ISSN 1870-4069

Modelo inteligente para determinar el esfuerzo de desarrollo de software mediante machine learning

Edgar Gonzalo Cossio Franco[1], Mariela Chávez Marcial[2], Emma Yesenia Rivera Ruiz[2], Yotziri Paloma Pérez Rios[2], Marco Julio Franco Mora[2]

[1] Instituto de Información Estadística y Geográfica de Jalisco, México

[2] Tecnológico Nacional de México / ITS de Ciudad Hidalgo, México

edgar.cossio@iieg.gob.mx, mchavez@cdhidalgo.tecnm.mx, {yess.rivera.1311, ypelekai, darkmarksdoe}@gmail.com

Resumen. La estimación de costos, esfuerzo, personal y tiempo siempre ha sido un tema fundamental para cualquier desarrollo de software, de cualquier tamaño, dado a que desarrollar un producto consta de múltiples variables y tomar la decisión correcta a la hora de realizar y cotizar un proyecto conlleva dinero, personal y tiempo, para esto existen múltiples herramientas. COCOMO junto a Puntos de Función son las herramientas que contemplan variables importantes respecto a un desarrollo y que se utilizarían para la fase de entrenamiento. Este trabajo abarca la creación de una herramienta que mediante la implementación de Machine Learning determina el esfuerzo de desarrollo.

Palabras clave: COCOMO, Machine Learning, Predicción, Puntos de función.

Intelligent Model for Determining Software Development Effort using Machine Learning

Abstract. Estimating costs, effort, personnel and time has always been a fundamental issue for any software development, of any size, since developing a product consists of multiple variables and making the right decision when making and pricing a project It takes money, staff and time, for this there are multiple tools. COCOMO together with Function Points are the tools that contemplate important variables regarding a development and that would be used for the training phase. This work covers the creation of a tool that through the implementation of Machine Learning determines the development effort.

Keywords: COCOMO, Machine Learning, Prediction, Functions Points.

1. Introducción

Entregas a tiempo, control de presupuesto y alta calidad de los productos son objetivos críticos para la administración de proyectos de software. El costo, la calidad y la entrega del software se ven afectados por la precisión de la estimación del esfuerzo del software [13]. Las prácticas de ingeniería de software tienen características específicas que diferencian este campo de la ingeniería tradicional.

En particular, varios factores afectan la estimación del esfuerzo de software en organizaciones y proyectos, incluidos procesos de software inconsistentes y definiciones de medición en proyectos, diversidad sustancial entre proyectos, y diferencias extremas en los tamaños de los productos.

En consecuencia, estas situaciones crean desafíos en la práctica de la estimación del esfuerzo del software, lo que dificulta el grado de rendimiento en la precisión de la estimación. Muchos estudios se han centrado en el desarrollo de modelos y técnicas de estimación de costos de software [9].

Estos incluyen modelos algorítmicos, como COCOMO [3], SLIM [8], SEER-SEM [4] y técnicas de machine learning [27]. Estos modelos y las técnicas se han introducido y utilizado en la industria del software.

En todos los modelos, las entradas y las relaciones son específicas del dominio y dependen totalmente de la opinión de los expertos. Por esta razón, estos modelos tienden a funcionar mal o incluso fallan cuando se intenta cambiar los límites de su aplicación. En tal escenario, existe la necesidad de una técnica que pueda sustituir el juicio experto. Debido a que la mayoría de los estudios en los que diferentes profesionales de software estiman la misma tarea de desarrollo de software informan una gran variación de las estimaciones de esfuerzo. [19].

Entre otras técnicas, Machine Learning, representa una interesante perspectiva que, mediante, la recopilación de datos, la adquisición de conocimientos, la clasificación, el reconocimiento de patrones conforma un proceso que permite predecir escenarios.

Cuando se estima el tamaño del software, se pueden obtener varias métricas, como el esfuerzo, el tiempo y el costo necesario para desarrollar un proyecto, costos de mantenimiento, métricas de calidad y productividad del equipo [5]. La pregunta es qué tan bien funcionan los métodos propuestos por estos modelos en lo que respecta al error de medición de los datasets y las estadísticas comparativas de rendimiento con otras técnicas.

El objetivo general de la presente propuesta es desarrollar un modelo inteligente para determinar el esfuerzo de software basado en un algoritmo supervisado por machine learning haciendo uso de un dataset público. La propuesta es utilizar COCOMO I junto a Puntos de Función para la fase de entrenamiento. Luego, se determina la efectividad de la técnica de aprendizaje automático para el campo de estimación del esfuerzo y se extrae la conclusión posterior al análisis.

