**Universidad Nacional**

**Sistema de Información Académica**

**Propuesta de proyecto académico**

SIGESA-SIA Fecha: 26/05/2023 03:51 PM

## 1 Información general básica

| | |
|---|---|
| **1.1 Código SIA** | 0127-23 |
| **1.2 Fase** | 1 |
| **1.3 Nombre proyecto antecesor** | |
| **1.4 Título del proyecto** | **Un estudio empírico de la configuración de hiperparámetros en el contexto de la predicción de densidad de defectos de software mediante el uso de algoritmos bioInspirados** |
| **1.5 Vigencia** | |
| **Fecha inicio** | 17/07/2023 |
| **Fecha fin** | 17/07/2026 |
| **1.6 Instancias y personas ejecutoras** | |
| **1.6.1 Unidad titular o principal** | Escuela de Informática y Computación |
| **1.6.2 Jornada total ( horas pagas )** | 30 |

**1.6.3 Persona responsable**

| Nombre | Jornada | Tipo de jornada | Instancia académica o afín | Tipo de participante |
|---|---|---|---|---|
| M U R I L L O MORERA JUAN DE DIOS | 20 | PAGAS | Escuela de Informática y Computación | FUNCIONARIO |

**1.6.4 Otros participantes**

| Nombre | Jornada | Tipo de jornada | Instancia académica o afín | Tipo de participante |
|---|---|---|---|---|
| | 10 | PAGAS | Escuela de Informática y Computación | POR DEFINIR - ESTUDIANTE |

**1.7 Áreas académicas**

- Investigación

**1.8 Tipo de investigación**

APLICADA

### 1.9 Mayor componente

No hay un mayor componente seleccionado.

### 1.10 Abordaje

DISCIPLINAR

### 1.11 Adscripción

#### 1.11.1 Aporte del proyecto al programa de adscripción

No se ha descrito un aporte para el proyecto.

## 2 Información técnica

### 2.1 Resumen

El uso de la minería de datos en el contexto de ingeniería de software ha existido desde mediados 1990. La combinación de estos dos campos abre toda una brecha de investigación de gran interés para la mejora del proceso de desarrollo del software, siendo posible obtener a partir de un conjunto de datos las métricas que describen cómo evoluciona dicho proceso en función de resultados esperados: tiempo, costo y calidad.

Otro aspecto importante para considerar y que le da relevancia a la combinación de estas dos áreas de investigación, es el acelerado crecimiento de los repositorios de información, lo cual complica aún más la tarea de poder clasificar y obtener categorías representativas de métricas que le permitan a los expertos llegar de una forma más rápida y eficiente al análisis y corrección de posibles problemas dentro del ciclo de desarrollo del software.

Basado en lo anterior, es que nace una interesante área de investigación: La predicción de fallos de software, cuyo aporte irá destinado a la mejora del proceso de construcción del software, mediante la temprana detección de errores y de la corrección de los ya existentes, manteniendo la calidad de este a través de un mejor uso de la evaluación y de la asignación de recursos.

Varios son los factores que pueden afectar en el correcto desempeño de los modelos predictivos, por ejemplo: la selección de la base de datos histórica, la técnica de pre-procesamiento de los datos, la técnica de selección de atributos (predictores) y la selección de los algoritmos de aprendizaje. Sin embargo, pocos son los estudios que han experimentado con la automatización de las diferentes configuraciones que se le puede dar a un mismo algoritmo de aprendizaje en un mismo contexto. Esta situación ha hecho que los investigadores utilicen para sus experimentaciones la configuración "default" de estos algoritmos, lo cual no garantiza necesariamente que sea la que brinde el mayor desempeño dado las características del contexto.

Es por esta razón que se está proponiendo variar la configuración de los diferentes algoritmos de aprendizaje, la configuración de parámetros de estos algoritmos mediante la elección de los hiperparámetros, mediante el uso de los algoritmos bioinspirados. Trabajando así, de una manera más automática. Convirtiéndose así la investigación en un problema de optimización.

Dentro de los productos esperados están: una Revisión Sistemática de Literatura, un Modelo Formal, un Programa Computacional y un Diseño Experimental de Validación del Modelo.

### 2.1.1 Vinculación con los Objetivos de Desarrollo Sostenible (ODS) y sus metas

| ODS | Metas |
|---|---|
| Objetivo 8. Promover el crecimiento económico sostenido, inclusivo y sostenible, el empleo pleno y productivo y el trabajo decente para todos. | 8.2 Lograr niveles más elevados de productividad económica mediante la diversificación, la modernización tecnológica y la innovación, entre otras cosas centrándose en los sectores con granvalor añadido y un uso intensivo de la mano de obra |

### 2.1.2 Acción estratégica del PMPI

No hay acciones estratégicas.

### 2.1.3 Áreas estratégicas del conocimiento institucionales

| Área | Subárea |
|---|---|
| Desarrollo CientíficoTecnológico e Innovación | Prospección y Políticas del Desarrollo Científico y Tecnológico |

### 2.1.4 Áreas temáticas o estratégicas de la (s) instancias (es) participante (es)

| Unidad ejecutora | Área temática |
|---|---|
| Escuela de Informática y Computación | |

### 2.1.5 Descripción del problema

Actualmente la industria de desarrollo del software ha tenido un acelerado crecimiento. La diversidad y complejidad tanto de los proyectos como de las herramientas para la elaboración de éstos, ha hecho que la posibilidad de cometer un error por parte de los desarrolladores sea mayor, lo cual requiere una correcta documentación, modularización y reutilización de la programación hecha.

Los proyectos de software presentan varios problemas asociados a los defectos producidos durante su desarrollo. Dentro de los más destacables se tienen: baja calidad en los productos entregados, gastos considerables de dinero, exceso de pruebas (testing) y retrasos considerables en las entregas de los productos.

La predicción de defectos de software es el proceso de desarrollo de modelos que pueden ser usados por los profesionales en etapas tempranas del ciclo de vida del software para detectar fallos a nivel de clases o de módulos [7]. Esta ha sido una importante área de investigación en el campo de la ingeniería de software por más de 30 años [7], [4], [8], [9], [10],[11], [12], [13], [14], [15], [16], colaborando en la creación de modelos predictivos caracterizados por un mayor desempeño en cuanto a la predicción de defectos.

