ISSN 1870-4069

Estimación de esfuerzo en desarrollo de software ágil utilizando redes neuronales artificiales

Eduardo Rodríguez Sánchez, Eduardo Vázquez Santacruz,
Humberto Cervantes Maceda

Universidad Autónoma Metropolitana,
Posgrado en Ciencias y Tecnologías de la Información,
México

erodsmx@gmail.com, evazquez.santacruz@izt.uam.mx,
hcm@xanum.uam.mx

Resumen. La estimación de esfuerzo es importante para planificar correctamente el uso de recursos en un proyecto de TI. En la fase de planeación de un proyecto, durante la elaboración de los artefactos de visión y alcance, el equipo involucrado realiza una estimación inicial aproximada de tiempo y costo. Para mejorar la precisión de la estimación de esfuerzo en desarrollo de software existen varias técnicas de estimación: function points, object points, use case points, story points, etc. Los puntos de historia de usuario o story points, son la base de marcos de trabajo ágil que actualmente están tomando más fuerza en el desarrollo de software. Una de las principales lagunas de conocimiento se encuentra en la aplicación de redes neuronales y aprendizaje profundo en estimación de esfuerzo en desarrollo ágil. Este artículo contribuye a fortalecer el uso de redes neuronales como método de estimación de esfuerzo a nivel de proyecto en marcos de trabajo que usan un enfoque de puntos de historia de usuario como Scrum. Este trabajo de investigación presenta un estudio sobre el análisis de la precisión de la predicción del proceso de estimación ejecutado con técnicas de aprendizaje automático, tomando como referencia el modelo de estimación de esfuerzo para el desarrollo de software ágil propuesto en [8]. Los algoritmos de aprendizaje supervisado propuestos para realizar las estimaciones son redes neuronales de tipo perceptrón multicapa y redes neuronales recurrentes. El desempeño de los modelos se compara a través del error cuadrático medio, error relativo medio y el coeficiente de determinación R^2. Los algoritmos se ejecutan aplicando validación cruzada 10-Fold. Los resultados comparan la ejecución de los algoritmos con y sin la propuesta de utilizar categorías de clasificación por tamaño probando que el uso de categorías mejora la precisión de la predicción.

Palabras clave: Estimación, tiempo, costo, desarrollo ágil, aprendizaje automático, redes neuronales.

Effort Estimation in Agile Software Development Using Artificial Neural Networks

Abstract. Effort estimation is important to correctly plan the use of resources in an IT project. In the planning phase of a project, during the elaboration of the

vision and scope artifacts, the team involved makes an initial rough estimate of time and cost. To improve the accuracy of effort estimation in software development, there are several estimation techniques: function points, object points, use case points, story points, etc. User story points or story points are the basis of agile frameworks that are currently gaining strength in software development. One of the main knowledge gaps is in the application of neural networks and deep learning in effort estimation in agile development. This article contributes to strengthening the use of neural networks as a project-level effort estimation method in frameworks that use a user story point approach such as Scrum. This research paper presents a study on the analysis of the prediction accuracy of the estimation process executed with machine learning techniques, taking as reference the effort estimation model for agile software development proposed in [8]. The supervised learning algorithms proposed to perform the estimations are multilayer perceptron-type neural networks and recurrent neural networks. The performance of the models is compared through the mean square error, mean relative error and the determination coefficient R^2. The algorithms are executed applying 10-Fold cross-validation. The results compare the execution of the algorithms with and without the proposal to use size classification categories, proving that the use of categories improves the prediction accuracy.

Keywords: Estimation, time, cost, agile development, machine learning, neural networks.

1. Introducción

El proceso de estimación en el desarrollo de software es una actividad complicada para el equipo involucrado [3]. La planeación de un proyecto contempla tres elementos: recursos, alcance y fecha de entrega. La estimación requiere de una aproximación a los costos, tiempos, y factores que pueden afectar el desarrollo del proyecto [8], también conocidos como riesgos. Un modelo de estimación ayuda a tomar acciones de control y mejores decisiones que impactan directamente en la organización.

El principal problema es que cuando no se aplica un modelo de estimación, ni se realizan proyecciones adecuadas, las fechas estimadas no consideran el trabajo relacionado y las predicciones son aleatorias poniendo en riesgo al proyecto al no tener actividades y fechas claras.

El impacto de una mala planeación se refleja en entregas tardías, aumento de costos, deuda técnica, y a largo plazo posible pérdida de clientes. En el 14vo. informe anual del estado ágil [12] se reporta que una de las principales razones para adoptar un enfoque ágil es la aceleración en la entrega de software y la mejora en la habilidad de adaptarse al cambio de prioridades.

Como lo menciona el informe, el éxito de un proyecto se mide a través del valor de negocio entregado, la satisfacción del cliente, la velocidad del desarrollo, el presupuesto vs el costo actual, la planeación vs la ejecución real de las historias, entre otras características. En metodologías ágiles como Scrum, una historia de usuario es un enunciado corto que refleja los requerimientos funcionales del sistema, similar a un caso de uso.

La estructura de una historia de usuario permite identificar al actor, la acción que quiere realizar y la justificación de por qué el sistema debe realizar dicha acción. El conjunto de historias que componen el sistema se implementan en ciclos de trabajo conocidos como *Sprints*.

El presente proyecto de investigación está orientado a proponer una solución que apoye el proceso de estimación de proyectos desarrollados con un enfoque ágil. Aplicando técnicas de aprendizaje supervisado, este trabajo proporciona un modelo de estimación de esfuerzo híbrido basado en redes neuronales artificiales e historias de usuario que en conjunto con categorías de clasificación se mejora la precisión de la predicción de las estimaciones de tiempo de finalización y costo total de un proyecto.

En la sección 2 se presenta el trabajo relacionado, resaltando aquellas investigaciones que se encuentran dentro del mismo campo de estudio: desarrollo de software ágil utilizando puntos de historia. Posteriormente en la sección 3 se desglosa la propuesta del trabajo de investigación, seguido del marco teórico.