Con este trabajo de investigación, se intenta no solo desarrollar una mejor oportunidad para la estimación de costos, sino que también se pretende aplicar machine learning a la ingeniería de software. Dicha fusión de áreas de conocimiento entre Inteligencia Artificial (IA) e Ingeniería de Software (IS) se conoce como SBSE (ingeniería de software basada en búsqueda).

1.1. Estimación

Ha habido una serie de esfuerzos entre desarrolladores de software, ingenieros e investigadores para definir modelos de estimación del esfuerzo del proyecto, destacándose los modelos de análisis de puntos de función y COCOMO [6, 30]. Estos modelos calculan la distancia entre el proyecto de software que se estima y el histórico de proyectos de software (entradas como el tamaño, el número de funciones de atributos y otros generadores de costos), recuperando los más similares para definir una estimación de esfuerzo [31].

La Tabla 1 muestra los trabajos publicados de Análisis de puntos de función de Keremer [32], Chandrasekaran and Kumar [33] y Arnuphaptrairong [34]. Los trabajos de COCOMO de MayaZaki and Mori [35], Keremer [32], Chandrasekaran and Kumar [33]. En los trabajos se incluye la media del error relativo utilizada para evaluar la precisión de los modelos de estimación de costos [30], observándose que solo el estudio de Chandrasekaran and Kumar [33] presenta un porcentaje bajo de error en la estimación.

Tabla 1. Estudios de los modelos de análisis de puntos de función y COCOMO [30].

No.	Modelo de Estimación	Autor	Año de publicación	Media del error relativo (%)
1	Análisis de puntos de función	Kemer	1987	102.74
2	Análisis de puntos de función	Chandrasekaran and Kumar	2012	13.8
3	Análisis de puntos de función	Arnuphaptrairong	2013	82
4	COCOMO	MayaZaki and Mori	1985	165.6
5	COCOMO	Kemerer	1987	583.82
6	COCOMO	Chandrasekaran and Kumar	2012	8.4

Múltiples técnicas y paradigmas están disponibles en la industria y el machine learning es uno de los enfoques crecientes en esta área [15], la introducción de algoritmos de aprendizaje automático combinados con el modelo COCOMO han mejorado el rendimiento de la predicción de los modelos en comparación con enfoques estadísticos, ya que permite el ajuste de los parámetros y el aprendizaje de los datos de la experiencia [36]. La tabla 2 muestra el resumen de las técnicas de aprendizaje automático aplicadas del año 2007 al 2018 [37].

La Tabla 3 muestra la precisión de predicción entre las técnicas Linear Regression (LR), Artificial neural network (ANN), Support Vector Regression (SVR) y K-Nearest

Tabla 2. Técnicas de aprendizaje automático aplicadas del año 2007 al 2018 [37].

Year	Techniques	Dataset
2007	NeuroFuzzy	NASA&COCOMO
2015		
2008	PSO	NASA
2008	Regression Techniques	COCOMO
2010	Genetic Algorithm	-Turkish % Industry data set -COCOMO & NASA
2013.2015		
2011	ANN	COCOMO & NASA, PROMISE repository
2014		
2011	Fuzzy Logic	NASA
2018		
2013	PSO-ANN-COCOMO II	COCOMO I &NASA93
2014	Radial Basis function NN	COCOMO & NASA
2014	Bayesian with PSO	NASA 93
2014	PCA with ANN	COCOMO 81
2014	Simulated Anncaling	NASA
2014	Hybrid COCOMO and FP	NASA
2015	Bee Colony Optimization	Interactive Voice Response Softwareproject datset
2016	Cuckoo, Hybrid Cuckoo Optimization	NASA60, NASA63, NASA93, MAXWELL, KEMERER and MIYAZAKI
2017		
2018		
2016	COCOMO TLBO	NASA
2016	Differential Evolution Method	NASA
2016	Hybrid PSO-ANFIS	NASA
2016	Bat Algorithm	NASA 93
2016	COCOMO-GA	NASA
2017	ANN-Dragonfly Algorithm	NASA

Neighbors (KNN) combinados con COCOMO, estableciendo el porcentaje relativo de error [28]. Se observa el algoritmo ANN con mejor rendimiento.

El resultado del trabajo actual radica en la ecuación empírica de COCOMO I y el algoritmo de aprendizaje supervisado de regresión lineal.

