COMPUTACIÓN E INFORMÁTICA
Recibido 17/06/2019
Aceptado 20/07/2019
ReCIBE, Año 8 No. 1, octubre 2019

Estimación del esfuerzo de proyectos de software con algoritmos de aprendizaje de máquinas

Software project effort estimation with machine learning algorithms

Jesús Iván Saavedra Martínez[1]
ivan.saavedra@ciencias.unam.mx

María Guadalupe Elena Ibarguëngoitia González[1]
gig@ciencias.unam.mx

Gibran Fuentes Pineda[1]
gibranfp@unam.mx

[1]*Universidad Nacional Autónoma de México Ciudad universitaria, Ciudad de México, México*

Resumen

La estimación del esfuerzo de proyectos de software es el proceso de predecir el esfuerzo requerido para desarrollar o mantener un sistema de software. Desarrollar modelos de estimación y técnicas apropiadas es fundamental para evitar pérdidas causadas por una estimación deficiente, donde se termina invirtiendo más esfuerzo del estimado.

La precisión y confiabilidad de las estimaciones desempeñan un papel muy importante en la gestión de proyectos, ya que permiten un monitoreo y control factible para garantizar que los proyectos se terminarán de acuerdo a lo planeado.

Este documento presenta una comparación entre modelos de estimación tradicionales basados en modelos estadísticos y modelos generados a partir de algoritmos de regresión de aprendizaje de máquinas.

Palabras clave

estimación de software; aprendizaje de máquinas, modelos de estimación, algoritmos de regresión, tamaño funcional.

Abstract

Software project effort estimation is the process of predicting the effort required to develop or maintain a software system. Developing estimation models and appropriate techniques is essential to avoid losses caused by a poor estimation, where more effort is spent than estimated.

The accuracy and reliability of the estimates play a very important role in project management, as they allow a feasible monitoring and control to ensure that the projects will be finished as planned.

This document presents a comparison between traditional estimation models based on statistical models and models generated from machine learning regression algorithms.

Keywords

software estimation, machine learning, estimation models, regression algorithms, functional size.

1. Introducción

La estimación de proyectos de software es una actividad importante en pequeñas y grandes organizaciones alrededor del mundo. Se han realizado investigaciones en la estimación de proyectos de software durante más de 20 años y una gran cantidad de modelos han sido propuestos. La cuestión es: ¿qué tan eficiente es actualmente la estimación de proyectos en la industria? la respuesta es: no muy eficiente [1].

A pesar de que en la industria se han desarrollado un amplio número de prácticas basadas en la estimación de proyectos, estas son poco favorables, es decir, tienen resultados poco confiables [2].

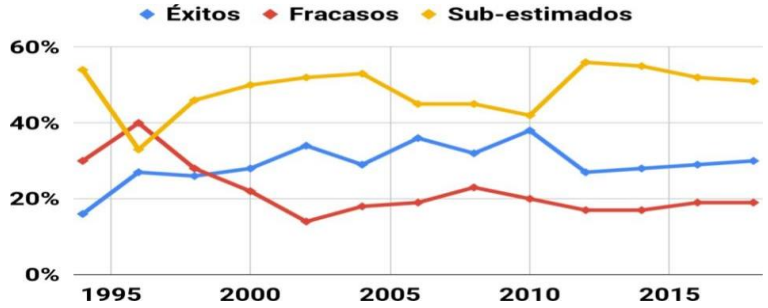

Figura 1: Tendencias de éxito de proyectos de software (Fuente: [1])

La Fig. 1 muestra las tendencias de éxito de proyectos de software del año 1994 al 2018. Los resultados se obtuvieron utilizando los datos del Standish Group Chaos Report [1]. Podemos observar que en la actualidad cerca del 50 % de los proyectos son subestimados, es decir, se termina invirtiendo más esfuerzo del estimado. Alrededor del 30% de los proyectos son exitosos, terminando el proyecto de acuerdo a lo planeado. Por último, aproximadamente el 20% son proyectos fracasados, donde estos fueron cancelados o no se utilizaron.

En este trabajo proponemos estimar el esfuerzo de proyectos de software aplicando algoritmos de aprendizaje de máquinas. Los algoritmos utilizados en este estudio incluyen: regresión lineal (LR), máquinas de regresión de vectores de soporte (SVR) y árboles de decisión de regresión (DTR). Además, el estudio busca comparar el desempeño de distintos modelos de estimación. Los resultados del estudio muestran que no siempre una regresión lineal simple, con la selección de un subconjunto de datos dada una característica del proyecto, genera el modelo de estimación con el mejor desempeño.

El resto del documento está organizado de la siguiente manera: la Sección 2 presenta información general relacionada con los modelos de estimación de software. La Sección 3 describe los algoritmos de aprendizaje de máquinas, haciendo énfasis en los algoritmos de regresión. La Sección 4 muestra el caso de estudio, aplicando aprendizaje de máquinas para generar modelos de estimación utilizando un subconjunto de proyectos de la base de datos internacional gestionada por el International Software Benchmarking Standards Group (ISBSG) [3], donde los proyectos seleccionados fueron medidos utilizando el método de Puntos de Función del International Function Point Users Group (IFPUG) [4]. Por último, la Sección 5 establece las conclusiones y el trabajo futuro.

2. Estimación de software

¿Qué es la estimación? PMBOK [5] lo define de la siguiente manera:

" Una evaluación cuantitativa de la cantidad o resultado probable. Normalmente se aplica a los costos, recursos, esfuerzo y duración del proyecto y suele estar precedida por un modificador".

Cuando se pide una estimación, a menudo se está pidiendo un compromiso o un plan para cumplir con un objetivo. Las distinciones entre las estimaciones, las metas y los compromisos son fundamentales para la comprensión de lo que es una estimación, lo que no es y cómo hacer mejores estimaciones. Una estimación es la predicción más optimista con una probabilidad distinta de cero de ser cierta, que tiene la misma probabilidad de estar por encima o por debajo del valor real [6].