Con base a los problemas descritos es que la predicción de defectos de software tiene como principal objetivo la creación de nuevos modelos de predicción y la correcta configuración de estos, con el fin de mejorar el desempeño en las predicciones. Permitiendo de esta manera disminuir o eliminar el efecto negativo que puedan tener los problemas mencionados.

Definición del problema

Actualmente, el área de predicción de defectos presenta tres líneas de investigación: estimación del número de defectos en los productos de software, asociaciones de defectos, así como sus relaciones y clasificación de módulos o clases en propensos a fallos y no propensos a fallos [4].

La primera línea de investigación emplea métodos estadísticos para estimar el número de defectos o densidad de defectos [19], [20], [21], [22], [23], [24]. La segunda línea de investigación se centra en las asociaciones entre defectos, es decir la relación que existen entre ellos [25]. Finalmente, la tercera línea de investigación está basada en la clasificación de los componentes de software en propensos y no propensos a fallos: [7], [10], [12], [26], [27], [28], [29], [30], [31], [32], [33], [34] y [35]. La presente investigación sigue el primer enfoque por ser una la línea emergente en cuanto a la cantidad de investigaciones que presentan una serie de problemas abiertos vigentes, conferencias y revistas de alto impacto.

La línea de investigación correspondiente a clasificación es un problema en constante cambio, ya que diferentes modelos y sus combinaciones dan diferentes resultados para proyectos de software iguales. Lo que conlleva a que un esquema de aprendizaje (pre-procesamiento, selección de atributos y algoritmo de aprendizaje) [4] y [37] sea considerado como uno de los más importantes métodos de configuración para un proyecto determinado, no siendo esto una tarea fácil debido a que requiere de considerable experimentación.

Los investigadores han estado utilizando los esquemas de aprendizaje como una forma más precisa de configuración, haciendo combinaciones manuales a nivel de esquemas de aprendizaje [4] y [37].

En el área de predicción de densidad de fallos, la investigación se ha centrado tanto en la selección de las métricas como la combinación de las técnicas o algoritmos de aprendizaje. La métrica se puede definir como la medida del grado de un sistema, componente o proceso respecto a un atributo. Las métricas también pueden ser definidas como formas de medición representadas por diferentes escalas [38]. Dentro de las más comunes en el campo de predicción de defectos.

están: las de complejidad ciclomática McCabe [39] y las de manejo de operadores Halstead [40]. Tanto McCabe como Halstead son métricas basadas en módulos o en un módulo, donde un módulo se puede

definir como la unidad más pequeña de funcionalidad (en otros lenguajes los módulos pueden ser llamadas: (función o método). Este tipo de métricas son muy utilizadas para la medición de atributos de código estático por ser fáciles de recolectar y usar [37].

Las técnicas o algoritmos de aprendizaje se caracterizan por llevar a cabo transformaciones sobre los datos con el objetivo de encontrar de forma automática relaciones, asociaciones y patrones.

Lo anterior ha producido que se generen nuevos problemas abiertos de importancia en el área de la predicción de fallos de software [42]:

1. Se deben realizar más estudios en el área de predicción de defectos que utilicen algoritmos de aprendizaje con el fin de obtener resultados más generalizables.

2. Se recomiendan estudios que lleven a cabo comparaciones de desempeño entre algoritmos de aprendizaje y técnicas de regresión logística.

3. Se requieren más estudios que examinen la capacidad predictiva de algoritmos de aprendizaje menos frecuentes tales como: RBL, LB, AB, Bagging, RBF y ADT.

4. Es necesaria una mayor investigación que compruebe la efectividad de algoritmos evolutivos en el campo de predicción de defectos. Como por ejemplo los algoritmos genéticos.

5. La correcta configuración de los parámetros de los diferentes algoritmos de aprendizaje. Lo anterior, debido a que por cada algoritmo de aprendizaje existen varias combinaciones posibles de configuración que pueden ser tendientes a diferentes resultados de desempeño de los modelos (AUC).

6. Experimentar más con los proyectos de software privados que con los de libre acceso. Lo anterior con el fin de medir la efectividad de los algoritmos de aprendizaje en ambos casos.

A su vez, se han llevado a cabo varios estudios experimentales que ratifican la necesidad de buscar modelos que mejoren desempeño, garantizando mejores resultados en las predicciones futuras de la densidad a fallos de los módulos. Sin embargo, es importante rescatar que pocos estudios han realizado experimentación contemplando las posibilidades de combinaciones de los parámetros de los diferentes algoritmos de aprendizaje.

Es importante rescatar que el problema abierto seleccionado en la presente investigación es el relacionado a seleccionar los hiperparámetros de los diferentes algoritmos de aprendizaje, de forma tal que busque mejorar el desempeño de predicción del modelo a través del uso de la métrica AUC.

Dentro de las razones del por qué se escogió este problema abierto están: es un problema poco explorado a nivel de la literatura, se pueden analizar los diferentes resultados de predicción de densidad de defectos en los modelos tomando en cuenta las posibilidades de configuración y no solamente la "default" como se hace actualmente. Lo anterior, haciendo uso de los hiperparámetros y de los algoritmos bioinspirados que se encargarían de buscar esas posibles combinaciones que brinden mejores resultados.

La pregunta de investigación que resume el problema a resolver es: ¿Cuál es la mejor configuración de los parámetros de los algoritmos de aprendizaje que busque mejorar su desempeño, en cuanto a la densidad de defectos, según un proyecto de software determinado?

La pregunta anterior presenta otras sub-preguntas de investigación:

¿Cuáles son las combinaciones de parámetros y la mejor configuración de los algoritmos de aprendizaje que busquen mejorar la densidad de fallos de software?

¿Cuáles son los algoritmos de aprendizaje que buscan mejorar la predicción de densidad en los fallos de software, tomando como base las combinaciones de parámetros de configuración seleccionadas previamente?

¿Cuál debe ser la configuración más apropiada a nivel de algoritmos bioinspirados, para seleccionar los parámetros de los algoritmos de aprendizaje, con el fin de generar un mejor desempeño de predicción de densidad de fallos de software?

