Software Development Effort Estimation based-on Multiple Classifier system and Lines of Code

H. Velarde, C. Santiesteban, A. García and J. Casillas

***Abstract*— The development effort estimation is one of the most difficult problems in software project management. It is one of the most critical aspects in the early stages of the software project. Several software development effort estimation models have been proposed, however, these models are not able to obtain more than a 25 percent of accuracy, neither provide an understandable model for experts in the application area. Therefore, in this paper, we present EEpred, an explanatory model to estimate the development effort based on data of known software projects. It is a serial multiple classifier system based-on several decision trees. The model performance was evaluated by an internal validation procedure, analyzing their robustness and predictive performance. This procedure demonstrates that EEpred is able to estimate the software development effort with a 71 percent of precision. The main advantage of EEpred, regarding to other algorithms, is its ability to translate the process into a collection of simple decision rules, providing more easily interpretable knowledge that can help software engineer to improve decision-making on development planning.**

***Keywords*— Explanatory models, Effort estimation, Multiple Classifiers, Decision Trees, Regression Trees, Software metrics, Software projects.**

I. INTRODUCCIÓN

LA ESTIMACIÓN del esfuerzo de desarrollo de software (EEDS) es una de las primeras y más importantes actividades de un proyecto. Es una tarea complicada debido a la gran cantidad de factores (atributos del proceso o del producto) que influyen en el valor de la variable objeto de la predicción (tamaño, esfuerzo, coste del proyecto, características de calidad del software). Para poder realizar predicciones sobre características del proyecto actual es necesario disponer de información obtenida en proyectos concluidos. Lo cual permitiría establecer un modelo de relaciones entre dichas variables que constituya la base del proceso de estimación [1].

Hasta la actualidad han sido desarrollados múltiples modelos de predicción de EEDS. Los modelos de predicción basados en técnicas de aprendizaje automático han demostrado ser superiores a los modelos tradicionales [2]. Dentro de los métodos de aprendizaje más empleados se encuentran las redes neuronales, algoritmos genéticos, árboles de decisión y regresión, algoritmos de agrupamiento, combinación de clasificadores, entre otras técnicas de minería de datos [1]–[9]. Sin embargo, a pesar de estos esfuerzos aún no se logran modelos capaces de predecir el esfuerzo con adecuada precisión, ni capaces de explicar su base de conocimiento [1], [10], [11].

En el presente artículo se presenta EEpred, un modelo nuevo para la EEDS, con capacidad explicativa. Este modelo se basa en la combinación de clasificadores en serie. Donde, el primer nivel realiza una predicción granular del tipo del proyecto y el segundo nivel, en base al resultado de la anterior, realiza una predicción fina capaz de estimar la EEDS.

El procedimiento de evaluación incluyó: (1) medición apropiada de la robustez y capacidad de generalización del algoritmo, mediante un proceso de validación interna; (2) el análisis del error; (3) el mecanismo de interpretación.

El artículo está estructurado como sigue: la sección de materiales y métodos donde se muestran los datos empleados, la arquitectura del modelo propuesto y el método de evaluación del desempeño del método. La sección de análisis y discusión de los resultados, donde se mide la calidad del modelo. Finalmente, en las conclusiones del trabajo se resumen los resultados obtenidos y su significación.

II. MATERIALES Y MÉTODOS

EEpred es un predictor basado en la combinación de dos niveles de clasificadores. Donde el primer nivel realiza una predicción granular, mientras que el segundo afina ésta, en función de su base de conocimientos.

A. Bases de datos

Para realizar la experimentación del modelo propuesto fueron seleccionadas cuatro bases de datos públicas, extraídas del repositorio PROMISE (*PRedictOr Models In Software Engineering Software*) [12]. Como se muestra en la Tabla I, este conjunto de datos recoge un amplio número de proyectos y atributos, los cuales representan un elevado número de casos útiles para la EEDS.

TABLA I
CARACTERIZACIÓN DEL SET DE DATOS EMPLEADOS EN LA EXPERIMENTACIÓN

Base de datos	Dimensiones		Esfuerzo			
	Proyectos	Atributos	Min	Max	Med	Dev-est
Cocomo81	63	17	5,9	11400	683,3	1821,6
Nasa_1	60	16	8,4	3240	406,4	657
Nasa_2	93	24	8,4	8211	624,4	1135,9
Cocomo_Sdr	12	25	1	22	5,7	6,8
Total a usar	**228**	**11**	**1**	**11400**	**550,8**	**1252,9**

Para entrenar el modelo propuesto, fueron escogidos 11 atributos comunes a todas las bases de datos. Éstos describen características de los proyectos en cuanto a: tipo de software, personal que participará en el proyecto y datos propios de la

H. Velarde, Facultad de Ciencias e Ingenierías. Físicas y Formales, Universidad Católica de Santa María, Arequipa, Perú, hvelardeb@gmail.com.

