kalinowski, M., Conte, T., Prikladnicki, R., Krusche, S., Coşkunçay, A., Scott, E., Calefato, F., Pimonova, S., Pfeiffer, R. H., Schultz, U. P., Heldal, R., Fazal-Baqaie, M., Anslow, C., Nayebi, M., Schneider, K., Sauer, S., Winkler, D., Biffl, S., Bastarrica, M. C., & Richardson, I. (2022). What makes agile software development agile? *IEEE Transactions on Software Engineering*, 48(9), 3523–3539.

Kula, E., Greuter, E., Van Deursen, A., & Gousios, G. (2021). Factors affecting on-time delivery in large-scale agile software development. *IEEE Transactions on Software Engineering*, 48(9), 3573-3592.

Kumar, C., Khan, P. S., Srinivas, M., Jha, S. K., Prakash, S., & Rathore, R. S. (2025). Ensemble learning for software requirement-risk assessment: A comparative study of bagging and boosting approaches. *Future Internet*, 17(9), 387.

- Laplane, P. A., & Kassab, M. (2022). What every engineer should know about software engineering. New York, NY: CRC Press.
- Leong, J., May Yee, K., Baitsegi, O., Palanisamy, L., & Ramasamy, R. K. (2023). Hybrid project management between traditional software development lifecycle and agile based product development for future sustainability. *Sustainability*, 15(2), 1121.
- Lock, D. (2020). Project management. 10th Edition. New York, NY: Routledge.
- López-Martínez, J., Juárez-Ramírez, R., Huertas, C., Jiménez, S., & Guerra-García, C. (2016, April). Problems in the adoption of agile-Scrum methodologies: A systematic literature review. In: 2016 4th International Conference in Software Engineering Research and Innovation (CONISOFT) (pp. 141-148). IEEE.
- Ma, X., Zhang, Y., & Wang, Y. (2015, July). Performance evaluation of kernel functions based on Grid Search for support vector regression. In: 2015 IEEE 7th International Conference on Cybernetics and Intelligent Systems (CIS) and IEEE Conference on Robotics, Automation and Mechatronics (RAM) (pp. 283-288). IEEE.
- Mahnič, V., & Hovelja, T. (2012). On using planning poker for estimating user stories. *Journal of Systems and Software*, 85(9), 2086-2095.
- Mahmood, Y., Kama, N., Azmi, A., Khan, A. S., & Ali, M. (2022). Software effort estimation accuracy prediction of machine learning techniques: A systematic performance evaluation. *Software: Practice and experience*, 52(1), 39-65.
- Mallidi, R. K., & Sharma, M. (2021). Study on agile story point estimation techniques and challenges. *International Journal of Computer Applications*, 174(13), 9-14.
- Mantovani Fontana, R., & Marczak, S. (2020). Characteristics and challenges of agile software development adoption in Brazilian government. *Journal of Technology Management & Innovation*, 15(2), 3-10.
- Marnada, P., Raharjo, T., Hardian, B., & Prasetyo, A. (2022). Agile project management challenge in handling scope and change: A systematic literature review. *Procedia Computer Science*, 197, 290-300.
- Martínez, B. B., Cruz-Barbosa, R., & García, I. (2025). Mejora de la estimación del esfuerzo en proyectos de software mediante métodos de sobremuestreo y aprendizaje computacional. *Revista de Investigación en Tecnologías de la Información*, 13(31 Especial), 80-93
- Matsubara, P. G. F., Gadelha, B. F., Steinmacher, I., & Conte, T. U. (2022). SEXTAMT: A systematic map to navigate the wide seas of factors affecting expert judgment software estimates. *Journal of Systems and Software*, 185, 111148.
- McConnell, S. (2006). Software estimation: Demystifying the black art. Microsoft Press.
- Mendes, E. (2013). Improving project management of healthcare projects through knowledge elicitation. In: Lares, M., Lawless, W., Lopes, N., Martin, I., Martinho, R., Putnik, G. D., Rijo, R., Simões, R., Suomi, R., Varajão, J. (Eds.), *Handbook of Research on ICTs and Management Systems for Improving Efficiency in Healthcare and Social Care* (pp. 750-769). Hoboken, NJ: IGI Global.
- Mendoza, J. G. F., Vera, F. G., González, C. C. O., & Contreras, A. S. (2019). Propuesta metodológica para la estimación de proyectos gestionados mediante Scrum, con enfoque a la pequeña industria del software. *Pistas Educativas*, 41(134), 120-128.
- Mishra, A., Abdalhamid, S., Mishra, D., & Ostrovska, S. (2021). Organizational issues in embracing Agile methods: An empirical assessment. *International Journal of System Assurance Engineering and Management*, 12, 1420-1433.