Seguidamente, se le presenta al lector la justificación de la investigación donde se establece tanto por literatura como por experimentación por qué se debe considerar importante el planteamiento del problema propuesto.

### 2.1.6 Antecedentes, estado del arte o línea base

El software ha llegado a ser más y más integral en nuestras vidas [45], siendo la densidad de fallos de software una actividad de gran importancia dentro de la industria del software. Según [45] la densidad de fallos en el software son los responsables de la pérdida de millones de millones de dólares dentro de la industria. Esta es una razón que justifica la constante investigación que se lleva a cabo mediante la creación y configuración de nuevos modelos de predicción de defectos, con el fin de determinar la densidad de fallos en los módulos de software.

Los modelos de predicción de defectos pueden ser utilizados con diferentes fines u objetivos. Se pueden emplear para mejorar la calidad del software, así como para guiar el proceso de construcción de éste. La maximización del desempeño en los modelos de predicción de defectos ha permitido ayudar a mantener la calidad del software. Esta maximización se ha logrado mediante la correcta selección de métricas, así como la creación de modelos y sus combinaciones. Sin embargo, no hay evidencia en la literatura de estudios enfocados en las configuraciones de los algoritmos de aprendizaje ya que es usual que se manejen configuraciones por defecto, sobre todo por la cantidad de combinaciones posibles que pueden tener cada una de ellas.

Un modelo de predicción de defectos que proporcione un alto desempeño de predicción permite a los profesionales a ubicar y asignar recursos de forma inteligente enfocarse más en el testing de los módulos que sean más propensos a fallos [34], [47],[48], [49],[50] y [51].

A nivel de literatura, existe un gran interés por experimentar con estrategias de programación evolutiva, específicamente con algoritmos bioinspirados, como se señala en la revisión sistemática publicada por la referente en el área, Malhotra [42]. Esta revisión propone como problema abierto: "`Existen muy pocos estudios que han evaluado la efectividad de los algoritmos genéticos en el contexto de predicción de densidad de fallos de software". En dicha revisión se especifica también: "Futuros estudios deberían enfocarse en la exactitud de la predicción de los modelos a través del uso de algoritmos genéticos" [42]. Siendo esta otra de las razones de la importancia de la investigación propuesta.

Por otra parte, a nivel experimental, se ha hecho un primer trabajo en la generación automática de esquemas de aprendizaje, utilizando como estrategia de maximización del desempeño predictivo la programación genética. Este estudio ha reportado resultados donde el uso de este tipo de metodología, ha producido una mejora de hasta el 3.2%

[36] respecto a otras metodologías utilizadas [4].

A nivel de literatura, existe un gran interés por experimentar con estrategias de programación bioinspirada como se señala en la revisión sistemática publicada por la referente en el área, Malhotra [42]. Esta revisión propone como problema abierto: "Existen muy pocos estudios que han evaluado la efectividad de los algoritmos genéticos en el contexto de predicción de fallos de software". En dicha revisión se especifica también: "Futuros estudios deberían enfocarse en la exactitud de la predicción de los modelos a través del uso de algoritmos genéticos" [42]. Siendo esta otra de las razones de la importancia de la investigación propuesta.

Por otra parte, a nivel experimental, se ha hecho un primer trabajo en la generación automática de esquemas de aprendizaje, utilizando como estrategia de maximización del desempeño predictivo la programación genética. Este estudio ha reportado resultados donde el uso de este tipo de metodología, ha producido una mejora de hasta el 3.2% [36] respecto a otras metodologías utilizadas [4].

A continuación se presentan los conceptos esenciales para el entendimiento del problema planteado en la presente investigación. Dentro de los apartados están: métricas, modelos de predicción, técnicas de minería de datos y aprendizaje automático (máquina).

1. Métricas:

La medición es el proceso por medio del cual los números o los símbolos son asignados a atributos o entidades en el mundo real. También se puede definir como un mapeo de un atributo de una entidad a un valor medido, usualmente un valor numérico. La idea de llevar a cabo el mapeo de los atributos tiene la finalidad de poder tratarlos y procesarlos de una manera formal [52].

Por su parte, una medida es el número o símbolo asignado a una entidad por la relación que caracteriza un atributo. Una de las características básicas de la medición es que se preserven las observaciones empíricas. Así por ejemplo: si se tiene un objeto A mayor a un objeto B, se debería cumplir que la medición de A debe ser mayor que la de B [52].

Cuando se utiliza una medida en un estudio empírico, se debe garantizar la validez de la medida. Una medida valida le permite a los objetos distinguirse unos de otros.

Dentro de la medición existen escalas. Seguidamente se definen las escalas más comunes [52]: Nominal

La escala nominal está caracterizada por ser la menos poderosa de los tipos de escalas. Solamente mapea los atributos de las entidades ya sea por el nombre o por el símbolo. Este mapeo puede ser visto como una clasificación de entidades de acuerdo a los atributos. Como ejemplos están: la clasificación, las etiquetas y los errores en la escritura o bien otros ejemplos más concretos pueden ser: el género de una persona, el color favorito, la religión, etc.

Ordinal

La escala ordinal ranquea las entidades después de establecer un criterio de orden. Además se puede considerar de más alcance que la escala nominal. Como ejemplos de esta escala están: mayor que, menor que, más complejo que. Algunos criterios de la definición pueden ser: grado de satisfacción (alto, medio, bajo), nivel de salud (sano, enfermo), etc. Dentro de los ejemplos de este tipo de escala se pueden mencionar: los grados y la complejidad del software.

Intervalo

La escala intervalo es utilizada cuando la escala entre dos medidas son significativas. Esta escala ordena las cosas de la misma forma que la escala ordinal, sin embargo hay una noción de distancia relativa entre dos entidades. Esta escala tiene mayor alcance que la escala ordinal. Ejemplos de esta escala son la medición de los grados en Celsius o Fahrenheit, porcentaje de retorno de una inversión, dirección medida en grados con referencia al Norte, etc.

Tasa

Se utiliza cuando hay un valor cero significativo y cuando la tasa entre dos medidas es significativa. Ejemplos de esta escala son el tamaño, la medición de la temperatura en Kelvin y la duración de una fase de desarrollo.