¿Qué es una buena estimación? Expertos han propuesto diversas definiciones de una buena estimación del esfuerzo de proyectos de software. McConnell [6] ha declarado que una exactitud de 10% es posible, pero sólo en proyectos bien controlados, ya que proyectos poco controlados tienen demasiada variabilidad para lograr ese nivel de precisión. La mayor parte de los métodos de estimación tienen como objetivo predecir el costo y esfuerzo del proyecto. De acuerdo a Tuya [7] estos métodos pueden ser clasificados en cuatro categorías [8]:

1. Juicio de experto: basado en la experiencia de una persona.
2. Analogía: un enfoque más formal, donde los expertos comparan el proyecto a estimar con uno o más proyectos anteriores intentando encontrar similitudes y diferencias.
3. Descomposición: basado en un análisis de las características del proyecto, haciendo estimaciones individuales sobre los componentes del proyecto.

4. Modelos algorítmicos: basados en técnicas que identifican los factores clave que contribuyen al esfuerzo requerido para desarrollar el proyecto, generan un modelo matemático que relaciona dichos factores con el esfuerzo. Los modelos se basan normalmente en información obtenida de proyectos pasados.

Los modelos algorítmicos representan el enfoque más formal y son los que proporcionan resultados más confiables [7]. Estos modelos utilizan procedimientos como la regresión [9] para generar un modelo construido por una o varias expresiones matemáticas que relacionan el esfuerzo con una variable primaria, generalmente el tamaño funcional, y en algunos casos varios factores de ajuste secundarios.

El momento en el que se define el método de estimación también es importante, ya que algunos necesitan de datos históricos para poder ser implementados. Por ello, Abran [10] define otra clasificación de estimación: a priori y a posteriori.

- A priori: se tiene un ambiente de alta incertidumbre y un nivel de abstracción de necesidad muy alto. Usualmente se tiene una ventana de tiempo y recursos limitados, la mayoría de las características que se conocen son cualitativas. Los métodos utilizados generalmente son el juicio de experto.
    - Ventajas: no se necesitan datos históricos, manejo de incertidumbre a través de precisión de significado y la decisión sobre el juicio de experto puede ser muy valiosa debido a la información incompleta con la que se cuenta.
    - Desventajas: la experiencia le pertenece al experto y no a la organización, la experiencia no se puede replicar sistemáticamente.
- A posteriori: no hay incertidumbre y se conoce toda la información. Los métodos utilizados se basan en el análisis numérico y estadístico, como IFPUG [4], COSMIC [11], COCOMO [12], etc.
    - Ventajas: fácil de interpretar debido a que siguen un algoritmo.
    - Desventajas: requiere datos históricos, información oportuna, confiable y precisa.

Para la clasificación a posteriori, los modelos matemáticos típicamente son construidos utilizando datos de proyectos terminados. Por lo tanto, la mayor parte de los llamados “modelos de estimación” en la literatura, son en realidad modelos de productividad [10]. Algunos de los beneficios de los modelos de productividad basados en proyectos pasados son los siguientes:

- La eficiencia de estos modelos puede ser analizada y descrita. Cualquiera puede utilizar estos modelos para estimar futuros proyectos.

- Siempre que se ingresa la misma información al modelo, se tendrá el mismo resultado.

3. Algoritmos de aprendizaje de máquinas

El uso de algoritmos de aprendizaje de máquinas en la estimación de software recientemente ha ganado una gran popularidad, esto a causa del amplio margen de error en los métodos de estimación tradicionales [13].

En la actual industria de desarrollo software, donde hay avances tecnológicos frecuentes y nuevos lenguajes de programación disponibles, las mejoras metodológicas y los cambios en las habilidades de los equipos de proyectos demandan modelos de estimación que se adapten a entornos cambiantes a través del tiempo [14]. Los algoritmos de aprendizaje de máquinas tienen la capacidad de aprender y mejorar en función de las predicciones generadas, a través de la estimación más cercana a los datos observados y cualquier conocimiento previo [15].

Principalmente, el proceso de aprendizaje de los algoritmos puede tomar de una tarea descriptiva o predictiva elegida sin supervisión o supervisada. El aprendizaje sin supervisión se basa en patrones que no están vinculados a ninguna característica en el conjunto de datos de entrenamiento [16], para fines tales como agrupación, reglas de asociación o detección de anomalías. En cambio, el aprendizaje supervisado debe tener entradas definidas explícitamente, correspondientes a salidas conocidas por una característica objetivo-determinada [17] que se utiliza para la clasificación o la predicción. Para ambos, hay numerosos algoritmos disponibles que se pueden aplicar según el resultado deseado. Este estudio se centra en la predicción del esfuerzo del proyecto. Para ello se utilizaron tres algoritmos predictivos: LR, SVR y DTR.

3.1. Algoritmos de regresión

El objetivo de los algoritmos de regresión es crear una función *f* (*x*) que asigne adecuadamente un conjunto de variables independientes $x_1, x_2, \ldots, x_n$ a una variable dependiente *y*. En este caso, el objetivo es construir una serie de modelos de regresión utilizando la base de datos de la ISBSG y comparar el desempeño que tienen para estimar el esfuerzo de desarrollo de proyectos de software de manera precisa.

La regresión lineal es uno de los casos más comunes. Es fundamental para la práctica de la estadística y es parte del conocimiento básico de cualquier estadística aplicada [18]. En los modelos de estimación de software tradicionales, el modelo viene dado por una recta, donde la variable x representa el tamaño funcional, la variable y *representa* el esfuerzo, α *representa* la ordenada al origen que podríamos tomar como un costo fijo y *β representa* la pendiente de la recta que indica la cantidad que crecerá (o decrecerá) la variable respuesta en términos de un incremento de la variable independiente [2].