C. Santiesteban, Centro de Bioplantas, Universidad de Ciego de Ávila, Cuba, cosme@bioplantas.cu.

A. García, División Territorial Villa Clara, DESOFT, Cuba, anamaria.garcia@vcl.desoft.cu.

J. Casillas, Departamento de Ciencias de la Computación e Inteligencia Artificial, Universidad de Granada, España, casillas@decsai.ugr.es.

organización (Tabla II). El valor de los atributos empleados es nominal.

TABLA II
ATRIBUTOS ESCOGIDOS PARA CREAR EL MODELO

N	Atributo	Objetivo	Descripción
1	**Rely**	Software	Confiabilidad del software.
2	**Data**		Dimensión de la base de datos.
3	**Cplx**		Complejidad del software.
4	**Acap**	Equipo de desarrollo	Calificación de los analistas.
5	**Aexp**		Experiencia en el tipo de aplicación.
6	**Pcap**		Calificación de los programadores.
7	**Vexp**		Experiencia en la máquina virtual.
8	**Lexp**		Experiencia en el lenguaje de programación.
9	**Modp**	Proyecto	Prácticas de programación.
10	**Tool**		Empleo de herramientas de desarrollo.
11	**Sced**		Problemas de planificación.

A partir de estos atributos, se construyeron cuatro sets de datos diferentes. Esta distribución se corresponde con los modos de clasificación de los proyectos de software que sugiere el modelo COCOMO [13] y los criterios establecidos por Galinina y colaboradores [14] y por Bedini [15].

DS1: Incluye todos los proyectos (228 instancias) y la clase es discreta. Para ello, se hizo necesario realizar un procedimiento de discretización de los atributos de la base de datos cocomo81. Empleando como criterio la escala propuesta por el modelo COCOMO [13]. Este set de datos se emplea para entrenar el primer nivel del multiclasificador.

DS2: Incluye solo los proyectos catalogados como orgánicos, por debajo de las 50 mil líneas de código (74 instancias). Este set de datos se emplea para entrenar el clasificador especializado en proyectos orgánicos, en el segundo nivel del multiclasificador.

DS3: Incluye solo los proyectos catalogados como semi-libres, entre 50 y 300 mil líneas de código (105 instancias). Este set de datos se emplea para entrenar el clasificador especializado en proyectos semi-libres, en el segundo nivel del multiclasificador.

DS4: Incluye solo los proyectos catalogados como empotrados, mayores de 300 mil líneas de código (49 instancias). Este set de datos se emplea para entrenar el clasificador especializado en proyectos empotrados, en el segundo nivel del multiclasificador.

B. Evaluación del modelo

EEpred es un multiclasificador en serie, donde cada nivel tiene diferentes responsabilidades. En el primer nivel se realiza clasificación y, en el segundo, regresión. Es por ello que se hace necesario contar con diferentes estadígrafos, tanto discretos como continuos, que permitan medir adecuadamente el desempeño en cada uno de estos niveles. Es importante tener en cuenta que muchas de las medidas comúnmente empleadas para evaluar la efectividad de los predictores no funcionan en predicción numérica. Los principios básicos (empleando validación cruzada para la evaluación del desempeño), son igualmente aplicables a la predicción numérica. Pero la calidad de la medición ofrecida por la razón de error no es muy apropiada: los errores no están, simplemente, presentes o ausentes; ellos se presentan en diferentes magnitudes [16].

Es por ello que para la evaluación del desempeño del predictor en el primer nivel fueron empleadas medidas discretas. Éstas fueron: precisión, recuerdo o sensibilidad y la media armónica [17]. Otra medida empleada para evaluar el desempeño del predictor en el primer nivel, es el área bajo la curva de operación (ROC). La cual permitirá conocer el desempeño del predictor para cada clase [17].

Mientras que para el segundo nivel se emplearon medidas continuas. Estas medidas fueron: la media del error relativo y el coeficiente de correlación (***CC***), el cual toma valores en el rango de 1 para los resultados perfectamente correlacionados, 0 para la no correlación, y –1 para los perfectamente correlacionados negativamente [17].

III. MODELO PROPUESTO

La estimación del esfuerzo de un proyecto de software se considera compleja, debido a que es un proceso altamente subjetivo. Esto, unido a la gran variabilidad de los datos, lo convierte en un problema de espacio desconocido de clasificadores, donde es complicado encontrar el clasificador óptimo. Por tal razón, proponemos una solución basada en la combinación de clasificadores [18].