Mishra, D., Tripathi, S. M., Chaurasia, A., & Chaurasia, P. K. (2025). A review on ensemble learning methods: Machine learning approach. *International Journal of Research Publication and Reviews*, 6(2), 3795.

Moradbeiky, A., Khatibi, V., & Jafari Shahbazzadeh, M. (2022). 3LEE: A 3-layer effort estimator for software projects. *International Journal of Industrial Electronics Control and Optimization*, 5(1), 31-42.

Mustapha, H., & Abdelwahed, N. (2019). Investigating the use of random forest in software effort estimation. *Procedia Computer Science*, 148, 343-352.

Nisyak, A. K., Rizkiyah, K., & Raharjo, T. (2020, October). Human related challenges in agile software development of government outsourcing project. In: 2020 7th International Conference on Electrical Engineering, Computer Sciences and Informatics (EECSI) (pp. 222–229). IEEE.

Noteboom, C., Ofori, M., Sutrave, K., & El-Gayar, O. (2021, January). Agile project management: A systematic literature review of adoption drivers and critical success factors. In: *Proceedings of the 54th Hawaii International Conference on System Sciences* (pp. 6776-6783). ScholarSpace.

Obondi, K. (2022). The utilization of project risk monitoring and control practices and their relationship with project success in construction projects. *Journal of Project Management*, 7(1), 35-52.

Ocaña-Fernández, Y., & Fuster-Guillén, D. (2021). La revisión bibliográfica como metodología de investigación. *Revista Tempos y Espacios en Educación*, 14(33), e15614.

Ogunleye, J. O. (2020). Review of current data mining techniques used in the software effort estimation. In: Silhavy, R., Silhavy, P., Prokopova, Z. (Eds.), *Software Engineering Perspectives in Intelligent Systems* (pp. 393-408). Heidelberg, Germany: Springer International Publishing.

Pacheco, C., Garcia, I., Calvo-Manzano, J. A., & Reyes, M. (2023). Measuring and improving software requirements elicitation in a small-sized software organization: a lightweight implementation of ISO/IEC/IEEE 15939: 2017—systems and software engineering—measurement process. *Requirements Engineering*, 28(2), 257-281.

Palomo, S. R. G., & Gil, E. M. (2020). Aproximación a la ingeniería del software. Centro de Estudios Ramón Areces, S. A.

Pandey, M., Litoriya, R., & Pandey, P. (2020). Validation of existing software effort estimation techniques in context with mobile software applications. *Wireless Personal Communications*, 110(4), 1659-1677.

Pastrana-Pardo, M. A., Ordóñez-Erazo, H. A., & Cobos-Lozada, C. A. (2022). Approach to the best practices in software development based on DevOps and Scrum used in very small entities. *Revista Facultad de Ingeniería*, 31(61), 134-141.

Pasuksmit, J., Thongtanunam, P., & Karunasekera, S. (2024). A Systematic Literature Review on Reasons and Approaches for Accurate Effort Estimations in Agile. *ACM Computing Surveys*.

Patt, D. (2024). Too critical to fail: Getting software right in an age of rapid innovation. Pennsylvania, NW: The Hudson Institute.