Cuando un modelo de productividad no puede modelar de la mejor manera todas las circunstancias posibles de datos de entrada, puede ser necesario identificar más de un modelo para el mismo conjunto de datos. Los subconjuntos de datos para generar los diferentes modelos de estimación pueden tener algunas características en común, como el tipo de desarrollo, arquitectura, lenguaje de programación, sistema operativo, etc. [2].

En este caso, se debe analizar la correlación de las características con el esfuerzo del proyecto, con el fin de determinar qué subconjuntos de datos generarán los modelos más precisos, ya que un error común es hacer una partición de los datos de acuerdo a las características de los proyectos que deseamos estimar, sin validar los modelos de estimación generados.

Del mismo modo, una regresión lineal simple, tomando solo la característica del tamaño funcional, no siempre genera el modelo con la mayor precisión, ya que, dependiendo de los datos de entrada, una regresión lineal múltiple u otros algoritmos de regresión pueden ajustarse de mejor manera y tener predicciones más precisas, tal y como se muestra en la sección 4 de este documento.

Además de la regresión lineal, los otros dos algoritmos que utilizaremos son máquinas de regresión de vectores de soporte [19] y árboles de decisión de regresión [20]. El primero tiene la capacidad de modelar problemas complejos lineales y no lineales y generar resultados altamente precisos, incluso en datos ruidosos. Sin embargo, su proceso de entrenamiento puede llevar mucho tiempo en un gran volumen de datos.

En el caso de los árboles de decisión de regresión, son un tipo especial de árbol de decisión desarrollado para tareas de regresión. Esta técnica construye un modelo en forma de árbol, dividiendo recursivamente el conjunto de datos hasta que un criterio de paro se satisface. Esta división se realiza con el propósito de alcanzar la máxima homogeneidad posible relativa a la variable dependiente. Todos los nodos en el árbol, menos los terminales (también llamados hojas), especifican una condición basada en una de las variables que tienen influencia en la variable dependiente. Luego de que el árbol es generado, se puede utilizar para realizar predicciones siguiendo un camino a través de él de acuerdo con los valores específicos de las variables de entrada [21]. En los árboles de regresión, la elección de cada nodo está usualmente guiada por el criterio del error cuadrático medio mínimo [22].

4. Estimación de software con algoritmos de aprendizaje de máquinas

4.1. Preprocesamiento de los datos

La aplicación de los algoritmos de aprendizaje de máquinas requiere que los datos estén en un formato matemáticamente factible a través de un preprocesamiento. Estas técnicas consisten en la reducción de datos, la proyección de datos y el tratamiento de datos faltantes. La reducción de datos tiene como objetivo reducir el tamaño de las entradas de datos mediante la selección de características. La proyección de datos tiene la intención de transformar los datos, por ejemplo, escalando todas las características en un mismo rango predefinido. El tratamiento de datos faltantes incluye eliminar valores perdidos y/o reemplazarlos [23]. Para este estudio aplicaremos las tres técnicas.

La base de datos a utilizar contiene un total de 951 registros con las siguientes características:

1. FunctionalSize (*FS*) [numeric]
2. ValueAdjustmentFactor (*VAF*) [numeric]
3. ProjectElapsedTime (*PET*) [numeric]
4. DevelopmentType (*DT*) [Enhancement, New Development]
5. BusinessAreaType (*BAT*) [Banking, Telecommunications]
6. ClientServer (*CS*) [Yes, No]
7. DevelopmentPlatform (*DP*) [MF, PC, Multi, MR]
8. LanguageType (*LT*) [3GL, 4GL]
9. FirstOS (*FOS*) [Unix, Mainframe, Windows]
10. MaxTeamSize (*MTS*) [numeric]
11. NormalisedWorkEffortLevel1 (*NWE1*) [numeric]

Es importante mencionar que deseamos predecir el esfuerzo de desarrollo

del proyecto, por lo que la variable objetivo será la característica *NWE1*.

La Tabla 1 muestra las características categóricas de los primeros 10 registros de la base de datos. Primero convertiremos las características categóricas de representación textual a numérica. Realizaremos la conversión de los datos de la siguiente manera:

- *DT* [Enhancement=0, New Development=1]
- *BAT* [Banking=0, Telecommunications=1]
- *CS* [Yes=1, No=0]
- *DP* [MF=0, PC=1, Multi=2, MR=3]
- *LT* [3GL=0, 4GL=1]
- *FOS* [Unix=0, Mainframe=1, Windows=2]

DT	BAT	CS	DP	LT	FOS
Enhancement	?	Yes	?	?	?
New Development	?	?	MF	3GL	Unix
Enhancement	?	?	MF	3GL	Unix
Enhancement	?	?	?	3GL	?
Enhancement	?	?	?	?	?
Enhancement	?	Yes	?	4GL	?
Enhancement	?	No	MF	3GL	Mainframe
Enhancement	?	?	MF	3GL	Mainframe
New Development	?	?	PC	4GL	Windows
New Development	?	Yes	Two	Two	?

TABLA 1: Características categóricas de los primeros 10 registros

FS	VAF	PET	DT	BAT	CS	DP	LT	FOS	MTS	NWE1
3	1.09	0	0	?	1	?	?	?	?	28
620	?	7	1	?	?	0	0	0	?	18160
730	1.14	?	0	?	?	0	0	0	?	20975
114	1.18	?	0	?	?	?	0	?	?	7290
460	1	4	0	?	?	?	?	?	5	2253
7	1.1	1.87	0	?	1	?	1	?	?	396
80	1.12	13.7	0	?	0	0	0	1	3.5	3199
4	?	0.1	0	?	?	0	0	1	?	145
309	?	3	1	?	?	1	1	2	?	2516
69	0.95	9	1	?	1	?	0	?	?	2915

TABLA 2: Primeros 10 registros después de la conversión de las características categóricas

La Tabla 2 muestra los primeros 10 registros después del

preprocesamiento de datos. Lo siguiente es el tratamiento de valores faltantes, para ello obtendremos las siguientes estadísticas para cada característica: valor mínimo (min), máximo (max), media (median), mediana (median), moda (mode) y porcentaje de valores faltantes (m v (%)).