En la sección 5 se desglosa la evaluación empírica reportando la estructura de los experimentos, el diseño de los modelos de aprendizaje automático utilizados y los criterios de evaluación aplicados. Posteriormente se describen los resultados agrupando las gráficas y tablas correspondientes a la evaluación de los modelos y la comparativa con el estado del arte. Por último se presentan las conclusiones y el trabajo a futuro.

2. Trabajo relacionado

El desarrollo de software ágil surgió en la primera década del siglo XXI, en 2002 Grenning propuso el uso de Planning Poker como técnica de estimación y más tarde en 2005 M. Cohn sugirió que este método era útil en estimación de proyectos ágiles [10]. Coelho [9] destaca la importancia de los puntos de historia de usuario y la velocidad como medición del progreso por iteración del proyecto.

Fue hasta el año 2012, que Wen J. y otros investigadores sugirieron el aprendizaje automático como otra categoría de las técnicas de estimación [10]. M. Gultekin & O. Kalipsiz [19] prueban técnicas de aprendizaje automático y dos notaciones de puntos de historia, aritmética y fibonacci, probando que la segunda da mejor precisión al realizar estimaciones.

Marta Fernández et al. [15] realiza una revisión sistemática de la literatura destacando investigaciones sobre métodos ágiles y clasificando los trabajos dentro de las categorías de juicio de experto (Planning Poker, Wideband Delphi) y aquellas basadas en datos (técnicas de aprendizaje automático).

Dentro de los modelos de estimación de esfuerzo se encuentran el trabajo de Zia [8] orientado a producir estimaciones de tiempo y costo de un proyecto utilizando un modelo de regresión lineal. Rashmi Popli & Naresh Chauhan [13] calculan el costo, esfuerzo y duración de un proyecto pequeño y mediano usando un enfoque de historias de usuario.

Atef Tayh Raslan et al. [4] propone mejorar la precisión de la estimación aplicando un modelo de estimación basado en puntos de historia y lógica difusa. Sakshi Garg et al. [16] plantea un modelo basado en Principal Component Analysis y programación con restricciones asociado a los costos del desarrollo de software.

Tabla 1. Micro estado del arte, selección de tres artículos que comparte el mismo modelo de estimación y conjunto de datos.

No	Autor	Título	Técnica de Machine Learning reportadas	Precisión(%) Tiempo	Precisión(%) Costo	Entradas	Salidas	Criterios de evaluación	Dataset
1	Aditi Panda et al.	Empirical Validation of Neural Network Models for Agile Software Effort Estimation based on Story Points	General Regression Neural Network	85.92	No	Total de puntos de historia y velocidad inicial del proyecto	Predicción de tiempo	Error cuadrático medio	21 proyectos desarrollados por seis compañías
			Probabilistic Neural Network	87.66				Coef ciente de determinación (R^2)	
			GMDH Polynomial Neural Network	89.67				Magnitud media de error relativo	
			Cascade-Correlation Neural Network	94.76				Porcentaje de precisión	
2	Ch. Prasada Rao et al.	An Agile Effort Estimation Based on Story Points Using Machine Learning Techniques	Adaptive Neuro-Fuzzy Interface System	76.19	57.14		Predicción de tiempo y costo	Magnitud media de error relativo	
			Generalized Regression Neural Networks	76.19	76.19			Porcentaje de precisión	
			Radial Basis Function Networks	76.19	76.19				
3	S. M. Satapathy, S. K. Rath	Empirical assessment of machine learning models for agile software development effort estimation using story points	Decision Tree	90.48	No		Predicción de tiempo	Error absoluto medio	
			Random Forest	95.24				Magnitud media de error relativo	
			Stochastic Gradient Boosting (SGB)	95.24				Porcentaje de precisión	

E. Scott, & D. Pfahl [5] proponen la estimación de puntos de historia a través de un modelo que asigna puntos de historia a los reportes de incidencia tomando en cuenta el perfil del desarrollador. Jitender Choudhari y Ugrasen Suman [11] plantean un modelo basado en Extreme Programming (XP) que usa puntos de historia para calcular el volumen de esfuerzo en fase de mantenimiento.

O. Malgonde & K. Chari [17] plantea un modelo para la predicción de esfuerzo de la historia de usuario mediante un modelo predictivo, un modelo basado en ensamble y un modelo de optimización para el esfuerzo. Otras propuestas con un enfoque a nivel de historia de usuario son los trabajos de Morakot Choetkiertikul et al.[7] y M. Durán et al. [18] en donde se propone un modelo de predicción basado en redes neuronales de tipo LSTM y Recurrent Highway, y un método para estimar la complejidad de la historia a través de su descomposición utilizando redes bayesianas respectivamente.

El presente trabajo de investigación tiene particular interés en el modelo propuesto por Zia et al. [8], con enfoque a nivel de proyecto. El modelo de Zia forma parte del conjunto de modelos de estimación de esfuerzo en desarrollo ágil y genera un porcentaje de precisión del 57.14 % para tiempo de finalización y 61.40 % para costo total.

Tres investigaciones retomaron este modelo para realizar estimaciones utilizando aprendizaje automático, Aditi Panda et al. [1] aplicando cuatro tipo de redes neuronales, Ch. Prasada Rao et al. [3] con tres tipos de redes neuronales, y S. M. Satapathy, S. K. Rath [6] con árboles de decisión, bosques aleatorios y Stochastic Gradient Boosting. En la tabla del Cuadro 1 se agrupan estas tres investigaciones que toman como datos de entrada los 21 proyectos de [8].

En los tres casos, los autores evalúan los modelos de aprendizaje automático con el error relativo y generan un porcentaje de precisión de la predicción. La comparativa de la tabla 1 muestra que los árboles de decisión ofrecen los mejores resultados. El trabajo de Satapathy [6] reconoce que hay poca disponibilidad de investigación para proporcionar un procedimiento sistemático con el fin de estimar el esfuerzo de los proyectos desarrollados con metodología ágil, por lo que el presente proyecto pretende

contribuir a esta línea de conocimiento.