COCOMO I

El modelo de esfuerzo se puede escribir en la siguiente forma general:

$$PM = A * Size^{B} * \prod_{I=1}^{P} EM_{I}, \quad (1)$$

Ec. 1 Fórmula general COCOMO [24], donde:

PM = Esfuerzo estimado calculado en meses,

Tabla 3. Comparación de rendimiento entre LR, ANN, SVR y KNN [28].

Performance Comparison	LR	ANN	SVR	KNN
Correct Predicition on Training Data	%74	%87	%95	%68
Correct Prediction on Testing Data	%60	%95	%80	%60

A = Constante multiplicativa,
Size = Tamaño estimado del software, medido en KSLOC,
B = Factores de escala,
EM = Multiplicadores de esfuerzo.

En COCOMO 81, el término B es una constante exponencial que generalmente es mayor que 1.0, lo que indica deseconomías de escala.

Regresión lineal

El modelo para la regresión lineal múltiple se expresa como:

$$y_i = \boldsymbol{\beta}_0 + \beta_i x_{i1} + \ldots + \beta_p x_{ip} + \epsilon_i. \quad (2)$$

Ec. 2 Fórmula general de Regresión Lineal Múltiple [24], donde $\beta_0, \beta_1, \ldots, \beta_p$ son los coeficientes de regresión, y εi es el término de error para la i-ésima observación.

1.2. COCOMO

La estimación del esfuerzo es un desafío en cada proyecto de software [7]. Las estimaciones afectarán los costos y las expectativas según el cronograma, la funcionalidad y la calidad. Si bien las estimaciones de expertos se usan ampliamente, son difíciles de analizar y la calidad de la estimación depende de la experiencia de expertos de proyectos similares.

Alternativamente, se pueden utilizar modelos de estimación más formales. Tradicionalmente, el tamaño del software estimado en la cantidad de líneas de código fuente (SLOC), puntos de función (FP) y puntos de objeto (OP) se utilizan como entrada a estos modelos, p. COCOMO y COCOMO II [2].

COCOMO desarrollado por Barry Boehm [3] permite realizar estimaciones en función del tamaño del software, y de un conjunto de factores de costo y de escala. Los factores de costo describen aspectos relacionados con la naturaleza del producto, hardware utilizado, personal involucrado, y características propias del proyecto.

El modelo COCOMO 81 identifica 15 multiplicadores de esfuerzo (ver Tabla 4), mientras que COCOMO II usa 17 en su modelo Post-Arquitectura. Además, podemos transformar los modelos COCOMO en una forma lineal aplicando la transformación logarítmica de ambos lados con la siguiente ecuación (COCOMO 81):

$$log(PM) = \beta_0 + \beta_1 \log(\text{Size}) + \beta_2 \log(EM_1) + \ldots + \beta 16 \ \log(EM_{15}). \quad (3)$$

Ec. 3 es transformación algorítmica de COCOMO I [24].

Tabla 4. Valores de costo COCOMO 81 [24].

Cost Driver	Very Low	Low	Nominal	High	Very High	Extra High
ACAP	1.46	1.19	1.00	0.86	0.71	
PCAP	1.42	1.17	1.00	0.86	0.70	
AEXP	1.29	1.13	1.00	0.91	0.82	
MODP	1.24	1.10	1.00	0.91	0.82	
TOOL	1.24	1.10	1.00	0.91	0.83	
VEXP	1.21	1.10	1.00	0.90		
LEXP	1.14	1.07	1.00	0.95		
DATA		0.94	1.00	1.08	1.16	
CPLX	0.70	0.85	1.00	1.15	1.30	1.65
TURN		0.87	1.00	1.07	1.15	
VIRT		0.87	1.00	1.15	1.30	
STOR			1.00	1.06	1.21	1.56
TIME			1.00	1.11	1.30	1.66
RELY	0.75	0.88	1.00	1.15	1.40	
SCED	1.23	1.08	1.00	1.04	1.10	

1.3. Puntos de función

El tamaño es uno de los atributos del software y se define como un conjunto completo de funcionalidades comerciales que el software proporciona cuando se implementa el método de análisis de puntos de función el cual fue introducido por primera vez por Albrecht en [10, 11]. En este método, el Software se divide en Funciones de datos y Transacciones de datos. Las funciones de datos se dividen en archivos lógicos internos, archivos de interfaz externa.