En la Tabla 3 podemos observar que las características *BT* y *CS* tienen más del 70 % de valores faltantes, por lo que serán descartadas ya que no proporcionan suficiente información para construir modelos adecuados. Por otra parte, las características *DP*, *FOS* y *MTS* tienen entre el 50% y 70% de valores faltantes. Para DP y FOS asignaremos una categoría única a los valores faltantes (4 y 3 respectivamente), ya que son características categóricas y para MTS (continua) utilizaremos el promedio (7,5). Por último, *VAF, PET* y *LT* cuentan con menos de la tercera parte de valores faltantes. Para *VAF* y *PET* (continuas) utilizaremos los promedios (1,04 y 6,04 respectivamente) y para LT (categórica) utilizaremos la moda (0). Después de todo el preprocesamiento, los datos quedan de acuerdo a la Tabla 3.

caract	min	max	mean	median	mode	m v (%)
FS	3.0	4911.0	301.95	135.0	51.0	0.0
VAF	0.65	1.3	1.03	1.0	1.0	0.33
PET	0.0	44.0	6.00	5.0	3.0	0.30
DT	0.0	1.0	0.22	0.0	0.0	0.0
BAT	0.0	1.0	0.81	1.0	1.0	0.77
CS	0.0	1.0	0.79	1.0	1.0	0.71
DP	0.0	3.0	0.95	0.0	0.0	0.50
LT	0.0	1.0	0.35	0.0	0.0	0.25
FOS	0.0	2.0	0.82	1.0	1.0	0.68
MTS	0.5	80.0	7.69	5.0	4.0	0.57
NWE1	4.0	73920.0	3327.77	1520.0	480.0	0.0

TABLA 3: Estadísticas de las características

FS	VAF	PET	DT	DP	LT	FOS	MTS	NWE1
3	1.09	0	0	4	0	3	7.5	28

620	1.04	7	1	0	0	0	7.5	18160
730	1.14	6.04	0	0	0	0	7.5	20975
114	1.18	6.04	0	4	0	3	7.5	7290
460	1	4	0	4	0	3	5	2253
7	1.1	1.87	0	4	1	3	7.5	396
80	1.12	13.7	0	0	0	1	3.5	3199
4	1.04	0.1	0	0	0	1	7.5	145
309	1.04	3	1	1	1	2	7.5	2516
69	0.95	9	1	4	0	3	7.5	2915

TABLA 4: Primeros 10 registros después de todo el preprocesamiento de los datos

4.2. Correlación entre las características

Una vez realizado el preprocesamiento de los datos, creamos una matriz de correlación que mide las relaciones lineales entre las características. El coeficiente de correlación varía de -1 a 1. Si el valor es cercano a 1, significa que existe una fuerte correlación positiva entre las dos características. Cuando está cerca de 1, estas tienen una fuerte correlación negativa.

En la Fig. 2 podemos observar que las 3 características con mayor correlación a *NWE1* son: *FS* (0,69), *PET* (0,58) y *MTS* (0,5). Sin embargo, es común en las estimaciones de proyectos de software que la duración del proyecto se determine a partir del esfuerzo, por lo que es una característica que no conocemos de inicio cuando queremos estimar un nuevo proyecto. Por lo tanto, *PET* no será considerada en la generación de los modelos de predicción. Por otra parte, podemos observar 2 características con correlación negativa (*LT* y *FOS*), prácticamente con una correlación igual a cero. Sin embargo, esto no significa que quedarán fuera de los modelos de predicción que generaremos.

La Fig. 3 muestra el diagrama de dispersión de la característica con mayor correlación (*FS*) respecto a NWE*1*. Podemos observar que existe una fuerte correlación positiva, a mayor tamaño funcional mayor esfuerzo.

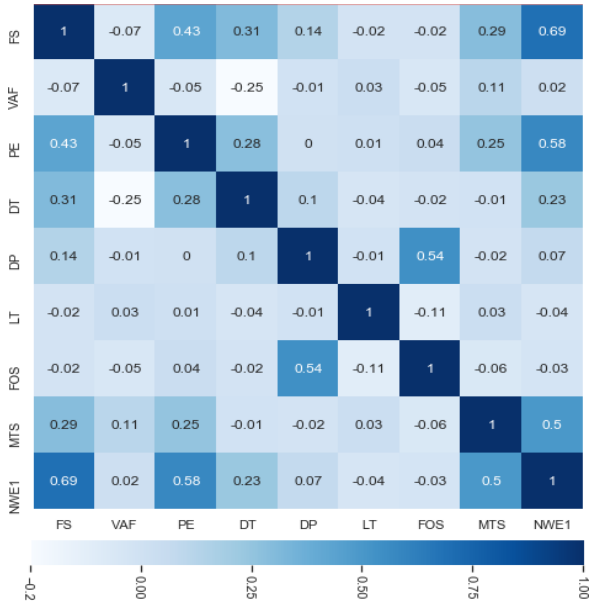

Figura 2: Matriz de correlación de las características

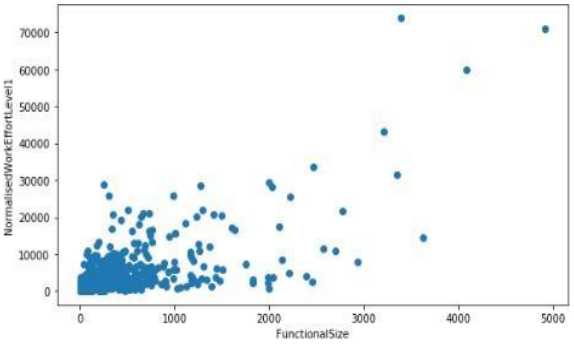

Figura 3: Diagrama de dispersión de FunctionalSize con NormalisedWorkEffortLevel1