3. Propuesta

El presente trabajo de investigación tiene la finalidad de estudiar algoritmos de aprendizaje automático, en particular redes neuronales artificiales aplicadas a la estimación de tiempo de finalización y costo total de un proyecto de software.

El estudio desglosa un análisis de la precisión de la predicción del proceso de estimación ejecutado con redes neuronales y aplicando un enfoque de puntos de historia de usuario. La hipótesis del proyecto establece que el uso de categorías de tamaño ofrece estabilidad en la precisión de la predicción de tiempo y costo de los proyectos, lo que reduce la desviación estándar de las estimaciones.

La propuesta contiene un modelo de estimación de esfuerzo basado en regresión lineal que considera el esfuerzo medido en puntos de historia de usuario, la velocidad del equipo,las categorías de clasificación de tamaño esfuerzo, tiempo y costo de un proyecto, así como un modelo de aprendizaje automático compuesto por una red neuronal de tipo perceptrón multicapa y una red neuronal recurrente encargadas de realizar las estimaciones.

A diferencia de los trabajos del estado del arte, ambas técnicas aplicadas componen un ensamble el cual mejora las estimaciones emitidas al promediar los resultados de las técnicas individuales.

Para probar la hipótesis establecida el proyecto genera dos escenarios, el primero recibe como entrada el esfuerzo medido como el total de puntos de historia de usuario requeridos (*Effort*), y la velocidad inicial del equipo (V_i).

Un segundo escenario ejecuta el modelo teniendo como entrada el esfuerzo, la velocidad y las categorías de clasificación de tamaño. Los resultados se comparan para mostrar que las categorías generan mejores predicciones. Los tamaños para las categorías de clasificación son:

$$0 = \text{Grande} = L, \quad 1 = \text{Mediano} = M, \quad 2 = \text{Pequeño} = S.$$

El conjunto de datos consiste en un total de 21 proyectos desarrollados por seis compañías de software en Pakistán, por lo que el costo de los mismos está dado en rupias pakistaníes. El método más simple para clasificar los datos consiste en dividirlos en tres grupos de siete, ordenados por esfuerzo, tiempo y costo.

Cada proyecto tiene un conjunto de características que lo definen como el total de esfuerzo para completar el proyecto, la velocidad del equipo, la desaceleración debido a factores de fricción y fuerzas dinámicas (riesgos), el tamaño del Sprint, los días laborales en el mes y el salario del equipo. Realizando un análisis de componentes principales, las características más importantes son el esfuerzo y la velocidad.

Los modelos de aprendizaje automático se entrenan utilizando 10-Fold cross-validation para evaluar la capacidad de generalización de los algoritmos. Debido a que el conjunto de datos es pequeño se usa una técnica llamada aumento de datos, la cual consiste en duplicar muestras agregando pequeñas cantidades de ruido a las características principales (esfuerzo y velocidad).

Esto da como resultado dos valores con ruido por cada proyecto, teniendo un total de 42 valores con ruido utilizados en entrenamiento y 21 valores sin ruido para prueba (datos reales). Con esta técnica se logra reducir el sobreentrenamiento u *overfitting*.

Adicionalmente los modelos de redes neuronales cuentan con dos capas de ruido gaussiano las cuales toman como datos de entrada la salida de las neuronas de la capa anterior y agregan ruido gaussiano, esto permite modificar los valores en cada época y permite un mejor ajuste del algoritmo de retropropagación al calcular los pesos de la red en cada ciclo.

Aplicando k-Fold los datos se dividen en k subconjuntos de aproximadamente el mismo tamaño y cada partición se usa para entrenar el modelo. De este subconjunto el 30 % se reserva para validación, la cual ocurre después de cada época de entrenamiento para ayudar a determinar si ocurre *underfitting* u *overfitting*. Una vez que finaliza el entrenamiento de los modelos, se emiten las predicciones tomando como valores de entrada los datos reales de los 21 proyectos.

4. Marco teórico

Los elementos clave del proyecto son el modelo de estimación algorítmico y las técnicas de aprendizaje automático descritas a continuación.

4.1. Modelo de estimación de esfuerzo en desarrollo ágil

El modelo de estimación basado en regresión lineal [8] emite estimaciones de tiempo y costo para 21 proyectos desarrollados por 6 organizaciones, los conceptos resumidos más relevantes del modelo son los siguientes.

Tamaño de la historia de usuario: Con una escala propuesta de uno a cinco, siendo uno el tamaño más pequeño y cinco una historia épica:

$$[1, 5] = \{S \in \mathbb{N} : 1 \leq S \leq 5\}.$$

Complejidad: Asocia la dificultad técnica:

$$[1, 5] = \{C \in \mathbb{N} : 1 \leq C \leq 5\}.$$

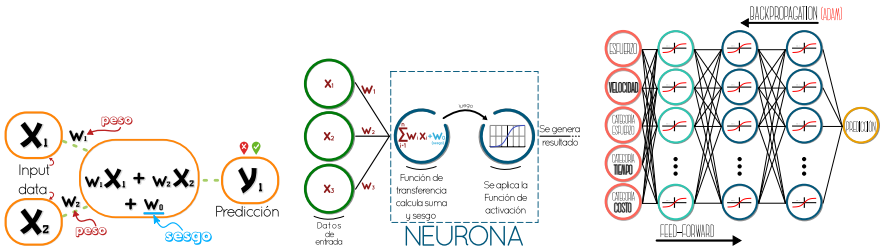
Esfuerzo: Combina el tamaño de la historia (S) y su complejidad (C):

$$E_S = C \times S.$$

El esfuerzo de un proyecto es la suma de los esfuerzos de las historias:

$$E = \sum_{i=1}^{n} (E_S)_i \, .$$

Velocidad: Considera el esfuerzo total (puntos de historia) y el tiempo (duración del sprint):



(a) Composición de un perceptrón simple.