Paulson, N. H., Kubal, J., Ward, L., Saxena, S., Lu, W., & Babinec, S. J. (2022). Feature engineering for machine learning enabled early prediction of battery lifetime. *Journal of Power Sources*, 527, 231127.

- Pérez Pupo, I., Piñero Pérez, P. Y., García Vacacela, R., Bello, R., & Acuña, L. A. (2020, June). Discovering fails in software projects planning based on linguistic summaries. In: International Joint Conference on Rough Sets (pp. 365-375). Cham: Springer International Publishing.
- Pinto, J. K. (2020). Project management: Achieving competitive advantage. 5th Edition. Pittsburgh, PA: Pearson.
- Piñeros, C. A., Sierra, L. M., Peluffo, D. H., & Timana, J. A. (2023). Effort estimation in agile software development: A systematic map study. INGE CUC, 19(1), 22-36.
- Project Management Institute. (2021). Guía de los Fundamentos para la Dirección de Proyectos (Guía del PMBOK) y El Estándar para la Dirección de Proyectos. 7th Edition. Boston, MA: Project Management Institute Inc.
- Project Management Institute (2022). Process groups: A practical guide. Boston, MA: Project Management Institute.
- Pospieszny, P., Czarnacka-Chrobot, B., & Kobylinski, A. (2018). An effective approach for software project effort and duration estimation with machine learning algorithms. Journal of Systems and Software, 137, 184-196
- Pressman, R. S., & Maxim, B. R. (2019). Software engineering: A practitioner's approach (9th Edition). McGraw Hill.
- Prieto-Bustamante, F. (2020). Estimación de esfuerzo en desarrollo de software ágil: Estudio del estado actual en Bogotá. ITECKNE, 17(2), 110–131.
- Putrianasari, R., Budiardjo, E. K., Mahatma, K., & Raharjo, T. (2024). Problems in the adoption of agile-Scrum software development process in small organization: A systematic literature review. Sinkron: Jurnal dan Penelitian Teknik Informatika, 9(1), 495-504.
- Rad, M., Rafiei, A., Grunwell, J., & Kamaleswaran, R. (2025). Tackling the small imbalanced horizontal dataset regressions by Stability Selection and SMOGN: A case study of ventilation-free days prediction in the pediatric intensive care unit and the importance of PRISM. International Journal of Medical Informatics, 196, 105809.
- Ramessur, M. A., & Nagowah, S. D. (2020). Factors affecting sprint effort estimation. In: Advanced Computing and Intelligent Engineering: Proceedings of ICACIE 2018, Volume 2 (pp. 507-518). Springer Singapore.
- Ramessur, M. A., & Nagowah, S. D. (2021). A predictive model to estimate effort in a sprint using machine learning techniques. International Journal of Information Technology, 13(3), 1101-1110.
- Rao, K. E., Terlapu, P. R. V., Naidu, P. A., Kumar, T. R., & Pydi, B. M. (2024). Feature importance for software development effort estimation using multi level ensemble approaches. Bulletin of Electrical Engineering and Informatics, 13(2), 1090-1102.
- Rashid, C. H., Shafi, I., Ahmad, J., Thompson, E. B., Vergara, M. M., Diez, I. D. L. T., & Ashraf, I. (2023). Software cost and effort estimation: Current approaches and future trends. IEEE Access, 11, 99268-99288.
- Rastogi, H., Dhankhar, S., & Kakkar, M. (2014, September). A survey on software effort estimation techniques. In: 2014 5th International Conference-Confluence The Next Generation Information Technology Summit (Confluence) (pp. 826-830). IEEE.
- Resmi, V., & Anitha, K. L. (2023). Software effort estimation using machine learning techniques. SAMRIDDHI: A Journal of Physical Sciences, Engineering and Technology, 15(01), 86-90.