Las escalas de medición están relacionadas a la investigación cualitativa y cuantitativa. Se ha relacionado más a la cualitativa con las escalas nominal y ordinal y a la cualitativa con las escalas intervalo y tasa.
Métricas en ingeniería de software

La medición es crucial en el control de los proyectos de software, productos y procesos y es una parte importante de los estudios empíricos en el área [52].

Los objetos que son de interés en ingeniería de software pueden ser divididos en tres diferentes clases: proceso, producto y recursos. La primera clase (procesos), son los que describen cuáles actividades son necesarias para producir el software. La segunda clase (producto), son los artefactos, entregables o documentos, que resultan de la actividad de un proceso. Finalmente, la tercera clase (recursos), son los objetos tales como personal, hardware, o bien software necesarios para ejecutar una actividad.

En cada clase, se puede hacer una distinción entre atributos internos y externos. Un atributo interno es aquel que puede ser medido, puramente en términos del objeto. Los atributos externos pueden ser medidos con respecto a como los objetos se relacionan con otros objetos. Así por ejemplo, algunos ejemplos de medición en ingeniería de software pueden ser:

Proceso

El proceso describe cuales actividades son necesarias para producir el software. Un ejemplo de objeto es la evaluación, el tipo de atributo es interno y externo y el ejemplo de medición es el esfuerzo y el costo.

Producto

Los productos son los artefactos, los entregables o los documentos que resultan de la actividad del proceso. Un ejemplo de objeto es el código, el tipo de atributo es interno y externo y el ejemplo de medición es el tamaño y la confiabilidad.

Recurso

Son objetos tales como personal, hardware, software o bien las necesidades para la actividad de un proceso. Un ejemplo de objeto es el Personal, el tipo de atributo es interno y externo y el ejemplo de medición es la edad y la productividad.

Las escalas más habituales en ingeniería de software son las escalas nominales y las ordinales. Las validaciones de las medidas indirectas son más complejas que las medidas directas y los modelos que derivan las medidas externas tienen que ser validados.

En ingeniería de software, los análisis estadísticos dependen de las escalas. Existe un regla básica, entre más poderosas son los tipos de escalas que utilicemos más poderosas serán las herramientas y métodos estadísticos que podemos utilizar para su medición. Esto significa que al ser las escalas nominal y ordinal las más frecuentemente utilizadas en ingeniería de software, no se llega a utilizar el potencial de las herramientas y métodos estadísticos ya que estos se pueden explotar de una mejor manera con escalas más complejas tales como: razón e intervalo.

Seguidamente, se le presenta al lector una explicación de las métricas utilizadas en la investigación. Específicamente las métricas de predicción de defectos de software desde la década de los setentas.

Métricas en predicción de defectos

Existen varias métricas en el área de predicción de defectos. Sin embargo, las más utilizadas por los expertos en esta área son las de tamaño: LOC, las de flujo de control: McCabe y las de ciencias del software: Halstead.

A continuación, se presenta una explicación detallada de cada una de ellas. McCabe

Se define como métricas ciclomáticas. Interpretan un programa computacional como un conjunto de grafos fuertemente conectados. Los nodos representan parte de importancia del código fuente [3]

La noción de programa representado como un grafo ha sido utilizado por esta métrica con el objetivo de medir y controlar el conjunto de caminos que se pueden llegar a tener a través de un programa. La percepción de esta métrica es que la complejidad de un programa computacional puede ser

correlacionado con la topología de la complejidad de un grafo. Es decir que un programa computacional se puede ver tan complejo como un grafo en teoría de grafos en el campo de las ciencias de la computación [3].

McCabe propuso el número ciclomático: V(G) de teoría de grafos como un indicador de la complejidad del software. El número ciclomático es igual al número de caminos independientes a través de un programa y su representación gráfica.

Para un grafo G que representa el control o flujo de un programa, el V(G) está dado por:

E = El número de aristas de un grafo G N = El número de nodos de un grafo G

P = El número de componentes conectados en un grafo G

Según McCabe, las métricas están compuestas por los siguientes tipos: Cyclomatic complexity: Mide el número de caminos linealmente independientes. El flujo de un grafo está representado por un grafo dirigido donde cada nodo corresponde a una instrucción de un programa. Essential complexity: Se refiere a la extensión en el cual un grafo puede ser reducido descomponiendo todos los sub grafos de G. Design complexity: Se refiere a la complejidad ciclomática de un módulo reducido a un grafo.

Halstead

El modelo de Halstead es conocido como teoría de las ciencias del software. Se basa en la hipótesis de que la construcción de un programa envuelve un proceso de manipulación mental de un único operador (n1) y de un único operando (n2). Esto significa, que un programa de N1 operadores y N2 operandos es construido por la selección de n1 único operadores y n2 únicos operandos. Basado en este modelo, Halstead deriva un número de ecuaciones relacionadas a la programación tales como: nivel de programa, esfuerzo de implementación y nivel de lenguaje. [1]

Una importante e interesante característica de este modelo, es que un programa puede ser analizado por varios aspectos tales como: tamaño y esfuerzo.

El vocabulario de un programa está definido como: n= n1+n2 y el tamaño de un programa actual como: N = N1 + N2. Una de las hipótesis de esta teoría es que el tamaño de un programa bien estructurado es una función de n1 y n2 solamente. La relación es conocida como: Nh = n1 log2 n1 + n2 log2 n2

Según Halstead [53], las métricas se pueden clasificar en las medidas: base y derivadas. Las medidas base mu1 = número único de operadores, mu2 = número único de operandos, N1 = total de ocurrencias de operadores, N2 = total de ocurrencias de los operandos, tamaño = N = N1 + N2.

Lines of code (LOC)

Representa una de las más fáciles y simples métricas para calcular el tamaño de un programa computacional. Es normalmente utilizado para comparar la productividad de los programadores. La productividad, es medida como LOC/horas hombre. Cualquier línea de programa de programa excluyendo comentarios y lineas en blanco es considerada una línea de código [1].

1. Modelos de predicción

Los modelos predictivos son representaciones o abstracciones de la realidad que permiten predecir el comportamiento a futuro de un cierto fenómeno. En el caso de la investigación el fenómeno de estudio son los productos de software ya implementados.