(b) Proceso ejecutado en una neurona

(c) Red neuronal de tipo feedforward

Fig. 1. Una red neuronal de tipo feed forward (c), se compone de perceptrones interconectados entre sí donde los elementos de una capa conectan con las neuronas de la capa siguiente.

$$V_i = \frac{\text{Unidades de esfuerzo completado}}{\text{Sprint}}.$$

Tiempo de finalización: Duración necesaria para completar el proyecto:

$$T = \frac{\text{Esfuerzo}}{\text{Velocidad}}.$$

T se mide en días. Dividiendo T entre el número de días laborales en el mes se obtiene el número de meses para completar el proyecto. Costo total: Contempla una relación de los recursos económicos invertidos como salarios, equipo tecnológico, licencias,marketing, rentas, mobiliario, etc:

$$\text{Costo} = (\alpha T_S)\, T.$$

El valor de α corresponde al valor neto de la relación de los costos asociados al proyecto. T_S es el salario mensual del equipo y T es el tiempo calculado en meses. En la siguiente sección se presentan los modelos que construyen un mecanismo capaz de predecir tiempo de finalización y costo total a partir de los datos de entrada.

4.2. Redes neuronales artificiales

Las redes neuronales artificiales (ANN) son un intento de modelar la capacidad del sistema nervioso para procesar información [2]. Una red neuronal se compone de perceptrones (Figura 1c) y un perceptrón es un modelo simplificado de una neurona capaz de realizar clasificación binaria.

La composición de un perceptrón se muestra en la Figura 1a y se puede pensar como una combinación lineal, el proceso de activación de una neurona se muestra en 1b.

Retomando la definición matemática de una red neuronal de Hornik et al. [14] se tiene que: Para todo $r \in \mathbb{N} \equiv \{1, 2, ...\}$, A^r es el conjunto de todas las funciones afines de $\mathbb{R}^r a\, \mathbb{R}$, esto es, el conjunto de todas las funciones de la forma $A(x) = w \cdot x + b$ donde *w* y *x* son vectores en R^r y *b* es un escalar.

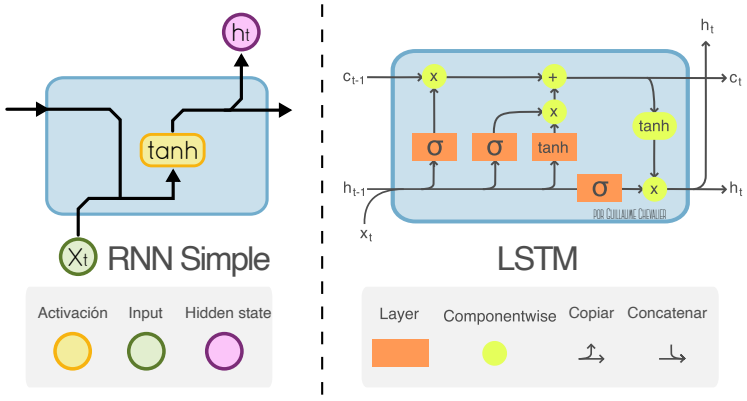


Fig. 2. Una red neuronal recurrente incorpora el concepto de memoria implementando el estado oculto y una capa de activación a la entrada.

$A(x)$ es entonces el producto punto de dos vectores y la adición de un escalar, w corresponde a los pesos de la red y x corresponde a las entradas de la red. El sesgo de una red corresponde al escalar b. Partiendo de esta definición varios autores probaron que una red neuronal de tipo *feedforward* con una capa oculta puede aproximar una función continua multivariante [14].

Red neuronal recurrente El siguiente tipo de red neuronal que mejora la arquitectura del perceptrón multicapa de la Figura 1c son las redes recurrentes o RNN *(Recurrent Neural Network)* que incorporan el concepto de memoria. Matemáticamente una red recurrente simple se formula como:

$$\begin{aligned} h(t) &= f_H\left(W_{IH}\, x(t) + W_{HH}\, h\left(t-1\right)\right) \\ y(t) &= f_O\left(W_{HO}\, h\left(t\right)\right), \end{aligned} \tag{1}$$

donde $x(t)$ y $y(t)$ son los vectores de entrada y salida. W_{IH}, W_{HH}, W_{HO}, son las matrices de los pesos y f_h, f_O son las funciones de activación ocultas y la de salida. En la Figura 2 se muestra una celda recurrente simple y una celda LSTM (*Long Short Term Memory*).

4.3. Métricas de evaluación

El entrenamiento de los modelos diseñados se someten a distintas funciones de error para evaluar su desempeño. Se busca aquella configuración de modelo que disminuya más el error de las predicciones.

Error Cuadrático Medio: La función error cuadrático medio se define como:

$$MSE = (y, \hat{y}) = \frac{1}{n}\sum_{i=0}^{n-1}\left(y_i - \hat{y}_i\right)^2, \tag{2}$$

donde n es el número de muestras, y son los datos reales, $\hat{y}$ las predicciones.

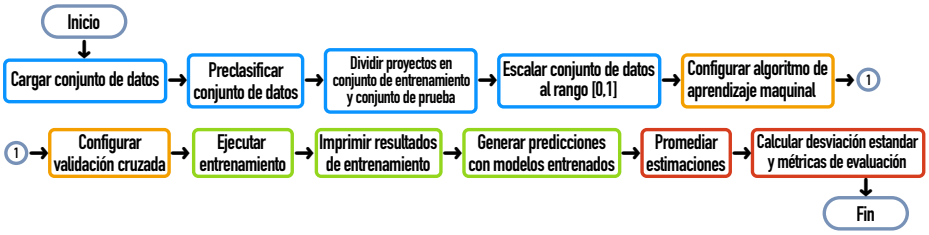


Fig. 3. Diagrama de flujo de los experimentos.