Rodríguez, C. A. P., Martínez, L. M. S., Ordoñez, D. H. P., & Peña, J. A. T. (2023). Effort estimation in agile software development: A systematic map study. *INGE CUC*, 19(1).

Rodríguez Sánchez, E., Vázquez Santacruz, E. F., & Cervantes Maceda, H. (2023). Effort and cost estimation using decision tree techniques and story points in agile software development. *Mathematics*, 11(6), 1477.

Rosa, W., Clark, B. K., Madachy, R., & Boehm, B. W. (2021). Empirical effort and schedule estimation models for agile processes in the US DoD. *IEEE Transactions on Software Engineering*, 48(8), 3117-3130.

Rubin, K. S. (2012). *Essential Scrum: A practical guide to the most popular agile process*. Boston, MA: Addison-Wesley.

Saeed, A., Butt, W. H., Kazmi, F., & Arif, M. (2018, February). Survey of software development effort estimation techniques. In: *Proceedings of the 2018 7th International Conference on software and computer applications* (pp. 82-86). ACM.

Saif, S. M. (2020). Software effort estimation for successful software application development. In: *Tools and Techniques for Software Development in Large Organizations: Emerging Research and Opportunities* (pp. 45-97). IGI Global.

Sakhrawi, Z., Sellami, A., & Bouassida, N. (2022). Software enhancement effort estimation using correlation-based feature selection and stacking ensemble method. *Cluster Computing*, 25(4), 2779-2792.

Salameh, H. (2014). What, when, why, and how? A comparison between agile project management and traditional project management methods. *International Journal of Business and Management Review*, 2(5), 52-74.

Saltz, J. S. (2021, December). CRISP-DM for data science: strengths, weaknesses and potential next steps. In: *2021 IEEE International Conference on Big Data (Big Data)* (pp. 2337-2344). IEEE.

Sandeep, R. C., Sánchez-Gordón, M., Colomo-Palacios, R., & Kristiansen, M. (2022). Effort estimation in agile software development: An exploratory study of practitioners' perspective. In: A. Przybyłek, A. Jarzębowicz, I. Luković, & Y. Y. Ng (Eds.), *International Conference on Lean and Agile Software Development* (pp. 136–149). Springer.

Schoch, K. (2020). Case study research. *Research design and methods: An applied guide for the scholar-practitioner*, 31(1), 245-258.

Schwaber, K., & Sutherland, J. (2020). *La guía definitiva de Scrum: Las reglas del juego*.

Schwaber, K., & Sutherland, J. (2021). *The Scrum Guide 2020*. San Francisco, CA: Scrum Alliance.

Schwalbe, K. (2021). *An introduction to project management. Predictive, agile, and hybrid approaches*. 7th Edition. Minneapolis, MN: Schwalbe Publishing.

Sehra, S. K., Kaur, J., Brar, Y. S., & Kaur, N. (2014, April). Analysis of data mining techniques for software effort estimation. In: *2014 11th International Conference on Information Technology: New Generations* (pp. 633-638). IEEE.

Sehra, S. K., Brar, Y. S., & Kaur, N. (2016). Predominant factors influencing software effort estimation. *International Journal of Computer Science and Information Security*, 14(7), 107-110.

Sembhoo, A., & Gobin-Rahimbux, B. (2023). *A SLR on Deep Learning Models Based on Textual Information For Effort Estimation in Scrum*. University of Mauritius.