Base de datos históricas

Las bases de datos históricas corresponden a los datos históricos donde se han analizado los diferentes proyectos de software que tienen un repositorio. Así por ejemplo en el caso de la NASA [2] existen hasta 14 repositorios, orientados a fallos de los cuales se suelen utilizar para experimentación 13 de ellos. A su vez, para el año 2011 en PROMISE existían 96 bases de datos históricas académicas orientadas a fallos, según PROMISE [56], a disposición de la comunidad académica-científica para llevar a cabo sus experimentaciones empíricas en el campo.

Es importante rescatar que este documento tiene la puntuación cuantitativa de cada uno de los módulos respecto a cada una de las métricas. El último atributo del documento corresponde a la etiqueta que determina si un módulo es o no propenso a fallos (problema de clasificación). Para representarlo, se utiliza Y/N y está ubicada como el último atributo del conjunto que se encuentran al principio del documento.

Por otra parte, los archivos son de extensión arff y el otro aspecto relevante es que por cada uno de los 13 proyectos de la NASA, existe un documento con este mismo formato y con una variación a nivel de las métricas.

## Proyectos de software

Existen varios repositorios de proyectos históricos de software, que han sido utilizados por la comunidad científica con el objetivo de experimentar y probar que tan eficaces son los modelos predictivos de fallos.

Para la presente investigación se utilizarán los dos repositorios de proyectos que frecuentemente son los más empleados en los estudios empíricos que se llevan a cabo entre la comunidad científica: Repositorio de la NASA [2] y el repositorio de PROMISE (Predictor Model in Software Engineering) [56].

## Aprendizaje

Dentro de la etapa de aprendizaje, se encuentra el entrenamiento y la evaluación. A continuación se definen cada uno de ellos con el fin de entender por qué el aprendizaje es una etapa previa de la predicción.

## Conjunto de Entrenamiento

Corresponde al conjunto de datos con el cual se entrena el modelo. Antes de poder util izar un modelo para predecir, este debe ser entrenado siguiendo algún tipo de estrategia que permita alcanzar su máximo desempeño y así crear un learner.

## Conjunto de Evaluación

Corresponde al conjunto de datos con el cual se evalúa el modelo. El modelo debe ser evaluado para llevar a cabo sus predicciones. La evaluación del modelo es de enorme importancia para la creación del predictor

## Predicción

Es la etapa posterior al entrenamiento y evaluación del modelo. En el caso de los modelos de predicción de fallos la etapa de predicción corresponde a la generación de los pronósticos binarios de si falla o no el módulo de un cierto proyecto de software.

1. Algoritmos de aprendizaje máquina

Seguidamente se presenta un resumen de las principales técnicas de aprendizaje máquina que se tomarán como base para el estudio. La categoría será representada entre paréntesis redondos [41]:

(Bayes) BayesianLogisticRegression: Es una aproximación bayesiana que utiliza un modelo de regresión logística lineal para el aprendizaje.

(Bayes) BayesNet: Redes Bayesianas para el aprendizaje.

(Bayes) ComplementNaiveBayes: Construye el complemento de un clasificador de naive bayes. (Bayes) NaiveBayes: Un clasificador probabilístico estándar de la familia naive bayes.

(Bayes) NaiveBayesMultinomial: Versión de un naive bayes multinominal. (Bayes) NaiveBayesSimple: Una implentación simple de naive bayes.

(Tree) LMT: Construye modelos logísticos de árboles.

(Tree) BFTree: Construye un árbol de decisión utilizando el algoritmo de búsqueda primero. (Tree) J48: Es un árbol de decisión basado en C4.5.

(Tree) LADTree: Construye un árbol de decisión multiclase utilizando logic boost.

(Tree) NBTree: Construye un árbol de decisión basado en un clasificador naive bayes en las hojas. (Tree) RandomForest: Construye bosques de forma aleatoria.

(Tree) RandomTree: Construye un árbol que toma en consideración un número dado de características aleatorias.

(Rules) OneR: Clasificador 1R.

(Rules) ZeroR: Predice la mayoría de las clases (si es nominal) o el valor promedio (si es numérico).

(Functions) LIBSVM: Clasificador que utiliza un Wrapper y emplea máquina de soporte vectorial. (Functions) LinearRegression: Clasificador con regresión lineal múltiple estándar.

(Functions) Logistic: Clasificador que construye modelos de regresión lineal logística.

(Functions) MultilayerPerceptation: Clasificador con red neuronal backpropagation.

(Functions) SimpleLinealRegression: Clasificador con un modelo de aprendizaje de regresión lineal basado en un atributo simple.

(Functions) SimpleLogistic: Clasificador que construye modelos de regresión logística lineal utilizando selección de atributos.

(Lazy) IB1: Clasificador que emplea vecino más cercano NNK básico. Utiliza aprendizaje basado en instancias. (Lazy) IBk: Clasificador que emplea vecino más cercano NNK.

(Lazy) LBR: Clasificador bayesiano de reglas perezoso.

(MI) MIBoost: Clasificador Boosting para datos multi-instancia. (MI) MILR: Clasificador con regresión logística multi-instancias.

(MI) MISMO: Clasificador de maquina de soporte vectorial que emplea núcleos multi-instancias.

(MI) MISVM: Aplica iterativamente un clasificador SVM de instancias simples para datos multi-instancias. (MI) SimpleMI: Aplica iterativamente un clasificador una aproximación de instancias agregadas.

(MI) MiscVFI: Método de intervalo de características simple y rápido.