Precisión: La precisión del modelo se calcula utilizando el valor del error relativo *MRE*. El porcentaje de precisión es una primera aproximación de fácil interpretación para conocer qué tan buena es la técnica evaluada:

$$MRE = \frac{|AE_i - PE_i|}{AE_i}, \quad \text{AE = Actual, PE = Predicción ,}$$
$$\text{Precisión } (\%) = (1 - MRE) \times 100. \tag{3}$$

Coeficiente de determinación (R^2): Proporciona una indicación de que tan bueno es el entrenamiento, y por lo tanto, una medida de qué tan probable es que el modelo prediga las muestras no conocidas a través de la proporción de varianza. Si el valor de R^2 es cercano a 1, consideraremos que el algoritmo de aprendizaje automático es bueno o que ofrece resultados confiables:

$$R^2\left(y, \hat{y}\right) = 1 - \frac{\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2}{\sum_{i=1}^{n}\left(y_i - \overline{y}_i\right)^2}, \quad \text{donde } \overline{y} = \frac{1}{n}\sum_{i=1}^{n} y_i.$$

Explained variance score: Mide la proporción de la variación de un conjunto de datos determinado:

$$\text{explained\_variance}(y, \hat{y}) = 1 - \frac{\text{Var}\{y - \hat{y}\}}{\text{Var}\{y\}}.$$

5. Experimentación y Resultados

La ejecución de los experimentos aplica la estrategia de la Figura 3. En una primera etapa se preparan los datos, preclasificandolos y escalándolos al rango $[0, 1]$, posteriormente se configura el algoritmo de aprendizaje automático y la validación cruzada 10-Fold. Al ejecutar el entrenamiento se aplican e imprimen las métricas de evaluación.

Cada partición de la validación cruzada genera un modelo entrenado que emite un conjunto de 21 predicciones. El valor final de tiempo y costo de cada proyecto consiste en el promedio de las estimaciones de los modelos entrenados. Al promediar todos los resultados se calcula la desviación estándar y se ejecutan las métricas de evaluación finales. La configuración de los dos modelos aplicados se muestran en las tablas del Cuadro 2a y 2b.

Tabla 2. Configuraciones para perceptrón multicapa y red neuronal recurrente simple.

Parámetro					
Configuración	Optimizador	Adam			
	Activación	Tanh			
	Capas Ocultas	2			
	Neuronas p/Capa	8			
	Tasa de aprendizaje	0.001			
	Función de pérdida (Loss)	Error Cuadrático Médio (MSE)			
	k-Fold Cross Validation	RepeatedKFold con 10-Fold y n_repeats=2			
	Normalización de datos	MinMaxScaler con valores entre (0,1)			
	Entradas (columnas)	Effort, Vi	Eff, Vi SizeEffort, SizeTime, SizeCost	Effort, Vi	Eff, Vi SizeEffort, SizeTime, SizeCost
	Salida	Tiempo		Costo	
Regularización	Dropout	No			
	Ruido Gaussiano	stddev=0.00001			
	No. Capas de ruido	2			
Máximo de épocas en entrenamiento		2000			

(a) Perceptrón Multicapa (MLP).

Parámetro					
Configuración	Optimizador	Adam			
	Activación	Relu			
	Activación recurrente	Tanh			
	Capas Ocultas	2			
	Tipo de celda recurrente	Simple RNN			
	Neuronas p/Capa RNN	40			
	Neuronas p/Capa Dense	10			
	Tasa de aprendizaje	0.001			
	Loss	Error Cuadrático Medio (MSE)			
	Normalización de datos	MinMaxScaler en el rango (0,1)			
	Entradas (columnas)	Effort, Vi	Eff, Vi SizeEffort, SizeTime, SizeCost	Effort, Vi	Eff, Vi SizeEffort, SizeTime, SizeCost
	Salida	Tiempo		Costo	
Regularización	Dropout recurrente	0.1			
	Ruido Gaussiano	stddev=0.00001			
	No. Capas de ruido	1			
Máximo de épocas en entrenamiento		900			

(b) Red Neuronal Recurrente Simple.

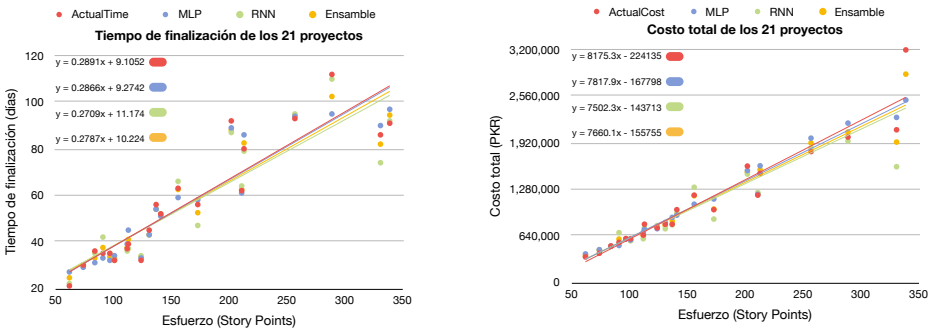


Fig. 4. Comparativa de estimaciones generadas con MLP, RNN y el ensamble.

Ambos casos consisten en redes neuronales de arquitectura pequeña ya que solo contienen capa de entrada, capa de salida y dos capas ocultas. Para mejorar la capacidad de generalización de las redes neuronales se aplica ruido gaussiano a la salida de las neuronas de las capas ocultas, lo que resulta en una estructura de seis capas aunque las capas de ruido solo modifican los valores de salida de las neuronas de la capa anterior y no contienen neuronas o pesos asociados.

Los modelos de redes neuronales se construyen utilizando el API de Keras, programadas en python y utilizando Scikit-learn para ejecutar la validación cruzada. La implementación se realiza de manera remota a través de la versión gratuita de Google Colab que brinda dos procesadores Intel(R) Xeon(R) CPU @ 2.20GHz, 13 GB de memoria RAM, 106 GB de espacio en disco y una GPU Tesla K80 con (x2) 2496 cores a una velocidad de 560 MHz y (x2) 12GB de memoria GDDR5 @ 2500 MHz. La ejecución del modelo basado en MLP con validación cruzada genera 20 estimadores debido a que se ejecuta una repetición de 10-Fold, por cada partición generada se entrena un modelo. La estimación de tiempo y costo corresponde al promedio de las estimaciones realizadas por todos los modelos entrenados. En el caso de la red neuronal recurrente la validación cruzada genera una ventana de n proyectos aumentando en tamaño en cada iteración.