Como se muestra en la Fig. 1, se propone un método que combina los resultados de un clasificador que realiza una predicción granular (primer nivel). En este primer nivel solo determina si el proyecto de software será: orgánico (***O***); semi-libre (***SL***); o empotrado (***E***). Además, brinda un rango en que podría encontrarse el número de líneas de código que podría tener el proyecto. El segundo nivel actúa en función de los resultados del primero. Donde, además de la información sobre el proyecto, se agrega la predicción sobre el rango de LOC del proyecto. En este nivel existe un clasificador especializado en cada tipo de proyecto, capaz de estimar el valor del esfuerzo que conllevaría el mismo.

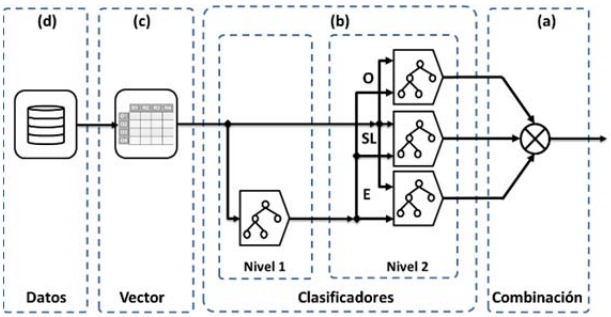

Figura 1. Esquema del modelo propuesto. (a) nivel de combinación de los resultados, (b) nivel de clasificadores base, (c) nivel de vector, y (d) nivel de datos.

Al igual que otros modelos de combinación de clasificadores, EEpred emplea diferentes algoritmos de construcción de los clasificadores base. No obstante, cada nivel es entrenado con bases de datos diferentes. Todo lo cual garantiza la diversidad. Donde, el primer nivel emplea la base de datos DS1. En el segundo nivel cada clasificador se entrena

con DS2, DS3 o con DS4, en dependencia de si se especializarán en proyectos orgánicos, semi-libres o empotrados, respectivamente. Esto garantiza que cada clasificador logre una mejor estimación del esfuerzo en su categoría.

A. Clasificador base

En el primer nivel, la predicción es tratada como un problema de clasificación. No obstante, una característica deseada del modelo propuesto es su capacidad explicativa, razón por la cual fueron escogidos algoritmos basados en árboles. Debido a que cada rama del árbol puede ser traducida como un conjunto de reglas de tipo ***Si–Entonces***, lo cual es fácilmente interpretable.

Por otra parte, los árboles pueden procesar clases binarias o múltiples, así como atributos numéricos y nominales, e incluso valores perdidos. Además de resultar eficientes y poco consumidores de recursos, han sido empleados satisfactoriamente en la EEDS [19].

Algoritmos de construcción de árboles empleados:

- J48: es la implementación del algoritmo C4.5 [20] presente en la herramienta Weka. Este algoritmo se puede considerar un híbrido entre CART y C4.5. Puede trabajar con conjuntos de datos categóricos y continuos [21].
- RandomTree: construye el árbol a partir de K atributos seleccionados aleatoriamente en cada nodo. Además, calcula la probabilidad de las clases mediante un procedimiento *hold-out*. Puede ser empleado tanto para decisión como para regresión [22].
- REPTree: Construye un árbol de decisión/regresión empleando la información de varianza y realiza la poda usando como criterio la reducción del error. Funciona en dos fases (datos de aprendizaje y datos de poda). Primero se crea un conjunto de reglas que se sobre ajusta a los datos usados para el aprendizaje, después poda el conjunto de reglas usando ejemplares que no participaron en el aprendizaje [22].
- M5P: éste es un algoritmo de construcción de árboles de regresión lineal, basado en M5 de Quinlan [20], [23].

Otros algoritmos de construcción de árboles no fueron tomados en cuenta debido a que sus características propias, no permitían su aplicación.

En el segundo nivel la predicción se trata como un problema de regresión. Por lo que se realizó el análisis con los árboles de regresión RandomTree [22], REPTree [22] y M5P [23].

B. Combinación de los Resultados

Existen varios métodos de combinación de la clasificación. Entre los que se destacan el voto mayoritario, voto mayoritario ponderado, recuento de Borda, *naive* Bayes, entre otros, ya sean a nivel abstracto, de ranking o de medición [18]. Para el modelo propuesto, se decidió por una combinación a nivel de medición, ya que se conoce previamente cuál es la efectividad de cada clasificador base del segundo nivel para cada uno de los tipos de proyectos (ecuación 1).

$$E = \sum_{j=1}^{P} w_j * e_j \quad (1)$$

Donde, $\boldsymbol{E}$ es la estimación final, $\boldsymbol{w_j}$ el peso asignado por el clasificador del primer nivel, $\boldsymbol{e_j}$, la estimación realizada por el clasificador del segundo nivel y $\boldsymbol{P}$, el tipo de proyecto. Si se toma en cuenta que el clasificador del primer nivel devuelve como resultado la probabilidad de que el proyecto sea de una clase y no de las otras dos, entonces el resultado de la combinación sería equivalente a la selección del clasificador adecuado.