Cada uno está hecho de tipos de datos y elementos de registro. La función de transacciones se realiza con entradas externas, salidas externas y consultas externas (ver Figura 2).

El tamaño del software se calcula después de identificar la complejidad de los puntos de función y encuentra su suma global que se conoce como puntos de función no ajustados. Los puntos de función luego se ajustan considerando los aspectos técnicos y operativos del proyecto (ver Figura 1).

1.4. Aprendizaje máquina en estimación de software

Para estimar el esfuerzo, la duración y el costo del software, se implementan enfoques algorítmicos y no algorítmicos, son conocidos como modelos de aprendizaje automático para predecir el software EDC (Estimation of effort, Cost and Duration) [1].

Fig. 1. Transformación de requisitos en puntos de función o características [22].

Type of Input	Count		
	Simple	Average	Complex
Number of external inputs	3	4	6
Number of external outputs	4	5	7
Number of external queries	3	4	6
Number of internal logical files	7	10	15
Number of external interface files	5	7	10

Fig. 2. Recuento inicial de puntos de función [22].

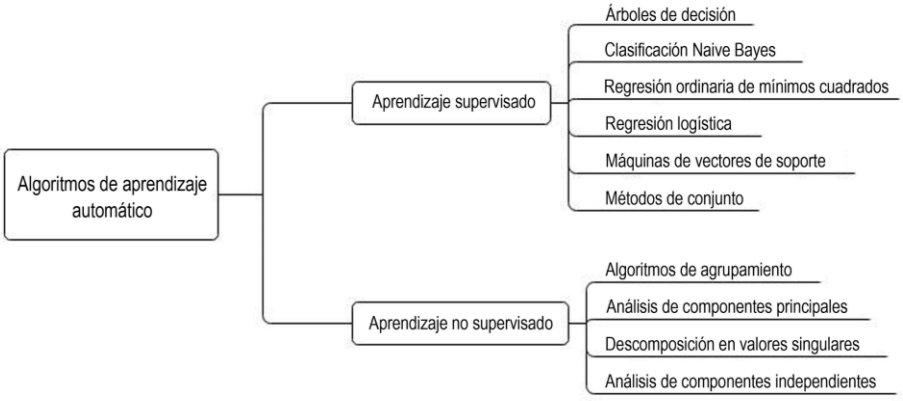

Fig. 3. Algoritmos esenciales en Machine Learning. Creación propia a partir de Raona [23].

El proceso de aprendizaje en ML (Machine Learning) puede partir de una tarea descriptiva o predictiva elegida de una tarea confiable, no supervisada o supervisada. El primero se basa en patrones que no están vinculados a ninguna variable en el conjunto de datos de entrenamiento [17] para fines tales como agrupamiento, reglas de asociación o detección de anomalías.

El aprendizaje supervisado debe tener entradas definidas explícitamente correspondientes a salidas conocidas por una variable objetivo determinada [18] que se utiliza para la clasificación o predicción.

Para ambos hay numerosos algoritmos disponibles que pueden aplicarse según el resultado deseado.

2. Problemática

Cumplir en tiempo y forma el desarrollo de un proyecto de software depende de la cantidad de esfuerzo destinada al mismo, lo cual influye directamente en el tiempo que deba invertirse para finalizarlo con éxito.

Actualmente las empresas que se dedican al desarrollo de software emplean técnicas para estimaciones con la finalidad de predecir variables que permitan proporcionar datos certeros sobre los recursos que serán empleados durante el ciclo de vida de un proyecto. La estimación en proyectos de software es una tarea extremadamente compleja, que requiere, entre otras cosas, disponer de información detallada del proyecto o de los proyectos a estimar, realizar una primera planificación del proyecto y conocer los recursos disponibles [21].

Dependiendo de las técnicas empleadas pueden encontrarse variaciones en los resultados que se arrojan a raíz de su aplicación, provocando la alteración de los datos de vital importancia que pueden llegar a afectar de manera considerable.

El surgimiento de nuevos sistemas integrados a tecnologías emergentes que emplean IA y ML, están tomando la delantera ofreciendo a las empresas la oportunidad de realizar estimaciones de software de manera sencilla, rápida y eficiente, ya que sus motores son entrenados con datos basados en resultados de proyectos de éxito ya finalizados; lo cual provee a dichos sistemas de predicciones certeras para calcular con base en datos del nuevo proyecto un conjunto de salidas más aproximadas a la inversión requerida en cuestión de tiempo, esfuerzo y recursos.