Para un mayor panorama del desempeño de los modelos, las características que formarán parte de los modelos utilizarán las siguientes selecciones:

- todas las características
- características con correlación positiva
- características con mayor correlación a *NWE1*
- únicamente la característica *FunctionalSize*

Esta última en particular es muy importante, ya que uno de los objetivos de este documento es comparar los modelos de LR generados únicamente con *FunctionalSize* contra modelos generados con algoritmos de aprendizaje de máquinas. Por otra parte, haremos una partición de los datos dada una característica del proyecto y analizaremos los modelos de estimación generados con la partición de los datos.

4.3. Criterios de calidad para modelos de estimación

Los criterios de evaluación de errores son esenciales para validar la precisión de los modelos.

Para evaluar la calidad de estos utilizaremos las siguientes métricas:

1. **Coeficiente de determinación (**R^2**)** [9]: Es la proporción de la suma total de los cuadrados de la variable dependiente por las variables independientes en el modelo.

$$R^2 = \frac{SS_R}{SS_T}$$

donde SS_T *es* el total de la suma de los cuadrados y SS_R es la regresión de la suma de los cuadrados. Es decir:

$$\frac{\left(n\sum xy - (\sum x)(\sum y)\right)^2}{(n(\sum x^2) - (\sum x)^2)(n(\sum y^2) - (\sum y)^2)}$$

Una R^2 cerca de 1 indica que existe una fuerte relación entre la variable independiente y la variable dependiente. Un valor de R^2 cercano a 0 estaría hablando de un modelo poco adecuado, pues la variabilidad del error sería muy grande

2. **Mediana del Error Absoluto (Median Absolute Error - MedAE)** [25]: Es particularmente interesante porque es robusto a los valores atípicos. La pérdida se calcula tomando la mediana de todas las diferencias absolutas entre el objetivo y la predicción. Si y_i' *es* la predicción de la muestra e y_i es el valor verdadero correspondiente, entonces la mediana del error absoluto estimado sobre *n* muestras se define de la siguiente manera:

$$MedAE = median(|y_1 - y_1'|, \dots, |y_n - y_n'|)$$

3. **Magnitud Media del Error Relativo (Mean Magnitude of Relative Error - MMRE)** [9]: Indica la media de la Magnitud del Error Relativo (Magnitude of Relative Error - MRE). El MRE corresponde al valor absoluto de la diferencia entre el esfuerzo conocido del proyecto *y* y la predicción de la muestra y^i dividido por el esfuerzo conocido:

$$MMRE = \frac{1}{n}\sum_{i=1}^{n}\frac{|y_i - y_i'|}{y_i}$$

4. **Nivel de predicción del modelo (PRED)** [9]: Indica la proporción de proyectos que son menores o iguales a un porcentaje establecido (comúnmente 25 %) respecto al MRE:

$$PRED(l) = \frac{k}{n}$$

donde *k* es el número de proyectos en el conjunto de datos de tamaño *n* para los cuales su

MRE ≤ *l*.

Por lo tanto, los resultados esperados idealmente de un modelo de estimación adecuado son: una R^2 y PRED cerca de 1 y un MedAE y MMRE cerca de 0.

4.4. Generación y validación de modelos

Una vez que hemos preprocesado los datos, seleccionado las características que se utilizarán para la generación de los modelos y definido los criterios de calidad para evaluarlos, lo siguiente es definir, entrenar y validar los modelos de predicción. Para ello utilizamos el lenguaje de programación Python y la biblioteca de sklearn [25]. Dentro de esta biblioteca podemos elegir distintos parámetros de los algoritmos, algunos principales son el kernel para el caso de SVR y el grado para DTR.

Las Fig. 4, 5 y 6 muestran el desempeño de los algoritmos utilizando distintos tamaños del conjunto de datos de entrenamiento. Podemos observar que las primeras dos gráficas (LR y SVR) presentan escenarios donde el desempeño aumenta muy poco a mayor cantidad de datos de entrenamiento, mientras que la tercera gráfica (DTR) aumenta el desempeño a mayor número de datos, pero muestra un escenario de alta varianza.

Para una primera aproximación, se aplicó una validación cruzada de 5 iteraciones a los 3 algoritmos utilizando todas las características. Este procedimiento verifica la precisión (R^2) de los modelos, obteniendo la media y desviación estándar de las predicciones obtenidas en las iteraciones. Además, asegura que los modelos no estén sobreajustados y permite su evaluación adecuada [24].