**2.2 Objetivo general**

Desarrollar un framework de configuración de hiperparámetros en el contexto de la predicción de densidad de defectos mediante el uso de algoritmos bioInspirados

**2.3 Objetivos específicos y actividades**

| Objetivo específico | Descripción de actividades | Responsable | Vigencia |
|---|---|---|---|
| 1. Elaborar una revisión sistemática de literatura, con el fin de analizar las posibles combinacionesde parámetros de configuración de las técnicas de aprendizaje máquina, utilizadas en el contexto de predicción de densidad de defectosde software. (Se requiere un estudiante asistente de la escuela de informática, durante toda la ejecución de este objetivo) | 1. Plantear la pregunta de investigación | 110470816 - MURILLO MORERA JUAN DE DIOS | 17/07/2023 - 17/08/2023 |
| 1. Elaborar una revisión sistemática de literatura, con el fin de analizar las posibles combinacionesde parámetros de configuración de las técnicas de aprendizaje máquina, utilizadas en el contexto de predicción de densidad de defectosde software. (Se requiere | 2. Definir un string de búsqueda según el protocolo PICO. | 110470816 - MURILLO MORERA JUAN DE DIOS | 17/07/2023 - 17/08/2023 |

| | | | |
|---|---|---|---|
| un estudiante asistente de la escuela de informática, durante toda la ejecución de este objetivo) | | | |
| 1. Elaborar una revisión sistemática de literatura, con el fin de analizar las posibles combinacionesde parámetros de configuración de las técnicas de aprendizaje máquina, utilizadas en el contexto de predicción de densidad de defectosde software. (Se requiere un estudiante asistente de la escuela de informática, durante toda la ejecución de este objetivo) | 3. Realizar la búsqueda de la literatura tomandocomo base la pregunta del punto 1. | 110470816 - MURILLO MORERA JUAN DE DIOS | 17/07/2023 - 31/12/2023 |
| 1. Elaborar una revisión sistemática de literatura, con el fin de analizar las posibles combinacionesde parámetros de configuración de las técnicas de aprendizaje máquina, utilizadas en el contexto de predicción de densidad de defectosde software. (Se requiere un estudiante asistente de la escuela de informática, durante toda la ejecución de este objetivo) | 4. Clasificar los artículos según criterios de inclusión y exclusión propuestos. | 110470816 - MURILLO MORERA JUAN DE DIOS | 17/07/2023 - 31/12/2023 |
| 1. Elaborar una revisión sistemática de literatura, con el fin de analizar las posibles combinacionesde parámetros de configuración de las técnicas de aprendizaje máquina, utilizadas en el contexto de predicción de densidad de defectosde software. (Se requiere un estudiante asistente de la escuela de informática, durante toda la ejecución de este objetivo) | 5. Extraer la información mediante un formulariode extracción previamente definido acorde a los objetivos de la investigación. | 110470816 - MURILLO MORERA JUAN DE DIOS | 17/07/2023 - 31/12/2023 |
| 1. Elaborar una revisión sistemática de literatura, con el fin de analizar las | | | |

| | | | |
|---|---|---|---|
| posibles combinacionesde parámetros de configuración de las técnicas de aprendizaje máquina, utilizadas en el contexto de predicción de densidad de defectosde software. (Se requiere un estudiante asistente de la escuela de informática, durante toda la ejecución de este objetivo) | 6. Proceder al análisis de los artículos seleccionados teniendo siempre en consideración la relación que existen entre los estudios | 110470816 - MURILLO MORERA JUAN DE DIOS | 17/07/2023 - 31/12/2023 |
| 2. Especificar formalmente el modelode predicción de densidad de defectos de software. (Se requiere un estudiante asistente de la escuela de informática, avanzado, durante todala ejecución de este objetivo) | 1. Planteamiento de la pregunta de investigacion /hipótesis. | 110470816 - MURILLO MORERA JUAN DE DIOS | 01/01/2024 - 02/01/2024 |
| 2. Especificar formalmente el modelode predicción de densidad de defectos de software. (Se requiere un estudiante asistente de la escuela de informática, avanzado, durante todala ejecución de este objetivo) | 2. Selección del modelo apropiado. | 110470816 - MURILLO MORERA JUAN DE DIOS | 01/01/2024 - 31/07/2024 |
| 2. Especificar formalmente el modelode predicción de densidad de defectos de software. (Se requiere un estudiante asistente de la escuela de informática, avanzado, durante todala ejecución de este objetivo) | 3. Formulación del modelo. | 110470816 - MURILLO MORERA JUAN DE DIOS | 01/01/2024 - 31/07/2024 |
| 2. Especificar formalmente el modelo de predicción de densidad de defectos de software. (Se requiere un estudiante asistente de la escuela de informática, avanzado, durante toda la ejecución de este objetivo) | 4. Compartir los resultados de la revisión de literatura y el modelo con la comunidad estudiantil mediante foros de discusión. | POR DEFINIR - ESTUDIANTE | 31/07/2024 - 30/09/2024 |

| | | | |
|---|---|---|---|
| 3. Diseñar e implementar un framework que mediante el uso de algoritmos bioinspirados, lleve a cabo la selección de forma automática de los parámetros de configuración de las diferentes técnicas de aprendizaje máquina, buscando mejorar el desempeño de los modelos empleados en | 1. Análisis del sistema. | 110470816 - MURILLO MORERA JUAN DE DIOS | 01/10/2024 - 31/12/2024 |

| | | | |
|---|---|---|---|
| predicción de densidad de defectos de software. (Se requiere un estudiante asistente de la escuela de informática, avanzado, durante toda la ejecución de este objetivo) | | | |
| 3. Diseñar e implementar un framework que mediante el uso de algoritmos bioinspirados, lleve a cabo la selección de forma automática de los parámetros de configuración de las diferentes técnicas de aprendizaje máquina, buscando mejorar el desempeño de los modelos empleados en predicción de densidad de defectos de software. (Se requiere un estudiante asistente de la escuela de informática, avanzado, durante toda la ejecución de este objetivo) | 2. Análisis de requisitos. | 110470816 - MURILLO MORERA JUAN DE DIOS | 01/10/2024 - 31/12/2024 |
| 3. Diseñar e implementar un framework que mediante el uso de algoritmos bioinspirados, lleve a cabo la selección de forma automática de los parámetros de configuración de las diferentes técnicas de aprendizaje máquina, buscando mejorar el desempeño de los modelos empleados en predicción de densidad de defectos de software. (Se requiere un estudiante asistente de la escuela de informática, avanzado, durante toda la ejecución de este objetivo) | 3. Diseño | 110470816 - MURILLO MORERA JUAN DE DIOS | 01/01/2025 - 01/07/2025 |
| 3. Diseñar e implementar un framework que mediante el uso de algoritmos bioinspirados, lleve a cabo la selección de forma automática de | | | |