IV. ANÁLISIS Y DISCUSIÓN

El método propuesto, EEpred, se considera como un modelo claramente definido, con un algoritmo no ambiguo y de fácil reproducción. El primer paso, para la construcción del multiclasificador, es la selección del clasificador base. Una vez seleccionado el clasificador base, se realiza el análisis del desempeño del multiclasificador. Para ello, primeramente, se realizó una evaluación de cada variable en el problema (A). Posteriormente, EEpred fue evaluado en cuanto a: robustez y capacidad de generalización (B), comportamiento del error (C), y mecanismo de interpretación (D).

A. Evaluación de las variables

Tomando en consideración que la EEDS es un problema no resuelto. El primer procedimiento que se realizó fue el análisis evaluativo de la influencia de cada variable en el problema. Para ello, se decidió emplear como criterio de evaluación, las medidas basadas en la teoría de la información: correlación entre variables, razón de ganancia y ganancia de información [24] (Tabla III).

TABLA III
RESULTADOS DE LA EVALUACIÓN DE LAS VARIABLES

Evaluación	Orden de los atributos	Mérito
Correlación	7, 5, 4, 2, 10, 1, 9, **3**, 8, **6, 11**	0,54
Razón de ganancia	5, 2, 7, 4, 8, 9, 10, **3, 6,** 1**, 11**	0,47
Razón de información	2, 5, 9, 7, 4, 8, 10, **3, 6,** 1**, 11**	0,52

La Tabla III muestra que los atributos 3, 6 y 11, los cuales representan las variables Cplx, Pcap y Sced (ver Tabla I), son los que menos aportan al problema. Teniendo en cuenta esto, fueron retirados del vector de entrada de los algoritmos propuestos.

B. Selección del clasificador base

Con el objetivo de seleccionar el clasificador base, en el primer nivel, fueron evaluados los algoritmos de construcción de árboles J48, RandomTree, y REPTree, en la base de datos DS1 (Tabla IV).

La Tabla IV muestra que el comportamiento de estos algoritmos es muy similar. Sin embargo, tomando en cuenta las buenas y malas clasificaciones, así como la magnitud del MRE, las medidas discretas, así como el área bajo la curva ROC, se determinó que RandomTree presenta mejor desempeño que J48, y REPTree.

TABLA IV
COMPARATIVA ENTRE ÁRBOLES DE DECISIÓN. DONDE P ES LA PRECISIÓN, R EL RECUERDO, FM ES LA MEDIA ARMÓNICA, ROC LA CURVA DE OPERACIÓN Y MRE LA RAÍZ DEL ERROR CUADRÁTICO MEDIO

Clasificadores	Reglas	Clasificaciones		Estadísticas				
		Buenas	Malas	MRE	P	R	Fm	ROC
J48	20	156	72	79,82	0,64	0,68	0,65	0,65
RandomTree	**82**	**164**	**64**	**63,60**	**0,71**	**0,72**	**0,72**	**0,75**
REPTree	17	153	75	85,78	0,62	0,67	0,64	0,67

Por otra parte, una de las características más importantes de los árboles es la capacidad de convertir su base de conocimientos en reglas entendibles. Por tanto, un indicador importante es el número de reglas que generan. Sin embargo, 82 no es un número de reglas que no pueda ser manejado.

De esta manera, tomando en cuenta que RandomTree es el mejor situado en cuanto a su desempeño y a que genera un número de reglas fácilmente manejable, se tomó la decisión de emplearlo como clasificador base del primer nivel.

En el segundo nivel la estimación se trata como un problema de regresión, debido a que se desea predecir el valor numérico del esfuerzo necesario para el desarrollo del software. Por lo que se realizó el análisis con los árboles de regresión RandomTree, REPTree y M5P. En la Tabla V se muestran los resultados experimentales.

TABLA V
COMPARATIVA ENTRE ÁRBOLES DE REGRESIÓN. DONDE R ES EL NÚMERO DE REGLAS, CC EL COEFICIENTE DE CORRELACIÓN Y MRE EL ERROR RELATIVO MEDIO

Clasificadores	Orgánico			Semi-libre			Empotrado		
	R	CC	MRE	R	CC	MRE	R	CC	MRE
RandomTree	253	0,26	87,80	**65**	**0,37**	**79,49**	43	0,32	90,22
REPTree	49	0,33	96,59	19	0,32	94,36	**1**	**0,40**	**97,53**
M5P	**6**	**0,47**	**79,48**	1	0,29	109,75	2	0,10	94,35

En la Tabla V se observa que, el comportamiento de los algoritmos de regresión varía en dependencia del tipo de proyecto. Por lo que se decidió emplear para los proyectos **orgánicos**, el M5P; para los proyectos **semi-libres**, el RandomTree; y para los proyectos **empotrados**, el REPTree.