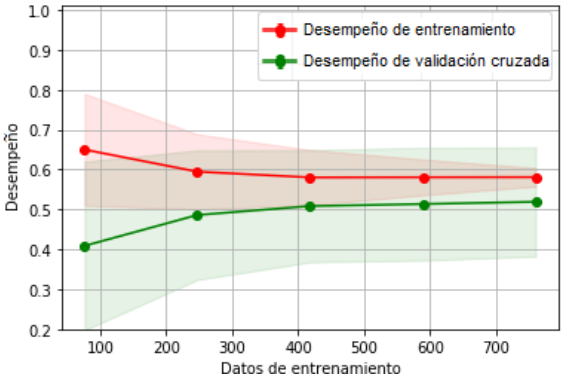

Figura 4: Gráfica de desempeño con distintos tamaños del conjunto de datos de entrenamiento con regresión lineal

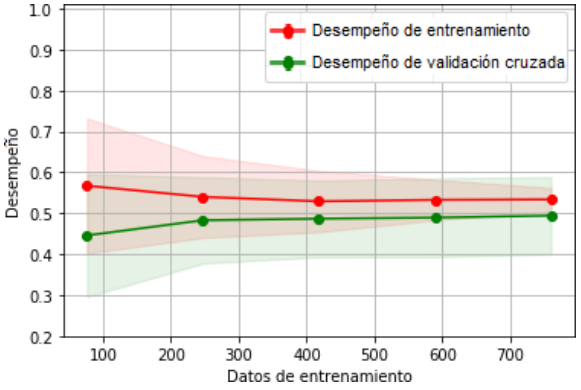

Figura 5: Gráfica de desempeño con distintos tamaños del conjunto de datos de entrenamiento con máquinas de vectores de soporte

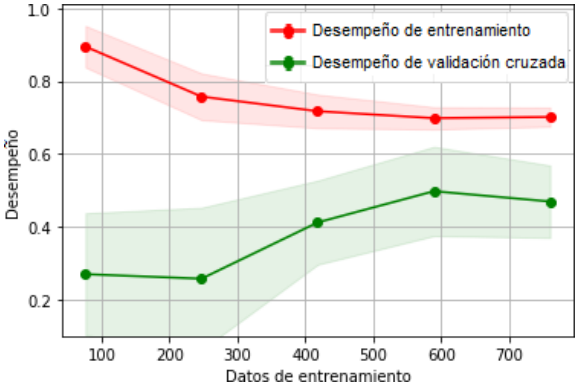

Figura 6: Gráfica de desempeño con distintos tamaños del conjunto de datos de entrenamiento con árboles de decisión

La Tabla 5 muestra la validación cruzada tomando el 70% de los datos para entrenamiento de los modelos y el 30 % para validación.

Una vez definidos los algoritmos, procedemos a entrenar y validar los modelos, utilizando el 70 % de todos los datos para entrenamiento y el 30% restante para evaluarlos con los criterios de calidad.

En la Tabla 6 podemos observar que no hay un algoritmo ni una selección de las características en las que se obtengan siempre los mejores resultados. LR tuvo mejores valores en la mayoría de los casos para la $R^{2,}$ pero los peores en MMRE. Para SVR siempre tuvo los mejores resultados para MedAE y MMRE, pero no para los demás criterios. Por último, DRT solo presentó un PRED más alto la mayoría de las veces.

Algoritmo	**Precisión**		**Iteraciones**				
LR	0.59	± 0,10	0.62	0.59	0.67	0.51	0.59
SVR	0.55	± 0,02	0.57	0.54	0.56	0.54	0.55
DTR	0.43	± 0,23	0.41	0.50	0.56	0.46	0.22

TABLA 5: Validación cruzada de 5 iteraciones de los 3 algoritmos

Selección	**Algoritmo**	**R^2**	**MedAE**	**MMRE**	**PRED**
Todas las características	LR	**0.62**	1110.81	3.2	0.19
	SVR	0.58	**825.07**	**2.13**	0.2
	DTR	0.39	901.57	2.56	**0.21**
Características con correlación positiva	LR	**0.56**	1212.49	3.48	0.18
	SVR	0.56	**851.67**	**2.36**	**0.22**
	DTR	0.53	1286.87	2.74	0.2
Características con mayor correlación	LR	0.55	1214.19	3.54	0.17
	SVR	0.54	**845.45**	**2.34**	0.21
	DTR	**0.58**	1247.5	2.63	**0.21**
Solo con Functional Size	LR	0.37	1153.28	2.96	0.14
	SVR	**0.38**	**858.3**	**1.79**	**0.22**
	DTR	0.21	1330.32	2.88	0.18

TABLA 6: Evaluación de criterios de calidad utilizando todas las características

Es importante resaltar que entrenar los algoritmos únicamente con la característica FS solo obtuvo mejores resultados en MMRE que las demás selecciones.

Lo siguiente es entrenar y comparar los algoritmos haciendo una partición de los datos dada una característica del proyecto, como lo hacen los modelos de estimación tradicionales. Se hace una partición de los datos para generar diferentes modelos de predicción de acuerdo con una característica en común, como el tipo de desarrollo [2]. En la base de datos tenemos 2 tipos distintos, Enhancement y New Development.

La Fig. 7 muestra el diagrama de dispersión de *FS* con *NWE1*, resaltando en distintos colores el tipo de desarrollo (Enhancement en rojo y New Development en gris). En principio parece no haber gran diferencia en el comportamiento de los datos dada la característica.

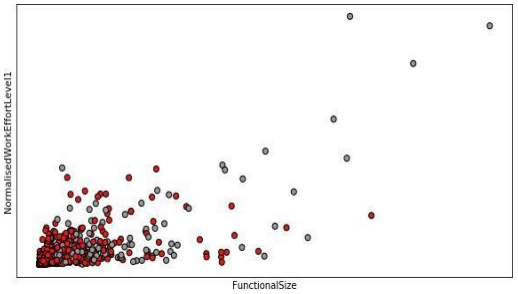

Figura 7: Diagrama de dispersión de FS con NWE1 resaltando el tipo de desarrollo (Enhancement rojo y New Development gris)

Las Tablas 7 y 8 muestran el resultado de entrenar nuevamente los algoritmos, en esta ocasión con la partición de los datos por tipo de desarrollo en Enhancement y New Development, con 733 y 218 registros respectivamente.