| | | | |
|---|---|---|---|
| los parámetros de configuración de las diferentes técnicas de aprendizaje máquina, buscando mejorar el desempeño de los modelos empleados en predicción de densidad de defectos de software. (Se requiere un estudiante asistente de la escuela de informática, avanzado, durante toda la ejecución de este objetivo) | 4. Codificación. | 110470816 - MURILLO MORERA JUAN DE DIOS | 01/01/2025 - 31/07/2025 |
| 3. Diseñar e implementar un framework que mediante el uso de algoritmos bioinspirados, lleve a cabo la selección de forma automática de los parámetros de configuración de las diferentes técnicas de aprendizaje máquina, buscando mejorar el desempeño de los modelos empleados en predicción de densidad de defectos de software. (Se requiere un estudiante asistente de la escuela de informática, avanzado, durante toda la ejecución de este objetivo) | 5. Pruebas. | 110470816 - MURILLO MORERA JUAN DE DIOS | 01/08/2025 - 01/09/2025 |
| 4. Evaluar mediante casos empíricos el desempeño obtenido porel framework planteado en el objetivo 3. (Se requiere un estudiante asistente de la escuela de informática, avanzado, durante toda la ejecución de este objetivo) | 1. Definir el alcance del experimento. | 110470816 - MURILLO MORERA JUAN DE DIOS | 01/10/2025 - 31/12/2025 |
| 4. Evaluar mediante casos empíricos el desempeño obtenido porel framework planteado en el objetivo 3. (Se requiere un estudiante asistente de la escuela de informática, avanzado, durante toda la ejecución de este objetivo) | 2. Definir la planificación del experimento. | 110470816 - MURILLO MORERA JUAN DE DIOS | 01/10/2025 - 31/12/2025 |
| | | | |

| | | | |
|---|---|---|---|
| 4. Evaluar mediante casos empíricos el desempeño obtenido porel framework planteado en el objetivo 3. (Se requiere un estudiante asistente de la escuela de informática, avanzado, durante toda la ejecución de este objetivo) | 3. Definir la ejecución del experimento. | 110470816 - MURILLO MORERA JUAN DE DIOS | 01/01/2026 - 01/02/2026 |
| 4. Evaluar mediante casos empíricos el desempeño obtenido porel framework planteado en el objetivo 3. (Se requiere un estudiante asistente de la escuela de informática, avanzado, durante toda la ejecución de este objetivo) | 4. Establecer las estrategias de análisis del experimento. | 110470816 - MURILLO MORERA JUAN DE DIOS | 01/01/2026 - 01/02/2026 |
| 4. Evaluar mediante casos empíricos el desempeño obtenido porel framework planteado en el objetivo 3. (Se requiere un estudiante asistente de la escuela de informática, avanzado, durante toda la ejecución de este objetivo) | 5. Ejecución de los experimentos. | 110470816 - MURILLO MORERA JUAN DE DIOS | 01/02/2026 - 01/05/2026 |
| 4. Evaluar mediante casos empíricos el desempeño obtenido por el framework planteado en el objetivo 3. (Se requiere un estudiante asistente de la escuela de informática, avanzado, durante toda la ejecución de este objetivo) | 6. Realizar una charla virtual acerca de los resultados obtenidos por el framework desarrollado. | POR DEFINIR - ESTUDIANTE | 01/06/2026 - 17/07/2026 |
| 5. Validar y transferir resultados a la industria y a la academia. | 1. Visitar empresas de desarrollo de software, mostrando los resultados del framework, con el fin de considerar sus datos | 110470816 - MURILLO MORERA JUAN DE DIOS | 01/06/2026 - 17/07/2026 |
| 5. Validar y transferir resultados a la industria y a la academia. | 2. Realizar un seminario para transmitir a la comunidad universitaria los resultados del proyecto de investigación. | POR DEFINIR - ESTUDIANTE | 01/06/2026 - 17/07/2026 |
| 5. Validar y transferir resultados a la industria y a la academia. | 3. Elaborar un descriptor de curso de Inteligencia Artificial aplicada a Ingeniería de Software. | POR DEFINIR - ESTUDIANTE | 01/06/2026 - 17/07/2026 |

**2.4 Objetivos específicos e indicadores**

| Objetivo específico | Descripción de indicador (es) de logro por objetivo | Fecha prevista de entrega |
|---|---|---|
| 1. Elaborar una revisión sistemática de literatura, con el fin de analizar las posibles combinaciones de parámetros de configuración de las técnicas de aprendizaje máquina, utilizadas en el contexto de predicción de densidad de defectos de software. (Se requiere un estudiante asistente de la escuela de informática, durante toda la ejecución de este objetivo) | Al menos un artículo enviado de la revisión sistemática de literatura a alguna conferencia de prestigio.<br><br>Al menos un foro de discusión para compartir los resultados de la revisión de literatura | 30/03/2024 |
| 2. Especificar formalmente el modelo de predicción de densidad de defectos de software. (Se | | |

| | | |
|---|---|---|
| requiere un estudiante asistente de la escuela de informática, avanzado, durante toda laejecución de este objetivo) | Al menos un artículo enviado con la especificación formal del modelo.<br><br>Al menos un foro de discusión para compartir los resultados del modelo formal | 30/09/2024 |
| 3. Diseñar e implementar un framework que mediante el uso de algoritmos bioinspirados, lleve a cabo la selección de forma automática de los parámetros de configuración de las diferentes técnicas de aprendizaje máquina, buscando mejorar el desempeño de los modelos empleados en predicción de densidad de defectos de software. (Se requiere un estudiante asistente de la escuela de informática, avanzado, durante toda la ejecución de este objetivo) | La herramienta funcional. | 01/10/2025 |
| 4. Evaluar mediante casos empíricos el desempeño obtenido por el framework planteado en el objetivo 3. (Se requiere un estudiante asistente de la escuela de informática, avanzado, durante toda la ejecución de este objetivo) | Un artículo enviado con los resultados de la ejecución de los casos empíricos, donde se evalúe el desempeño obtenido por el software planteado en el objetivo 3.<br><br>Al menos una charla virtual acerca de los resultados del framework propuesto | 01/06/2026 |
| 5. Validar y transferir resultados a la industria | Un reporte técnico con la información de al menos dos empresas de desarrollo de software, mostrando los resultados del framework considerando sus datos.<br><br>Al menos un seminario de divulgación de los resultados de la investigación a la comunidad universitaria.<br><br>Al menos un descriptor del curso Inteligencia Artificial aplicada a Ingeniería de Software. | 01/06/2026 |