Se ejecutan dos modelos de redes neuronales, uno para predecir tiempo de finalización y otro para las estimaciones de costo. La variación en número de neuronas indica

Tabla 3. Resultados del presente trabajo de investigación.

Parámetro		Modelo MultiLayer Perceptron (MLP)				Modelo Recurrent Neural Network (RNN)			
Datos	Entradas (columnas)	Effort, Vi	Eff, Vi SizeEffort, SizeTime, SizeCost	Effort, Vi	Eff, Vi SizeEffort, SizeTime, SizeCost	Effort, Vi	Eff, Vi SizeEffort, SizeTime, SizeCost	Effort, Vi	Eff, Vi SizeEffort, SizeTime, SizeCost
	Salida	Tiempo		Costo		Tiempo		Costo	
Prueba	Precisión (%)	**91.61**	**93.14**	**90.47**	**92.52**	**91.47**	**95.27**	**89.49**	**93.44**
	R^2	**0.9399**	**0.9614**	**0.9201**	**0.9341**	**0.9441**	**0.9751**	**0.9547**	**0.9658**
	MRE	0.0839	0.0686	0.0953	0.0748	0.0853	0.0473	0.1051	0.0656
	MSE	39.00	25.05	37,851,092,256	31,223,726,814	36.48	16.24	21,482,779,729	16,207,233,554
	RMSE	6.25	5.00	194,554	176,702	6.04	4.03	146,570	127,308
	Explained Variance	0.9402	0.9615	0.9201	0.9341	0.9477	0.9764	0.9584	0.9677
Máximo de épocas en entrenamiento		2000				900			
Promedio de épocas en entrenamiento		784	1196	473	700	-			

(a) Comparativa del uso de categorías de clasificación.

	Algoritmo	MLP	RNN	Ensamble	MLP	RNN	Ensamble
Datos	Entradas	Effort, Vi, SizeEffort, SizeTime, SizeCost			Effort, Vi, SizeEffort, SizeTime, SizeCost		
	Salidas	Tiempo			Costo		
Resultados	Predicción(%)	93.14	95.27	**95.58**	92.52	93.44	**95.58**
	Coef. determinación	0.9614	0.9751	**0.9855**	0.9341	0.9658	**0.9814**
	Error relativo MRE	0.0686	0.0473	**0.0442**	0.0748	0.0656	**0.0442**
	RMSE	5.00	4.03	**3.07**	176,702	127,308	**93,762**
	Explained Variance	0.9615	0.9764	**0.9856**	0.9341	0.9677	**0.9820**

(b) Métricas de evaluación.

Author	Estimación	Técnica	Precisión (%)	R^2	MMRE
Zia et al.	Tiempo	No Aplica	57.14	No incluida	0.0719
	Costo		61.90	No incluida	0.0576
Aditi Panda et al.	Tiempo	Cascade Correlation Neural Network	94.76	0.9303	0.1486
SM Satapathy et al.	Tiempo	Stochastic Gradient Boosting	95.24	No incluida	0.1632
Ch Prasada Rao et al.	Tiempo	Generalized Regression Neural Network	76.19	No incluida	0.0483
	Costo		76.19	No incluida	0.0276
Proyecto de investigación	Tiempo	Ensamble de Redes Neuronles	95.58	0.9855	0.0442
	Costo		95.58	0.9814	0.0442

(c) Resultados de la literatura.

que a mayor número de neuronas se reduce el número de épocas para entrenar los modelos, lo que reduce a su vez el tiempo de ejecución. El número mínimo de neuronas es igual a la dimensión de los datos de entrada y el número máximo de neuronas es de diez, al seguir incrementando el número de neuronas la red no mejora la precisión de la predicción. Los experimentos arrojan que ambas redes dan buenos resultados cuando se utilizan ocho neuronas en cada capa.

Como se observa en la Tabla 3a del Cuadro 3 las estimaciones realizadas por los algoritmos mejoran al clasificar los proyectos pues en los casos de tiempo y costo se logró reducir el error y aumentar los coeficientes de determinación. En la Figura 4 se grafican en rojo el tiempo de finalización real y el costo total de los 21 proyectos, en azul las estimaciones hechas con MLP, en verde las predicciones de la red neuronal recurrente y en amarillo el ensamble formado por ambas técnicas.

Como se puede observar en las rectas de regresión los mejores resultados se obtienen con la red neuronal de tipo perceptrón multicapa ya que fue el único algoritmo capaz de reducir el sobreentrenamiento. Sin embargo, el objetivo del modelo híbrido de estimación de esfuerzo consiste en aplicar distintas técnicas para obtener diversos puntos de comparación, por este motivo no se seleccionan solo los resultados del perceptrón multicapa.

Al tomar el promedio de las estimaciones realizadas por los dos algoritmos, se generan valores de tiempo de finalización y costo total que se comparan con los valores reales de los proyectos. En la tabla del Cuadro 4 se comparan los valores obtenidos al aplicar el modelo original propuesto en [8] con los valores centrales obtenidos por el ensamble.

Las primeras columnas corresponden a los valores de tiempo en días, ordenando

Tabla 4. Tabla comparativa de las diferencias de los valores predichos con los valores reales de cada proyecto y las estimaciones del modelo original.