En la Tabla 7 podemos observar que la mayoría de las veces la R^2 y el MMRE tuvo peores

resultados para LR y SVR, mientras que para DTR estos valores mejoran con la partición de los datos. Dada la partición, entrenar y validar los modelos utilizando todas las características obtuvo mejores resultados en R^2 y el MMRE. En esta ocasión, entrenar los algoritmos únicamente con *FS* obtuvo mejores resultados, dando mejores valores para MedAE y PRED con SVR.

Selección	Algoritmo	R^2	MedAE	MMRE	PRED
Todas las características	LR	0.29	1301.11	4.84	0.19
	SVR	0.31	**712.87**	2.43	0.24
	DTR	**0.55**	778.41	**2.01**	**0.26**
Características con correlación positiva	LR	0.32	1253.51	5.03	0.19
	SVR	0.3	**689.44**	2.47	0.24
	DTR	**0.54**	778.41	**2.01**	**0.25**
Características con mayor correlación	LR	0.32	1186.08	4.92	0.21
	SVR	0.29	**700.87**	2.45	0.24
	DTR	**0.52**	846	**2.03**	**0.25**
Solo con Functional Size	LR	0.2	1189.95	4.64	0.15
	SVR	**0.2**	**589.92**	**2.03**	**0.28**
	DTR	0.17	947.86	2.38	0.16

TABLA 7: Evaluación de criterios de calidad con tipo de desarrollo Enhancement

Selección	Algoritmo	R^2	MedAE	MMRE	PRED
Todas las características	LR	**0.71**	1824.19	2.64	**0.23**
	SVR	0.54	**1714.33**	**2.24**	0.2
	DTR	0.58	2707.99	2.89	0.12
Características con correlación positiva	LR	**0.73**	1898.3	2.63	**0.26**
	SVR	0.54	**1714.6**	**2.24**	0.23
	DTR	0.41	2707.99	2.89	0.14
Características con mayor correlación	LR	**0.74**	**1625.02**	2.28	**0.24**
	SVR	0.54	1691.3	**2.22**	0.21
	DTR	0.41	2707.99	2.9	0.15
Solo con Functional Size	LR	**0.67**	1720.26	2.17	0.17
	SVR	0.48	**1159.71**	**1.62**	**0.23**
	DTR	0.35	2863.26	2.8	0.15

TABLA 8: Evaluación de criterios de calidad con tipo de desarrollo New Development

Algoritmo	R^2	MedAE	MMRE	PRED
LR	**6**	1	0	3
SVR	2	**11**	**9**	4
DTR	4	0	3	**5**

TABLA 9: Número de ocasiones en que cada algoritmo presentó los mejores resultados

La partición de los datos por New Development (Tabla 8) resultó muy interesante, ya que presentó los mejores valores para R^2, MMRE y PRED, mientras que el mejor valor de MedAE fue con la partición de Enhancement. A diferencia de este, entrenar los algoritmos con todas las características presentó los resultados más bajos que las demás selecciones. Sin embargo, *FS* nuevamente presentó el mejor MedAE y en esta ocasión el mejor MMRE. Respecto a los algoritmos, DTR siempre presentó los peores resultados, LR fue mejor en la R^2 y en la mayoría de PRED, mientras que SVR presentó el mejor MMRE y en la mayoría de los casos el mejor MedAE.

Por último, la Tabla 9 muestra el número de veces que cada algoritmo presentó los mejores resultados para los criterios de calidad de los modelos validados. Podemos observar que SVR predominó en los criterios de MedAE y MMRE obteniendo casi siempre los mejores resultados. Por otra parte, LR obtuvo la mayoría de las veces mejores resultados en la R^2. Para el caso de PRED, se comportó de manera más uniforme respecto a la predicción de los algoritmos, ya que DTR obtuvo los mejores resultados en este criterio, pero solo una ocasión más que LR y dos más que SVR.

5. Conclusión

En los últimos años, investigadores y profesionistas han propuesto varias metodologías para la estimación del esfuerzo de proyectos de software. En este documento hemos propuesto el uso de algoritmos de aprendizaje de máquinas para generar modelos de estimación, pero ¿estos son mejores que los modelos de estimación de software tradicionales?, la respuesta es: no en todos los casos. De acuerdo con los resultados obtenidos, los modelos de estimación basados en algoritmos de aprendizaje de máquinas obtuvieron en la mayoría de las veces mejores resultados para los criterios de calidad. Los resultados de las comparaciones entre los modelos dieron resultados muy variados. En algunos casos, entrenar los algoritmos con todas las características de los proyectos dio mejores criterios de calidad, donde una ventaja de los algoritmos de aprendizaje de máquinas es que no se limitan a utilizar solo el tamaño funcional para el entrenamiento y validación de los modelos. Sin embargo, en otras ocasiones fue más conveniente utilizar solo el tamaño funcional como lo hacen los modelos de estimación tradicionales.

Los resultados que obtuvimos haciendo una partición de los datos dada una característica del proyecto, en este caso por tipo de desarrollo, fueron convenientes solo para New Development, ya que presentó mejores criterios

de calidad que los modelos sin la partición. Sin embargo, Enhancement tuvo resultados menos precisos. Por lo tanto, es necesario validar la calidad de todos los modelos que se generen haciendo una partición de los datos para asegurar que podemos generar estimaciones confiables a partir de ellos.

5.1. Trabajo futuro

En la actual industria de desarrollo de software, donde surgen nuevas metodologías, herramientas y lenguajes de programación, contar con modelos de estimación confiables es de suma importancia. Sería de gran apoyo contar con un sistema que haga el análisis y preprocesamiento de los datos y genere los modelos de estimación más precisos posibles para los proyectos que deseamos estimar, ya que además de darnos confianza en las estimaciones, nos ahorraría tiempo y recursos. Desarrollar un sistema con estas características es parte de un trabajo de tesis de maestría, el cual utiliza principalmente algoritmos de aprendizaje de máquinas, como los descritos en este documento.