3. Propuesta

El sistema propone mejorar las estimaciones de las variables de duración y consumo de recursos al momento de desear realizar un desarrollo de software; esto basándose en datos obtenidos de proyectos de éxito ya concluidos, para lo cual se pretende emplear algoritmos predictivos de ML como lo es Regresión Lineal.

La intención es ofrecer una alternativa a los problemas de estimación ya mencionados a lo largo del documento, la solución propuesta es ofrecer variables de estimación con un porcentaje alto de confiabilidad que a la vez coadyuve a la agilización en el proceso de generación de datos a partir de la utilización de técnicas de inteligencia artificial para sustituir la tarea de cálculos manuales.

4. Metodología

Como parte de la metodología se utilizó regresión lineal dado que la estimación del esfuerzo del software se reconoce como un problema de dicha naturaleza [12]. A continuación, se muestra la secuencia de pasos en la construcción del modelo. (Ver Figura 4).

4.1. Dataset

El término dataset se refiere a un archivo que contiene uno o más registros [20]. El dataset en esta investigación fue recopilado por Jairus Hihn y donado a The PROMISE

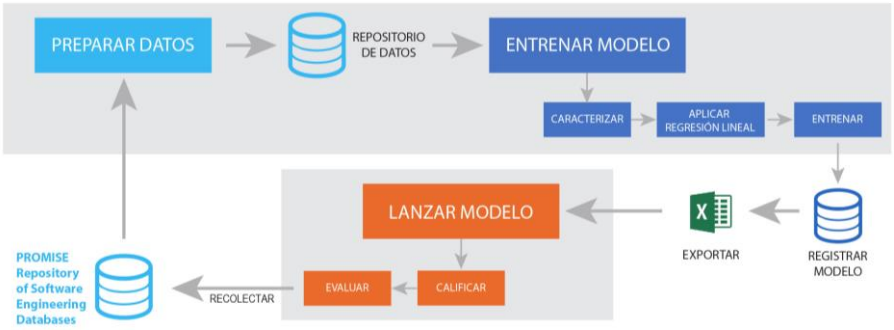

Fig. 4. Metodología para la construcción del Modelo inteligente. [26].

Repository of Software Engineering Databases [16]. Cubre información sobre 93 proyectos de desarrollo de software documentados por la NASA, entre los años 1971 y 1987, los cuales incluyen 24 atributos que cubren la información de los proyectos, 15 atributos discretos estándar de COCOMO I en el rango Very_Low hasta Extra_High, 7 que describen el proyecto; una línea de medida de código y un campo objetivo del esfuerzo real en persona meses.

Los proyectos de la NASA se utilizan para entrenar y probar el uso de técnicas inteligentes de estimación de esfuerzo en la producción de software [28, 29]. Se puede argumentar el overfitting en este caso con el hecho de que se utilizó ese dataset solamente para entrenamiento y en la actualidad los datos que alimentan es decir el 20% es para el testing.

4.2. Técnicas aplicadas

La construcción del modelo y el entrenamiento se realizó por medio de Microsoft Azure Machine Learning Studio (Classic). Azure ML [25] es una herramienta colaborativa que se basa en el manejo interactivo de componentes para crear, probar e implementar soluciones de análisis predictivo en datos; está diseñado para trabajar con datos tabulares, como datos de texto delimitados o datos estructurados de una base de datos.

Azure Machine Learning Studio publica modelos como web services que pueden generarse desde la misma herramienta. Para el proyecto se realizó la descarga del archivo excel que al ser ejecutado en local despliega un apartado del lado derecho que indica que el archivo se encuentra listo para la conexión con el web service de Azure ML que se creó, los nuevos datos de entrada son agregados en la hoja de cálculo de acuerdo al orden de las categorías que incluidas en el dataset usado, desde donde serán tomados para entregar el resultado de la predicción.