- Septian, W., & Gata, W. (2017, October). Software development framework on small team using agile framework for small projects (AFSP) with neural network estimation. In: 2017 11th International Conference on Information & Communication Technology and System (ICTS) (pp. 259-264). IEEE.
- Sharma, S., & Vijayvargiya, S. (2021). Applying soft computing techniques for software project effort estimation modelling. In: Nanoelectronics, Circuits and Communication Systems: Proceeding of NCCS 2019 (pp. 211-227). Springer Singapore.
- Sommerville, I. (2019). Engineering software products: An introduction to modern software engineering. Boston, MA: Pearson.
- Srivastava, A., Mehrotra, D., Kapur, P. K., & Aggarwal, A. G. (2020). Analytical evaluation of agile success factors influencing quality in software industry. *International Journal of System Assurance Engineering and Management*, 11(Suppl. 2), 247–257.
- Srivastava, P., Srivastava, N., Agarwal, R., & Singh, P. (2022). Software development estimation using soft computing techniques. In: *Emerging Trends in IoT and Computing Technologies* (pp. 321-325). Routledge.
- Strode, D., Dingsøyr, T., & Lindsjorn, Y. (2022). A teamwork effectiveness model for agile software development. *Empirical Software Engineering*, 27, Article 56.
- Sudarmaningtyas, P., & Mohamed, R. B. (2020, September). Extended planning poker: A proposed model. In: 2020 7th International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE) (pp. 179-184). IEEE.
- Sudarmaningtyas, P., & Mohamed, R. B. (2021). A review article on software effort estimation in agile methodology. *Pertanika Journal of Science & Technology*, 29(2), 837–861.
- Sudarmaningtyas, P., & Mohamed, R. B. (2022). Significant factors in agile software development of effort estimation. *Pertanika Journal of Science & Technology*, 30(4), 2481-2878.
- Suhendar, M. T. H., & Widayani, Y. (2023, September). Machine learning application development guidelines using CRISP-DM and Scrum concept. In: 2023 IEEE International Conference on Data and Software Engineering (ICoDSE) (pp. 168-173). IEEE.
- Swandari, Y., Ferdiana, R., & Permanasari, A. E. (2023, September). Research trends in software development effort estimation. In: 2023 10th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI) (pp. 625-630). IEEE.
- Tanveer, B., Guzmán, L., & Engel, U. M. (2016, May). Understanding and improving effort estimation in agile software development: An industrial case study. In: *Proceedings of the International Conference on Software and Systems Process* (pp. 41-50). ACM.
- Tanveer, B., Guzmán, L., & Engel, U. M. (2017). Effort estimation in agile software development: Case study and improvement framework. *Journal of Software: Evolution and Process*, 29(11), e1862.
- Tam, C., Moura, E. J. D. C., Oliveira, T., & Varajão, J. (2020). The factors influencing the success of on-going agile software development projects. *International Journal of Project Management*, 38(3), 165–176.
- Traini, L. (2022). Exploring performance assurance practices and challenges in agile software development: An ethnographic study. *Empirical Software Engineering*, 27, Article 74.
- Trendowicz, A., & Jeffery, R. (2014). Constructive Cost Model–COCOMO. *Software project effort estimation. Foundations and Best Practice Guidelines for Success* (pp. 277-293). Cham: Springer International Publishing.

Tsui, F., Karam, O., & Bernal, B. (2022). *Essentials of software engineering*. Jones & Bartlett Learning.

United States Government Accountability Office. (2023). *DOD needs to improve performance reporting and development planning*. IT Systems Annual Assessment. Washington, DC: GAO.

Usman, M., Mendes, E., & Börstler, J. (2015, April). Effort estimation in agile software development: A survey on the state of the practice. In: *Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering* (pp. 1-10). ACM.

Varshini, P., Kumari, A., & Varadarajan, V. (2021a). Estimating software development efforts using a random forest-based stacked ensemble approach. *Electronics*, 10(10), 1195.

Varshini, A. P., Kumari, K. A., Janani, D., & Soundariya, S. (2021b, February). Comparative analysis of machine learning and deep learning algorithms for software effort estimation. *Journal of Physics: Conference Series*, 1767(1), 012019. IOP Publishing.

Vegas, S., & Juristo, N. (2024). A Course on Experimentation in Software Engineering: Focusing on Doing. In *Handbook on Teaching Empirical Software Engineering* (pp. 301-324). Cham: Springer Nature Switzerland.