C. Robustez y capacidad de generalización

Para analizar la robustez y la capacidad de generalización del algoritmo, EEpred fue sometido a un proceso de validación interna, donde se aplicó la validación cruzada para 10 muestras (*10-fold cross-validation*). Este procedimiento se realizó tomando en cuenta el tipo de software a predecir (Tabla VI), por lo que se evaluó la capacidad de estimación de EEpred para proyectos que califican como orgánicos, semi-libres, empotrados y en general.

TABLA VI
RESULTADOS EXPERIMENTALES DEL PROCESO DE VALIDACIÓN INTERNA. DONDE P ES LA PRECISIÓN, R EL RECUERDO, FM ES LA MEDIA ARMÓNICA Y ROC LA CURVA DE OPERACIÓN

Proyectos	EEpred			
	P	R	Fm	ROC
Orgánico	0,78	0,82	0,80	0,76
Semi-libre	0,63	0,59	0,61	0,77
Empotrado	0,36	0,31	0,33	0,61
General	0.71	0.72	0.72	0.75

Como se puede observar en la Tabla VI, EEpred se comporta relativamente bien para los proyectos orgánicos y semi-libres, no siendo así para los empotrados cuya precisión cae drásticamente. Un análisis más detallado se puede realizar si se observa la Fig. 2 Donde, se representan gráficamente las medidas de calidad de EEpred, para cada tipo de proyecto por separado y en general.

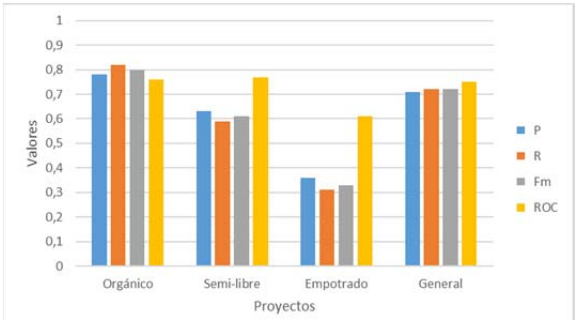

Figura 2. Diseño experimental de la evaluación del modelo propuesto.

La Fig. 2, muestra que EEpred presenta mejor precisión (P) a la hora de identificar los proyectos orgánicos, logrando un 78%. Lo mismo sucede con la capacidad de recuerdo (R), área bajo la curva (ROC) y la media armónica (Fm). No obstante, el desempeño de EEpred para los proyectos de tipo semi-libre es superior al 60%. Aunque, su capacidad de recuerdo es ligeramente inferior a la precisión alcanzada.

Sin embargo, un resultado interesante se muestra cuando EEpred es evaluado en una base de datos que contiene todo tipo de proyectos. Esto se debe a que, a pesar de su bajo desempeño en proyectos de tipo empotrado, éstos solo representan el 21% del total. Lo cual hace que el desempeño en general del algoritmo sea aceptable, 71%. Demostrando así que el esquema de multiclasificación presentado en EEpred brinda una estimación superior a la estimación con algoritmos individuales.

D. Comportamiento del error

Con el objetivo de estudiar el comportamiento del error del modelo propuesto, se construyó un gráfico que representa la matriz de confusión del primer nivel de clasificación, empleando el árbol de decisión RandomTree (Fig. 3).

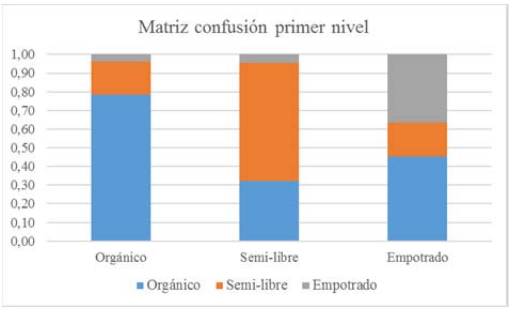

Figura 3. Matriz de confusión del primer nivel de clasificación, empleando el árbol de decisión RandomTree.

La Fig. 3 muestra los aciertos y desaciertos del primer nivel de clasificación, para cada uno de los tipos de proyectos.

Donde, en el eje de las abscisas se representan las clases de los proyectos que debía clasificar este nivel. Mientras que, en el eje de las ordenadas, se muestra el porcentaje que representa la clasificación real que alcanzó este nivel de clasificación.

Como resultado, en este nivel, se logran clasificar correctamente el 78% de los proyectos de tipo **orgánicos**. Cuando se intentan clasificar los proyectos de tipo **semi-libres**, solo 63% es correctamente identificado, mientras que el predictor tiende a identificar al 32% de los proyectos como **orgánicos**. Por otra parte, en el caso de los proyectos de tipo **empotrado**, solo se logra clasificar bien el 36% de éstos. Debido a que el clasificador tiende a ver al 45% de los proyectos como **orgánicos** y al 18% como **semi-libres**.