En la figura 5 se muestra el proceso de construcción del modelo para el cual fue necesario:

- Depurar el conjunto de datos, seleccionando las columnas numéricas,
- Fraccionar el conjunto de datos en un 0.8 para el entrenamiento,

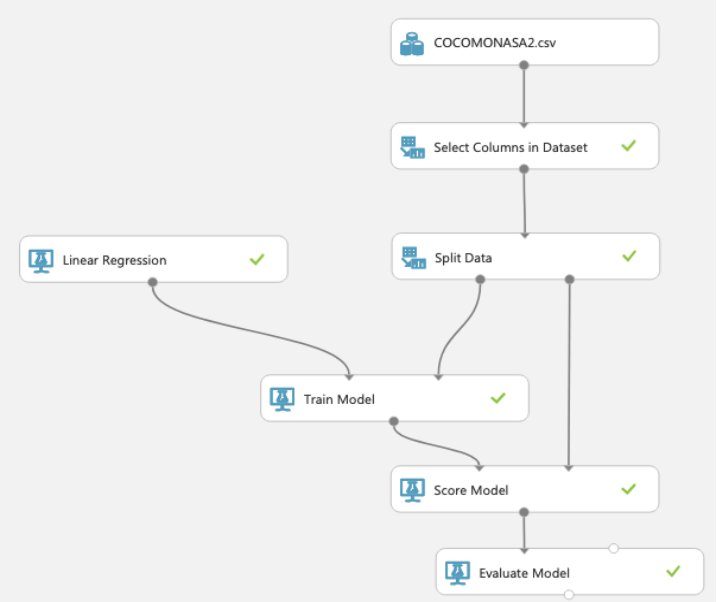

Fig.5. Modelo inteligente. Creación propia.

Tabla 5. Headers y contenido del dataset.

Headers		**Contenido (Valores)**	
Id	projectname	1-100	Nombre del Proyecto
System type	NASA center	Ground System(g), Flight System(f)	1 to 5
year	mode	Año del proyecto	Embedded, Organic, Semidetached
Category		Avionics, application_ground, avionicsmonitoring, batchdataprocessing, communications, datacapture, launchprocessing, missionplanning, monitor_control, operatingsystem, realdataprocessing, science, simulation, utility.	

- Aplicar regresión lineal,
- Entrenar el modelo mediante la variable KSLOC,
- Calificar el modelo,
- Evaluar el modelo.

4.3. Prueba y entrenamiento del modelo propuesto

Como parte del proceso de entrenamiento y pruebas, el dataset fue sometido a una serie de pasos, los cuales se indican en la siguiente lista, con los cuales fue posible obtener resultados.

Tabla 6. Atributos generales y su descripción.

Siglas	Nombre	Descripción
RELY	Required Software ReliabiLitY	Prueba de Confianza del Software en un periodo de tiempo y ambiente especifico.
DATA	DATAbase size	Tamaño de la base de datos.
CPLX	Process ComPLeXity	Evalúa las tareas o eventos por los que fluye un producto o actividad antes de pasar al estado de salida.
TIME	TIME Constraint	Restricción de Tiempo para CPU, acción dinámica que solicita un thread y un scheduler por un intervalo de tiempo.
STOR	Main MemORy ConSTraint	Limita el tamaño de la memoria utilizada buscando mejorar el rendimiento.
VIRT	VIRTual Machine Volatility	Refiere al complejo de Hardware y Software.
TURN	Computer TURNaround time	Tiempo de respuesta de la computadora.

Tabla 7. Atributos del personal y su descripción.

Siglas	Nombre	Descripción
ACAP	Analyst CAPalibility	Prueba de Confianza del Software en un periodo de tiempo y ambiente especifico.
AEXP	Application EXPerience	Toma experiencia de los campos UI, UX Networking, Middleware, así como lenguajes y herramientas.
PCAP	Programming CAPability	Toma la productividad de una persona y su capacidad para construir software.
VEXP	Virtual Machine EXPerience	Mide la experiencia del personal en el uso, configuración y familiaridad con máquinas virtuales.
LEXP	Programming Language EXPerience	Refiere a la experiencia (Intern, Jr., SSr., Sr.) respecto a los lenguajes de programación del proyecto.
MODP	MODern Programming Practices	Prácticas de programación modernas se puede entender mejor como “Practicas maduras de Ingeniería de Software”.
TOOL	Use Software TOOLS	Herramientas que saben y pueden usar tanto en testing, desarrollo y maquetación.
SCED	Required Development SChEDule	Esta variable es única e importante, ya que un proyecto con un tiempo acelerado requerirá más esfuerzo que un proyecto en su tiempo óptimo.

Tabla 8. Atributos de desarrollo y su descripción.

Siglas	Nombre	Descripción
KSLOC	Thousan (K) Lines of Code	Esta variable es importante, ya que un proyecto con un tiempo acelerado requerirá más esfuerzos qwue un proyecto en su tiempo óptimo.
Effort	Effort per Month	Este último valor, mostrara el esfuerzo requerido en meses. Cada mes es equivalente a 152, que incluye horas de desarrollo y gestión (Development & Management).