Vickler, A. (2021). *Agile project management: Beginner's step by step guide to learn Scrum*. New York, NY: Ladoo Publishing LLC.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., & Wesslén, A. (2024). *Experimentation in Software Engineering* (2024 edition).

Wohlin, C., Höst, M., & Henningsson, K. (2003). Empirical research methods in Software Engineering. In: Conradi, R., Wang, A. I. (Eds.), *Empirical Methods and Studies in Software Engineering* (pp 7-23). Heidelberg, Germany: Springer, Berlin.

Wu, J., & Gao, S. (2016, August). Software productivity estimation by regression and Nave-Bayes classifier - An empirical research. In: *International Conference on Promotion of Information Technology (ICPIT 2016)* (pp. 20-24). Atlantis Press.

Wu, D., Li, J., & Bao, C. (2018). Case-based reasoning with optimized weight derived by particle swarm optimization for software effort estimation. *Soft Computing*, 22, 5299-5310.

Zasa, F. P., Patrucco, A., & Pellizzoni, E. (2020). Managing the hybrid organization: How can agile and traditional project management coexist?. *Research-Technology Management*, 64(1), 54-63.

Zhang, F., & O'Donnell, L. J. (2020). Support vector regression. In: Mechelli, A., Vieira, S. (Eds.), *Machine learning* (pp. 123-140). Boston, MA: Academic Press.

Ziauddin, S. K. T., & Zia, S. (2012). An effort estimation model for agile software development. *Advances in computer science and its applications (ACSA)*, 2(1), 314-324.

Zúñiga, P. I. V., Cedeño, R. J. C., & Palacios, I. A. M. (2023). Metodología de la investigación científica: Guía práctica. *Ciencia Latina Revista Científica Multidisciplinar*, 7(4), 9723-9762.

7. Anexo A.- Acrónimos

AFSP	Agile Framework for Small Projects
ALO	Ant Lion Optimizer
ANFIS	Adaptive Neuro-Fuzzy Inference System
ANN	Artificial Neural Network
APM	Agile Project Management
Bagging	Bootstrap Aggregating
CART	Classification and Regression Trees
CBR	Case-Based Reasoning / Retrieval
CRISP-DM	Cross Industry Standard Process for Data Mining
DoD	Department of Defense
DOP	Duration of Project
DS	Daily Scrum
DT	Decision Tree
EE	Early Estimation
EEBAT	Energy Efficient BAT
EEE	Effort Estimation by Ensemble
GAO	Government Accountability Office
GBT	Gradient Boosted Tree
GEM	Generalized Estimation Method
IE	Iterative Estimation
ISBSG	International Software Benchmarking Standards Group
LR	Linear Regression
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
MdAE	Median Absolute Error
MdMAE	Median of the Magnitude of Absolute Error
MdMRE	Median of the Magnitude of Relative Error
MLP	Multilayer Perceptron
MMRE	Mean Magnitude of Relative Error

MRE	Magnitude of Relative Error
MSE	Mean Squared Error
NB	Naive Bayes
NSP	New Story Point
PB	Product Backlog
PD	Project Domain
PF	Priority Factor
PMBOK	Project Management Body of Knowledge
PMI	Project Management Institute
PO	Product Owner
PSO	Particle Swarm Optimization
RF	Random Forest
RMSE	Root Mean Squared Error
RSS	Residual Sum of Squares
SB	Sprint Backlog
SDECE	Software Development Effort and Cost Estimation
SDLC	Software Development Life Cycle
SM	Scrum Master
SOP	Size of Project
SP	Story Point
SPA	Single Page Application
SR	Sprint Review
SRP	Sprint Retrospective
ST	Scrum Team
SVM	Support Vector Machine
SVR	Support Vector Regression
TSS	Total Sum of Squares
UF	Uncertainty Factor
UV	Unadjusted Value
WBS	Work Breakdown Structure
XP	eXtreme Programming

mauge ror