No.	ActualTime	Zia et al (días)	ΔT Zia (días)	Ensamble	ΔT (días)	ActualCost	Zia et al. (PKR)	ΔC Zia (PKR)	Ensamble	ΔC (PKR)	¿Mejora Tiempo?	¿Mejora Costo?
1	63	58	5	63 ± 4	1	1,200,000	1,023,207	176,793	1,192,138 ± 116,558	7,863	Sí	Sí
2	92	81	11	88 ± 1	4	1,600,000	1,680,664	80,664	1,514,478 ± 24,438	85,522	Sí	No
3	56	52	4	53 ± 6	4	1,000,000	992,270	7,730	1,008,666 ± 138,458	8,666	Sí	No
4	86	87	1	82 ± 8	4	2,100,000	2,002,767	97,233	1,930,933 ± 339,441	169,067	No	No
5	32	29	3	34 ± 1	2	750,000	676,081	73,919	760,036 ± 21,891	10,036	Sí	Sí
6	91	95	4	95 ± 3	4	3,200,000	2,895,133	304,867	2,866,229 ± 356,537	333,772	Sí	No
7	35	29	6	34 ± 2	2	600,000	540,114	59,886	600,986 ± 3,033	986	Sí	Sí
8	93	84	9	95 ± 1	2	1,800,000	1,614,079	185,921	1,918,859 ± 66,126	118,859	Sí	Sí
9	36	35	1	33 ± 2	3	500,000	507,265	7,265	496,287 ± 1,980	3,713	No	Sí
10	62	66	4	63 ± 2	1	1,200,000	1,267,180	67,180	1,223,608 ± 12,456	23,608	Sí	Sí
11	45	41	4	43 ± 0	2	800,000	786,732	13,268	780,428 ± 42,805	19,573	Sí	No
12	37	39	2	38 ± 2	1	650,000	597,143	52,857	643,424 ± 43,528	6,576	Sí	Sí
13	32	35	3	33 ± 1	1	600,000	538,495	61,505	577,724 ± 4,662	22,277	Sí	Sí
14	30	26	4	30 ± 1	1	400,000	394,546	5,454	446,285 ± 2,913	46,285	Sí	No
15	21	22	1	25 ± 3	4	350,000	330,561	19,439	372,355 ± 18,420	22,355	No	No
16	112	103	9	103 ± 8	10	2,000,000	1,971,485	28,515	2,069,168 ± 122,723	69,168	No	No
17	39	40	1	41 ± 4	2	800,000	770,857	29,143	737,820 ± 9,324	62,181	No	No
18	52	50	2	52 ± 1	1	1,000,000	961,866	38,134	936,329 ± 11,850	63,671	Sí	No
19	80	76	4	83 ± 4	3	1,500,000	1,453,032	46,968	1,551,282 ± 52,870	51,282	Sí	No
20	56	51	5	54 ± 0	2	800,000	854,348	54,348	842,344 ± 51,656	42,344	Sí	Sí
21	35	34	1	38 ± 5	3	550,000	567,484	17,484	595,126 ± 88,635	45,126	No	No

el valor actual, el valor obtenido por el modelo original, la diferencia de días entre el modelo y el valor actual, el valor generado por el ensamble con su desviación estándar y la diferencia de días entre el valor real y el estimado por el modelo propuesto.

El modelo híbrido con categorías de clasificación mejora los tiempos en 15 de los 21 proyectos, las desviaciones estándar son pequeñas ya que solo uno de los proyectos tiene un desfase de dos semanas laborales o 10 días. Respecto a las estimaciones de costo la tabla ordena los valores de manera similar, comparando aquellos obtenidos por el modelo original, su diferencia, y los valores obtenidos con el modelo híbrido.

En este caso la mejora en las estimaciones ocurre solo en 9 de los 21 proyectos comparado con el modelo original. Sin embargo es importante resaltar que de los 12 proyectos donde no mejoró la predicción, la diferencia de costo no supera las 41,000 rupias. Solo en el proyecto número cuatro la diferencia de costo rebasó esta cantidad y las desviaciones estándar en tres proyectos superan las 100,000 rupias de diferencia entre el valor real y el valor central de la estimación.

Finalmente la comparativa con el estado del arte se muestra en la Tabla 3c mostrando que la aplicación de redes neuronales artificiales da resultados competitivos comparado con las técnicas aplicadas en la literatura.

6. Conclusión y trabajo a futuro

Los resultados obtenidos por el ensamble redujeron la desviación estándar de las predicciones brindando resultados confiables para el tiempo de finalización y el costo total de los proyectos. El mejor desempeño lo tuvo el perceptrón multicapa ya que la

Tabla 5. Ventajas y desventajas de los distintos algoritmos de aprendizaje automático empleados en la literatura y en la presente propuesta de investigación.

Técnica	Ventajas	Desventajas
Cascade Correlation Neural Network	Autoorganizadas	Con facilidad caen en sobreentrenamiento
	La arquitectura crece a medida que se ejecuta el entrenamiento	el algoritmo de aprendizaje define el tamaño correcto de la red
	Rapido entrenamiento	Contiene una capa oculta
Decision Tree	Adaptabilidad al contexto del problema	Con facilidad caen en sobreentrenamiento
	Se pueden generar distintos tipos de ensambles a bajo costo	Entre más debil el estimador, mayor el numero de modelos a entrenar en un ensamble
	Facil Interpretación y corto tiempo de entrenamiento, operación en tiempo real	
Generalized Regression Neural Network	Adaptabilidad al contexto del problema	No recomendables en problemas no lineales
	Se basa en la distancia euclidiana, la capa de patrón realiza operaciones elementales de suma y producto y la capa de decisión aplica cocientes.	Arquitectura fija de cuatro capas
MultiLayer Perceptron	Modelado flexible	Altos tiempos de entrenamiento
	Autoorganizadas	Costo computacional
	Adaptabilidad a problemas no lineales	Complejas para implementar desde cero
	Tolerancia a fallas y operación en tiempo real	Dificil interpretación de la ejecución
Recurrent Neural Network	Adaptabilidad a problemas no lineales y series temporales	Formateo especial de los datos de entrada
	Modelado flexible	Bajo desempeño en conjuntos de datos pequeños
	Autoorganizadas, tolerancia a fallas y operación en tiempo real	Arquitectura compleja en relación al MLP

validación cruzada permitió evaluar la capacidad de generalización de los algoritmos, siendo favorable para MLP y dando malos resultados para la red neuronal recurrente debido a que una ventana de tiempo pequeña no tiene los suficientes elementos para realizar las estimaciones de entradas desconocidos.

La preclasificación de proyectos mejoró la precisión de la predicción y también aumentó el coeficiente de determinación. Además, el ensamble de técnicas de aprendizaje automático también contribuye a la mejora de los resultados dando estabilidad a las predicciones y reduciendo la dispersión de los datos.