REFERENCIAS

[1]" The Standish Group", Standishgroup.com,2019. [Online]. Available: http://standishgroup.com/. [Accessed: 21- May- 2019].

[2] A. Abran, Software Project Estimation: The Fundamentals for Providing High Quality Information to Decision Makers, Illustrated ed., Hoboken, New Jersey: John Wiley & Sons, Inc., 2015, p. 261.

[3] International Software Benchmarking Standards Group, ISBSG Repository R1 - Field Descriptions, Software project data, 2017.

[4] International Function Point Users Group, Function Point Counting Practices Manual, Version 4.3.1, Measurement Manual, Jan 2010.

[5] Project Management Institute, Inc., A Guide to the Project Management Body of Knowledge (PMBOK), 5 ed., Newtown Square, Pennsylvania USA: PMI Publishing Division, 2000, p. 596.

[6] S. McConnell, Software Estimation: Demystifying the Black Art, Illustrated ed., the University of California: Microsoft Press, 19 Nov 2009, p. 308.

[7] J. Tuya, I. R. Román y J. J. D. Cosín, Técnicas cuantitativas para la gestión en la ingeniería del software, Netbiblo, 2007, p. 373.

[8] A. Idri, A. Abran, and T. M. Khoshgoftaar, "Estimating Software Project Effort by Analogy Based on Linguistic Values," in Proceedings - International Software Metrics Symposium, Jan 2002, p. 21-30.

[9] J. O. Rawlings, S. G. Pantula y D. A. Dickely, Applied Regression Analysis: A Research Tool, Second ed., Department of Statistics, North Carolina State University: Springer Science & Business Media, 2001, p. 660.

[10] A. Abran, Software Benchmarking, Estimation and Quality Models Based on Functional Size with COSMIC – ISO 19761, Draft Apr 2008.

[11] COSMIC Measurement Practice Commitee, The COSMIC Functional Size Measurement Method, Version 4.0.2, Measurement Manual, Dec 2017.

[12] B. W. Boehm, Clark, Horowitz, Brown, Reifer, Chulani, R. Madachy, and B. Steece, Software Cost Estimation with Cocomo II with Cdrom, 1st ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2000.

[13] K. Srinivasan and D. Fisher," Machine learning approaches to estimating software development effort, ¨IEEE Transactions on Software Engineering, vol. 21, 1995, p. 126-137.

[14] P. Pospieszny et al., An effective approach for software project effort and duration estimation with machine learning algorithms, The Journal of Systems and Software, 137 (2018), p. 184-196,

[15] T. Mitchell, Machine learning. New York: McGraw Hill, 2017.

[16] Linoff, G.S., Berry, M.J.A., Data Mining Techniques: For Marketing, Sales, and Customer Relationship Management. John Wiley Sons, 2011.

[17]Cios, K., Pedrycz, W., Swiniarski, R., Kurgan, L., Data Mining A Knowledge Dis- covery Apporach. Springer Science, New York, New York, USA, 2007.

[18] J. J. Faraway, Extending the Linear Model with R: Generalized Linear, Mixed Effects and Nonparametric Regression Models, Boca Raton: CRC Press, 2006, p. 345.

[19] Han, J., Kamber, M., Pei, J., Data Mining: Concepts and Techniques, Morgan Kaufmann, 2006.

[20] I. H. Witten, E. Frank, Data Mining: Practical Machine Learning Tools and Techniques, 2nd ed., Morgan Kaufmann, San Francisco, 2005.

[21] IDRI, A. & ELYASSAMI, S. Applying Fuzzy ID3 Decision Tree for Software Effort Estimation.
International Journal of Computer Science Issues, 2011.

[22] BRAGA, P. L., OLIVEIRA, A. L. I., RIBEIRO, G. H. T. & MEIRA, S. R. L. Bagging
Predictors for Estimation of Software Project Effort. In: International Joint Conference on Neural Networks, Orlando, Florida, 2007.

[23] N. Mittas, L. Angelis, LSEbA: least squares regression and estimation by analogy in a semi- parametric model for software cost estimation, Empirical Softw. Eng. 15, 2010.

[24] Krogh, Anders Jesper, V., Neural Network Ensembles, Cross Validation, and Active Learning.
In: Advances in Neural Information Processing Systems, 7, 1995, pp. 231-238.

[25]" scikit-learn: machine learning in Python — scikit-learn 0.21.1 documentation", Scikit- learn.org, 2019. [Online]. Available: https://scikit-

learn.org/stable/index.html. [Accessed: 21- May- 2019].

Notas biográficas

Jesús Iván Saavedra Martínez es estudiante de la Maestría en Ciencia e Ingeniería de la Computación en la UNAM y también es Ayudante de Profesor de Asignatura desde el 2016 en la Facultad de Ciencias de la misma universidad. Ha impartido cursos de Ingeniería de Software principalmente en el área de Medición y Estimación de Software y cursos con temas relacionados a la Inteligencia Artificial.

María Guadalupe Elena Ibargüengoitia González es profesora de la UNAM tanto en la Facultad de Ciencias en la carrera de Ciencias de la Computación, como en el Posgrado en Ciencia e Ingeniería de la Computación. Ha impartido cursos de Ingeniería de Software desde 1982 a nivel licenciatura y desde 1993 en el posgrado. Actualmente imparte cursos de Ingeniería de Software a nivel licenciatura y maestría en universidades nacionales e internacionales. Ha asesorado a empresas y organizaciones en el desarrollo de software.

Gibran Fuentes Pineda es investigador y desarrollador en minería de datos, aprendizaje de máquinas y visión por computadora con un enfoque en large-scale problems y robótica. Es profesor e investigador asociado del Instituto de Investigaciones en Matemáticas Aplicadas y Sistemas. Actualmente imparte cursos de aprendizaje de máquinas y aprendizaje profundo.