Etapas:

1.- Pre procesar datos (Datos necesarios),

2.- Garantizar valores numéricos (int, float),

3.- Normalizar etiquetas,

4.- Evitar datos vacíos.

El conjunto de datos sometido a entrenamiento y prueba se describe a continuación: En la tabla 5, podemos apreciar los primeros headers, los cuales hablan respecto a información de los proyectos.

A continuación, se muestran 15 atributos de COCOMO I los cuales tendrán un rango desde Very Low hasta Extra High, los rangos posibles son:

vl	- Very Low
l	- Low
n	- Nominal
h	- High
vh	- Very High
xh	- eXtra High

En la tabla 6 se puede ver los atributos generales del proyecto. En la tabla 7 se muestran los atributos del personal. En la tabla 8 podemos ver los atributos de desarrollo del proyecto.

5. Resultados

Este trabajo muestra como resultado la implementación de un modelo inteligente que permite determinar el esfuerzo de desarrollo de software mediante machine learning. Los resultados del experimento al cual fue sometido el conjunto de datos bajo el enfoque de regresión lineal fue prometedor dado que en el mejor resultado mostró una precisión del .40 en el coeficiente de determinación. Por supuesto es importante hacer hincapié en la búsqueda e implementación de otras técnicas y la modificación en cuestión de configuración dentro del modelo propuesto, para esperar como resultado el incremento de la exactitud que se busca.

6. Conclusiones

Se resalta que la implementación de Inteligencia Artificial y Machine Learning comenzará a convertirse en un soporte primordial en el desarrollo de herramientas que coadyuven en la ejecución de procesos dentro del área de la ingeniería de software. Mediante el desarrollo de sistemas inteligentes como el propuesto en el presente trabajo se busca impulsar la idea de desarrollos que además de ser innovadores se centren en

la necesidad de garantizar resultados certeros, mejorando de esta forma la estimación de recursos durante el proceso de desarrollo de software.

7. Trabajo futuro

Se contempla comunicar el modelo mediante un web service customizado que desde el servidor se comunique con Python mientras que el cliente permitirá la entrada de datos a través de un formulario construido a partir de HTML, CSS y JS.

La finalidad del desarrollo web es realizar el intercambio de datos de entrada en la UI y la salida de los resultados generados por el modelo, mejorando la UX.

Además de que considera ideal que sea de uso libre lo cual implica el pago de un servidor público para albergar el sistema y ofrecerlo a cualquier persona que desee realizar estimaciones con base en los datos de su proyecto.

Referencias

1. Prasad, B.M.G., Sreenivas, P.V.S.: An implementation of software effort duration and cost estimation with statistical and machine learning approaches. International Journal of Computer Engineering and Technology, 10(1), pp. 81−93 (2019)
2. Boehm, B., Clark, B., Horowitz, E., Westland, C., Madachy, R., Selby, R.: Cost models for future software life cycle processes: COCOMO 2.0. USC Center for Software Engineering (1995)
3. Boehm, B., et al.: Software Cost Estimation in COCOMO II. Prentice Hall (2000)
4. Galorath, D., Evans, M.W.: Software sizing, estimation, and risk management. Auerbach Publications (2006)
5. Parthasarathy, M.A.: Practical software estimation. Addison-Wesley, pp. 47−72 (2007)
6. Nasir, M.: A survey of software estimation techniques and project planning practices. Proc. 7th ACIS Int. Conf. on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD), pp. 305−10, IEEE Computer Society (2006)
7. Parastroo Mohagheghi, Bente Anda, Reidar Conradi: Effort estimation of use cases for incremental large-scale software development. 27th International Conference on Software Engineering (2005)
8. Putnam, L.H.: A general empirical solution to the macro software sizing and estimating problem. IEEE Transactions on Software Engineering, 4, pp. 345−361 (1978)
9. Du, W.L., Capretz, L.F., Nassif, A.B., Ho, D.: A hybrid intelligent model for software estimation. Journal of Computer Science, 9 (2013)
10. Albrecht, A.: Measuring application development productivity. In IBM Application Development Symp., 1079, pp. 83−92 (1979)
11. Albrecht, A.J., Gaffney, J.E.: Software function, source lines of code, and development effort prediction: a software science validation. IEEE transactions on software engineering, 6, pp. 639−648 (1983)
12. Adriano, L.O., Petrónio, L.B., Ricardo, M.F., Márcio, C.: GA-based method for feature selection and parameters optimization for machine learning regression applied to software effort estimation. Information & Software Technology, 52(11), pp. 1155−1166 (2010)
13. Nassif, A.B., Capretz, L.F., Ho, D.: Software estimation in the early stages of the software life cycle. Nanded, Maharashtra, India, pp. 5−13 (2010)
14. Peraphon Sophatsathit: An exploratory survey of phase-wise project cost estimation techniques. Advanced Virtual and Intelligent Computing (AVIC) Center (2014)