El ensamble tuvo resultados favorables teniendo una precisión de la predicción de

95.58 % con coeficientes de determinación mayores a 0.98. Como lo muestran las tablas comparativas de las predicciones emitidas, la desviación estándar de las estimaciones de tiempo no superan las dos semanas laborales y la mayoría de las estimaciones de costo tienen una desviación por debajo de las 100 mil rupias.

La presente propuesta en comparación con el modelo de estimación [8], brindó resultados confiables en las estimaciones de tiempo mejorando en 15 de los 21 proyectos y reduciendo la desviación estándar a no más de una semana laboral y tres días. Sin embargo, las estimaciones de costo no presentaron una mejora significativa ya que los mejores costos se obtienen al aplicar el modelo original [8].

Al comparar los resultados del perceptrón multicapa con las técnicas presentadas en la literatura se observa una mejora en la precisión con respecto a las redes neuronales de regresión generalizada. También superó a la red de tipo Cascade Correlation al obtener un coeficiente de determinación mayor en las estimaciones de tiempo. Una ventaja de la presente propuesta de investigación con respecto a los trabajos [1,6,3] es la facilidad de modelado y adaptabilidad de las redes neuronales, así como la capacidad de emitir predicciones de tiempo de finalización y costo total utilizando el mismo modelo. En el trabajo futuro, se buscará aplicar otras técnicas de aprendizaje automático como el algoritmo KNN, árboles de decisiones y autoencoders para generar estimaciones de tiempo y costo y robustecer el ensamble agregando todas estas técnicas al modelo híbrido ya existente.

Referencias

1. Panda, A., Satapathy, S. M., Rath, S. K.: Empirical validation of neural network models for agile software effort estimation based on story points. Procedia Computer Science, vol. 57, pp. 772–781 (2015) doi: 10.1016/j.procs.2015.07.474
2. Rojas, R.: Neural networks: a systematic introduction. Springer-Verlag (1996) doi: 10.1007/978-3-642-61068-4
3. Rao, C. P., Kumar, P. S., Sree, S. R., Devi, J.: An agile effort estimation based on story points using machine learning techniques. Bhateja, V., Tavares, J., Rani, B., Prasad, V., Raju, K., (eds). In: 2nd International Conference on Computational Intelligence and Informa-tics Advances in Intelligent Systems and Computing, vol. 712, pp. 209–219 (2018) doi: 10.1007/978-981-10-8228-3_20
4. Raslan, A. T., Darwish, N. R., Hefny, H. A.: Effort estimation in agile software projects using fuzzy logic and story points. In: 50th Annual Conference on statistics, computer scien-ces, and operation research, pp. 27–30 (2015)
5. Scott, E., Pfahl, D.: Using developers' features to estimate story points. In: International Conference on Software and System Process, pp. 106–110 (2018) doi: 10.1145/3202710.3203160
6. Satapathy, S. M., Rath, S. K.: Empirical assessment of machine learning models for agile software development effort estimation using story points. Innovations in Systems and Software Engineering, vol. 13, no. 2–3, pp. 191–200. (2017) doi: 10.1007/s11334-017-0288-z
7. Choetkiertikul, M., Dam, H. K., Tran, T., Pham, T., Ghose, A., Menzies, T.: A deep learning model for estimating story points. IEEE Transactions on Software Engineering, vol. 45, no. 7, pp. 637–656 (2019) doi: 10.1109/TSE.2018.2792473
8. Ziauddin, Tipu, S. K., Zia, S.: An effort estimation model for agile software development. Advances in Computer Science and its Applications, vol. 2, no. 1, pp. 314—324 (2012)

9. Coelho, E., Basu, A.: Effort estimation in agile software development using story points. International Journal of Applied Information Systems (IJAIS), vol. 3, no. 7, pp. 7—10 (2012) doi: 10.5120/ijais12-450574
10. Vyas, M., Bohra, A., Lamba, C. S., Vyas, A.: A review on software cost and effort estimation techniques for agile development process. International Journal of Recent Research Aspects, vol. 5, no.1, pp. 1–5 (2018)
11. Choudhari, J., Suman, U.: Story points based effort estimation model for software maintenance. Procedia Technology, vol. 4, pp. 761–765 (2012) doi: 10.1016/j.protcy.2012.05.124
12. Digital.ai.: The 14th Annual state of agile report is here. digital.ai software Inc. (2020)
13. Popli, R. N., Chauhan: Cost and effort estimation in agile software development. In: International Conference on Reliability Optimization and Information Technology (ICROIT), pp. 57–61 (2014) doi: 10.1109/ICROIT.2014.6798284
14. Hornik, K., Stinchcombe, M., White, H.: Multilayer feedforward networks are universal approximators. Neural Networks, vol. 2, no. 5, pp. 359–366 (1989) doi: 10.1016/0893-6080(89)90020-8
15. Fernandez-Diego, M., Mendez, E. R., Gonzalez-Ladron-De-Guevara, F., Abrahao, S., Insfran, E.: An update on effort estimation in agile software development: A systematic literature Review. IEEE Access, vol. 8, pp. 166768—166800 (2020) doi: 10.1109/ACCESS.2020.3021664
16. Garg, S., Gupta, D.: PCA based cost estimation model for agile software development projects. In: International Conference on Industrial Engineering and Operations Management (IEOM), pp. 1–7 (2015) doi: 10.1109/IEOM.2015.7228109
17. Malgonde, O., Chari, K.: An ensemble-based model for predicting agile software development effort. Empirical Software Engineering, vol. 24, no. 2, pp. 1017–1055 (2019) doi: 10.1007/s10664-018-9647-0
18. Durán, M., Juárez-Ramírez, R., Jiménez, S., Tona, C.: User story estimation based on the complexity decomposition using Bayesian networks. Programming and Computer Software, vol. 46, no. 8, pp. 569—583 (2020) doi: 10.1134/S0361768820080095
19. Gultekin, M., Kalipsiz, O.: Story point-based effort estimation model with machine learning techniques. International Journal of Software Engineering and Knowledge Engineering, vol. 30, no. 1, pp. 43–66 (2020) doi: 10.1142/S0218194020500035