De este análisis, se puede concluir que el modelo propuesto es capaz de estimar el esfuerzo con calidad en proyectos **orgánicos** y **semi-libres**. Mientras que, debido al bajo número de muestras de que se dispone para entrenar al predictor en proyectos de tipo **empotrados**, el error en la clasificación aumenta significativamente.

Este comportamiento se puede corroborar si se analiza el error relativo medio que comete el modelo propuesto al estimar el esfuerzo para cada una de las clases en el segundo nivel. Lo cual se puede observar en la Fig. 4. Representándose la calidad de la estimación, en un diagrama de dispersión.

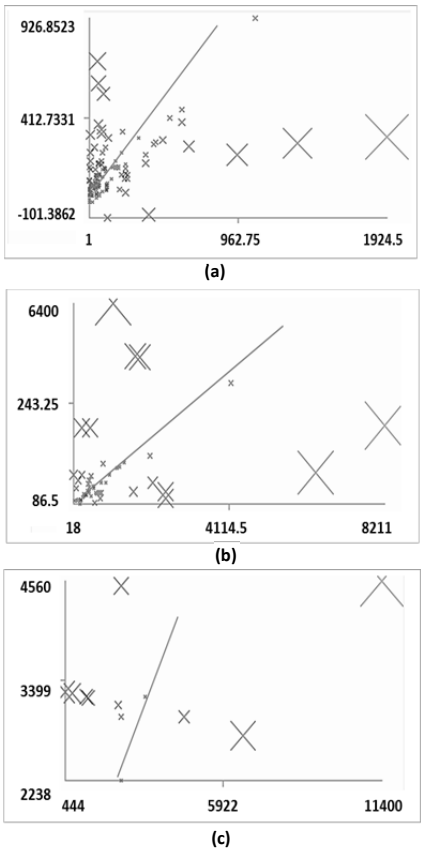

Figura 4. Representación del error cometido por el segundo nivel del multiclasificador en la EEDS. El tamaño de las cruces se corresponde con la magnitud del error. En el eje de las abscisas, se representan los valores de esfuerzo real. En el eje de las ordenadas, se representan los valores del esfuerzo estimado por cada clasificador. (a) Estimación en proyectos orgánicos. (b) Estimación en proyectos semi-libres. (c) Estimación en proyectos empotrados.

En la Fig. 4a (representación del error del estimador del esfuerzo en proyectos **orgánicos**), se observa que a medida que aumenta el esfuerzo necesario para el desarrollo del software, el estimador tiende a realizar una sub estimación de su valor. Sin embargo, la mayoría de las estimaciones se mantienen alrededor de la tendencia normal.

Mientras que, en la Fig. 4b (representación del error del estimador del esfuerzo en proyectos **semi-libres**), se evidencia que la estimación mejora para los valores pequeños. Mientras que, para los valores grandes, al igual que en la Fig. 4c (representación del error del estimador del esfuerzo en proyectos **empotrados**), se presenta una amplia dispersión en los resultados de la estimación.

E. Mecanismo de interpretación

Muchos de los modelos existentes para la estimación del esfuerzo de proyectos de software son cajas negras. Sin embargo, los árboles son capaces de describir explícitamente la naturaleza de su predicción.

Debido a que el árbol generado en el primer nivel tiene 82 reglas, a modo de ejemplo, se tomará de muestra solo dos ramas del mismo. De las cuales se pueden extraer dos reglas de decisión, las que pueden ser transformadas en tres pasos:

- **Conversión**: en este paso, cada rama del árbol se convierte en una implicación lógica.
  a) ***(Vexp = n) ∧ (Data = l) ∧ (Modp = n) ∧ (Lexp = h) ∧ (Aexp = h) → O (4/0)***
  b) ***(Vexp = n) ∧ (Data = l) ∧ (Modp = n) ∧ (Lexp = h) ∧ (Aexp = vh) → O (12/0)***
- **Refinamiento**: en este paso, se buscan patrones coincidentes y se simplifica la proposición. Por ejemplo, ***(Aexp = h) → O ∧ (Aexp = vh) → O*** por tanto, la proposición quedaría:
  (Vexp = n) ∧ (Data = l) ∧ (Modp = n) ∧ (Lexp = h) ∧ (Aexp = h, vh) → O (16/0)
- **Traducción**: en este paso, la proposición se traduce empleando el significado de las variables y se convierte en una regla de tipo ***Si–Entonces***.
  "Si la experiencia en la máquina virtual es nominal **Y** la base de datos es pequeña **Y** la práctica de programación es nominal **Y** la experiencia del analista en este tipo de aplicación es alta o muy alta, ***Entonces*** el tipo de proyecto es **orgánico** y esfuerzo se estima por la ecuación (2)**"**.

$$\begin{aligned}\text{Esfuerzo} = {} & 117{,}0168 * \text{rely} = \text{n, xh, h, vh} \\ & + 176{,}7444 * \text{cplx} = \text{vh, xh} \\ & + 79{,}29 * \text{vexp} = \text{xh, h, l, vl} \\ & + 118{,}5762 * \text{sced} = \text{vh, xh, h} \\ & - 120{,}4674\end{aligned} \tag{2}$$

Donde, los pesos de la ecuación resultante para del cálculo del esfuerzo está en dependencia del clasificador base que devolvió la regla, ya sea el M5P, el REPTree o el RandomForest.