15. Somya Goyal, Anubha Parashar: Machine learning application to improve COCOMO model using neural networks. Information Technology and Computer Science, 3, pp. 35−51 (2018)
16. Hihn, J.: COCOMO NASA 2 / Software cost estimation. The Promise Repository of Software Engineering Databases (2006)
17. Linoff, G.S., Berry, M.J.A.: Data mining techniques: for marketing sales, and customer relationship management. John Wiley & Sons (2011)
18. Cios, K., Pedrycz, W., Swiniarski, R., Kurgan, L.: Data mining a knowledge discovery approach. Springer (2007)
19. Grimstad, S., Jorgensen, M.: Inconsistency of expert judgment-based estimates of software development effort. J Syst Software, 80(11), pp. 1770−1777 (2007)
20. IBM Knowledge Center: What is a dataset?. https://www.ibm.com/support/knowledgecenter/zosbasics/com.ibm.zos.zconcepts/zconc_datasetintro.htm (2016)
21. Ruíz, Y.C., Cordero, M.D.: Software projects estimation integrating Bohem and Humphrey method's. Revista Cubana de Ciencias Informáticas, 7(3) (2013)
22. Lawrence, P.S., Wu, F., Lewis, R.: Software cost estimation and sizing methods: issues and guidelines. Santa Monica (2005)
23. Raona: Visible body: Los 10 Algoritmos Esenciales en Machine Learning. https://www.raona.com/los-10-algoritmos-esenciales-machine-learning/ (2017)
24. Nguyen, V., Steece, B., Boehm, B. A constrained regression technique for cocomo calibration. In Proceedings of the Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM'08). Association for Computing Machinery, pp. 213–222 (2008)
25. Microsoft: Visible body: What is Machine Learning Studio (classic)?. https://docs.microsoft.com/es-es/azure/machine-learning/studio/what-is-ml-studio (2020)
26. MCKnight, W.: Visible body: GIGAOM: Delivering on the Vision of MLOps v1.0 A maturity-based approach. https://gigaom.com/report/delivering-on-the-vision-of-mlops/ (2020)
27. Scikit-Learn: https://scikit-learn.org/stable/ (2020)
28. Zeynab Abbasi Khalifelua, F.S.: Comparison and evaluation of data mining techniques with algorithmic models in software cost estimation. ScienceDirect, pp. 65−71 (2012)
29. Sheta, A.: Estimation of the COCOMO model parameters using genetic algorithms for nasa software projects. Journal of Computer Science, pp. 118−123 (2006)
30. Arnuphaptrairong, T.: A literature survey on the accuracy of software effort estimation models. International MultiConference of Engineers and Computer Scientists, (2016)
31. Sweta-Kumari, S.P.: Comparison and analysis of different software cost estimation methods. International Journal of Advanced Computer Science and Applications, pp. 153−157 (2013)
32. Kemerer, C.F.: An empirical validation of software cost estimation models. Communication of the ACM, 30(5), pp. 416−429 (1987)
33. Chandrasekaran, R., Kumar, R.V.: On the estimation of the software effort and schedule using constructive cost Model-II and function point analysis. International Journal of Computer Applications, 44(9), pp. 3844 (2012)
34. Arnuphaptrairong, T.: Early stage software effort estimation using function point analysis: an empirical validation. International Journal of Design, Analysis and Tools for Integrated Circuits and Systems, 4(1), pp. 15−21 (2013)
35. MayaZaki, Y., Mori, K.: COCOMO Evaluation and tailoring. In Proceeding of the 8th International Conference on Software Engineering of the IEEE, pp. 292−299 (1985)
36. Satapathy, S.M.: Effort estimation methods in software development using machine learning algorithms. Department of Computer Science and Engineering, National Institute of Technology Rourkela (2016)

37. Siti Hajar Arbain, N.A.: Adoption of machine learning techniques in software effort estimation: an overview. IOP Conference Series: Materials Science and Engineering (2019)