Adicionalmente, cada una de estas reglas está acompañada

por el valor de la cobertura y la precisión. Lo cual permite al especialista que la interpreta conocer su grado de certeza.

V. CONCLUSIONES

En este trabajo se introduce EEperd, un método de estimación del esfuerzo de desarrollo de proyectos de software basado en un multiclasificador en serie. EEpred está formado por dos niveles de clasificación en serien. Donde, el primer nivel realiza una clasificación granular, basada en un árbol de decisión, que determina si el proyecto será ***orgánico***, ***semi-libre*** o ***empotrado***. Y, posteriormente, realiza una predicción fina, basada en la combinación de los resultados de tres árboles de regresión, encargados de calcular el valor del esfuerzo estimado. EEpred, fue sometido a un proceso de validación interna que permitió analizar su robustez y capacidad de generalización, así como el error en la predicción. Lo cual demostró que EEpred es capaz de predecir el esfuerzo de proyectos ***orgánicos*** *y* ***semi-libres***, con elevada calidad, 78% y 63% respectivamente, no siendo así para proyectos de tipo ***orgánico***. Y logra una precisión general de hasta el 71%, lo cual demuestra que el modelo de multiclasificación brinda una estimación superior a la estimación con algoritmos individuales. La principal ventaja de EEpred es que implementa un mecanismo de interpretación de los resultados, sobre la base de un conjunto de reglas semánticas que son de fácil entendimiento. Lo cual lo convierte en un modelo explicativo.

AGRADECIMIENTOS

Esta investigación se inserta en el programa doctoral Iberoamericano en Soft Computing, desarrollado por la Universidad de Las Villas, Cuba y la Universidad de Granada, España. Bajo el auspicio de la Junta de Andalucía y la AUIP. Especial agradecimiento a todo aquel que contribuyó con sus invaluables comentarios y consideraciones.

REFERENCIAS

[1] K. Dejaeger, W. Verbeke, D. Martens, and B. Baesens, "Data Mining Techniques for Software Effort Estimation: A Comparative Study," *IEEE Trans. Softw. Eng.*, vol. 38, no. 2, pp. 375–397, 2012.

[2] J. Wen, S. Li, Z. Lin, Y. Hu, and C. Huang, "Systematic literature review of machine learning based software development effort estimation models," *Inf. Softw. Technol.*, vol. 54, no. 1, pp. 41–59, 2012.

[3] M. Algabri, F. Saeed, H. Mathkour, and N. Tagoug, "Optimization of soft cost estimation using genetic algorithm for NASA software projects," in *2015 5th National Symposium on Information Technology: Towards New Smart World (NSITNSW)*, pp. 1–4, 2015.

[4] M. G. A. C, G. Raura, J. A. R. R, and E. R. F. C, "Modelo Neuronal de Estimación para el Esfuerzo de Desarrollo en Proyectos de Software (MONEPS)," *Rev. Latinoam. Ing. Softw.*, vol. 3, no. 3, pp. 148–154, 2015.

[5] S. Waghmode and K. Kolhe, "A Novel Way of Cost Estimation in Software Project Development Based on Clustering Techniques," *Int. J. Innov. Res. Comput. Commun. Eng.*, vol. 2, no. 4, pp. 3892–3899, 2014.

[6] J. G. Borade and V. R. Khalkar, "Software Project Effort and Cost Estimation Techniques," *Int. J. Adv. Res. Comput. Sci. Softw. Eng.*, vol. 3, no. 8, pp. 730–739, 2013.

[7] L. L. Minku and X. Yao, "Ensembles and locality: Insight on improving software effort estimation," *Inf. Softw. Technol.*, vol. 55, no. 8, pp. 1512–1528, 2013.

[8] H. Najadat, I. Alsmadi, and Y. Shboul, "Predicting Software Projects Cost Estimation Based on Mining Historical Data," *ISRN Softw. Eng.*, vol. 2012, pp. 1–8, 2012.

[9] S. Kad and V. Chopra, "Software Development Effort Estimation Using Soft Computing," *Int. J. Mach. Learn. Comput.*, vol. 2, no. 5, pp. 548–551, 2012.

[10] M. Saroha and S. Sahu, "Tools & methods for software effort estimation using use case points model – A review," in *International Conference on Computing, Communication & Automation*, pp. 874–879, 2015.

[11] Danh Nguyen-Cong and De Tran-Cao, "A review of effort estimation studies in agile, iterative and incremental software development," in *The 2013 RIVF International Conference on Computing & Communication Technologies - Research, Innovation, and Vision for Future (RIVF)*, pp. 27–30, 2013.

[12] J. . Sayyad Shirabad and T. J. Menzies, "The PROMISE Repository of Software Engineering Databases," Ottawa, Canada, 2005.

[13] A. F. Sheta, "Estimation of the COCOMO model parameters using genetic algorithms for NASA software projects," *J. Comput. Sci.*, vol. 2, no. 2, pp. 118–123, 2006.

[14] A. Galinina, O. Burceva, and S. Parshutin, "The Optimization of COCOMO Model Coefficients Using Genetic Algorithms," *Inf. Technol. Manag. Sci.*, vol. 15, no. 1, p. 15, Jan. 2012.

[15] A. Bedini, *Gestión de Proyectos de Software*, 2da ed. Valparaiso, Chile: Universidad Técnica Federico Santa María, 2005.

[16] I. H. Witten and E. Frank, *Data Mining: Practical Machie Learning Tools and Techniques*, 2th ed. Morgan Kaufmann, 2005.

[17] S. García, A. Fernández, J. Luengo, and F. Herrera, "A study of statistical techniques and performance measures for genetics-based machine learning: accuracy and interpretability," *Soft Comput*, vol. 13, pp. 959–977, 2009.

[18] L. I. Kuncheva, "Multiple Classifier Systems," in *Combining Pattern Classifiers. Methods and Algorithms*, 1st ed., Wiley Interscience, pp. 101–110, 2004.

[19] I. M. Alsmadi and M. S. Nuser, "Evaluation of Cost Estimation Metrics: Towards a Uni fi ed Terminology," *J. Comput. anf Inf. Technol.*, vol. 21, no. 1, pp. 23–34, 2013.

[20] N. Ramakrishnan, "Chapter 1. C4.5," in *The Top Ten Algorithms in Data Mining*, Taylor & Francis Group, LLC, 2009, pp. 1–19.

[21] I. H. Witten, E. Frank, L. Trigg, M. Hall, G. Holmes, and S. J. Cunningham, "Weka: Practical Machine Learning Tools and Techniques with Java Implementations."

[22] Y. Zhao and Y. Zhang, "Comparison of decision tree methods for finding active objects," *Adv. Sp. Res.*, Aug. 2007.

[23] D. Steinberg, "Chapter 10. CART: Classification and Regression Trees," in *The Top Ten Algorithms in Data Mining*, Taylor & Francis Group, LLC, pp. 179–201, 2009.

[24] D. J. Mackay, *Information Theory, Inference, and Learning Algorithms*, 1st ed. Cambridge, United Kingdom: University, Cambridge, 2003.

Héctor Velarde es doctorando en TIC con Diploma de Estudios Avanzados en Soft Computing de la Universidad de Granada, España; Magíster en Ingeniería de Software de la Universidad Católica de Santa María de Arequipa y titulado como Ingeniero de Sistemas en la Universidad Católica LACh., Perú. Actualmente se desempeña como Docente Principal e Investigador en la Universidad Católica de Santa María de Arequipa, Perú. Sus intereses en investigación se centran en minería de datos e ingeniería de software.

Cosme Santiesteban es Doctor en Ciencias de la Computación en la Universidad de Granada, España en el 2015, Máster en Inteligencia Artificial e Informática Aplicada en el 2010 y 2004 respectivamente. Es Investigador a tiempo completo en el Centro de Bioplantas y Profesor Auxiliar a tiempo parcial en la Universidad de Ciego de Ávila, Cuba. Se interesa en el reconocimiento de patrones, inteligencia artificial, procesamiento digital de imágenes, técnicas de soft computing y computación paralela/distribuida.

Ana García es Doctora en Ciencias Técnicas del Instituto Politécnico José A. Echevarría de la Habana, Cuba y Licenciada en Cibernética Matemática de la Universidad Central de las Villas. Actualmente se desempeña como docente Titular en la Universidad Central de las Villas y es Sub Directora Comercial de DESOFT en Villa Clara, Cuba., Cuba. Sus intereses en investigación se centran en sistemas de información, desarrollo de aplicaciones e ingeniería de software.

Jorge Casillas recibió el Máster y el Doctorado en Ciencias de la Computación en 1998 y 2001 respectivamente, en la Universidad de Granada, España. Es Profesor Asociado en el Departamento de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada, donde es miembro del grupo de investigación en Soft Computing y Sistemas de Información Inteligentes. Se interesa en *big data*, *data stream*, sistemas difusos, algoritmos evolutivos, robótica, sistemas inteligentes, entre otros.