**2.5 Productos esperados**

| Tipo de producto | Descripción | Cantidad | Fecha prevista de entrega |
|---|---|---|---|
| ARTÍCULO DE INVESTIGACIÓN | Un artículo de la revisión sistemática de literatura. | 1 | 30/03/2024 |

| | | | |
|---|---|---|---|
| ARTÍCULO DE INVESTIGACIÓN | Un artículo con la especificación formal del modelo. | 1 | 30/09/2024 |
| SOFTWARE | La herramienta funcional. | 1 | 01/10/2025 |
| ARTÍCULO DE INVESTIGACIÓN | Un artículo con los resultados de la ejecución de los casos empíricos, donde se evalúe el desempeño obtenido por el software planteado en el objetivo 3 | 1 | 01/06/2026 |
| ENTREGABLE DEL | Elaboración de un descriptor de curso de | 1 | 17/07/2026 |

| | | | |
|---|---|---|---|
| PROYECTO | Inteligencia Artificial aplicada a Ingeniería de Software. | | |
| REPORTE TÉCNICO | Un reporte técnico con los resultados obtenidos en la industria | 1 | 17/07/2026 |
| FOROS, CHARLAS VIRTUALES y FOROS DE DIVULGACION | Se harán foros y charlas vituales para divulgar los resultados de la investigación a la comunidad universitaria | | 17/07/2023-17/07/2026 |

**2.6 Metodología**

**2.6.1 Descripción**

En este apartado, se describen las diferentes fases que se seguirán para lograr el alcance de cada uno de los objetivos planteados, con el fin de proponer el modelo de predicción de defectos de software que maximice el desempeño a través de la combinación automática de parámetros de configuración de las técnicas de aprendizaje máquina mediante el uso de programación evolutiva.

Primera fase

La primera fase está ligada con el primer objetivo específico: Analizar las posibles combinaciones de parámetros de configuración de las técnicas de aprendizaje máquina utilizados en el contexto de predicción de defectos de software.

Para alcanzar este objetivo, se llevará a cabo un sumario (cuadro resumen) de los diferentes algoritmos de aprendizaje máquina utilizados en el contexto de predicción de fallos de software. Para la caracterización de estos algoritmos de aprendizaje y sus respectivos parámetros de configuración se propone la realización de una revisión sistemática de literatura, que permita de una forma estructurada y precisa mapear los diferentes algoritmos de aprendizaje máquina, así como la configuración de los parámetros utilizada. Para lograr esto será necesario la realización de una o varias preguntas de investigación que permitan identificar a nivel de la literatura aquellos estudios relacionados a nuestro objeto de estudio. Dentro de las actividades de esta primera fase estarán [36] y [58]:

1. Plantear la pregunta de investigación

2. Definir un string de búsqueda según el protocolo PICO.

3. Realizar la búsqueda de la literatura tomando como base la pregunta del punto 1.

4. Clasificar los artículos según criterios de inclusión y exclusión propuestos.

5. Extraer la información mediante un formulario de extracción previamente definido acorde a los objetivos de la investigación.

6. Proceder al análisis de los artículos seleccionados teniendo siempre en consideración la relación que existen entre los estudios

Segunda fase

La segunda fase toma como base la primera con el fin de plantear un modelo tanto a nivel formal como algorítmico. Esta fase está ligada al segundo objetivo específico. Para el cumplimiento de este objetivo se seguirá la metodología de los cinco pasos planteada en [63]:

1. Planteamiento de la pregunta.

2. Selección del modelo apropiado.

3. Formulación del modelo.

4. Solución del modelo.

5. Respuesta a la pregunta planteada en 2.1

Tercera fase

La tercera fase corresponde al tercer objetivo específico: Diseñar e implementar una herramienta que mediante el uso de la programación evolutiva seleccione de forma automática la combinación de parámetros de configuración de las técnicas de aprendizaje máquina obteniendo el mejor desempeño.

Para el cumplimiento de este objetivo se seguirá un modelo de desarrollo de software conocido como Modelo en cascada [52]. Dentro de las actividades de esta fase estarán:

1. Análisis del sistema.

2. Análisis de requisitos.

3. Diseño

4. Codificación.

5. Pruebas.

Las etapas de la 1-5 se repetirán hasta alcanzar la estabilidad y resultados esperados en cuanto a la maximización del desempeño de predicción.

Cuarta fase

La cuarta fase está relacionada al cuarto objetivo específico: Evaluar mediante una estrategia empírica el desempeño obtenido por el software propuesto. Para el cumplimiento de esta fase se llevarán a cabo las actividades propias de un experimento según [52] y [62]:

1. Definir el alcance del experimento.

2. Definir la planificación del experimento.

3. Definir la ejecución del experimento.

4. Establecer las estrategias de análisis del experimento.

Las fases metodológicas conjuntamente con sus actividades descritas anteriormente se repiten de la fase 2 a la fase 4 hasta poder optimizar los resultados finales del modelo planteado y de esta manera cumplir con el objetivo general de procurar mejorar el desempeño de predicción del modelo planteado.

### 2.6.2 Plan de Gestión de Datos

Dentro del plan de gestión de datos se considerarán:

1. Los Data Sets de la NASA-MDP y PROMISE, así como otros repositorios.
2. Los Data Sets propios que vayan a crear los investigadores.

## 2.7 Mecanismos de rendición de cuentas y estrategia de comunicación

En este caso la rendición de cuentas se haría cotejando cada indicador con su respectiva actividad y objetivo.

## 2.8 Matriz de riesgos

| Descripción del riesgo | Medida de mitigación |
|---|---|
| Categoría Financiero. Que no haya suficiente presupuesto institucional para boletos, estadía y pago de conferencias y/o revistas derivadas del proyecto de investigación. | Como plan de contigencia sería buscar elpresupuesto en otras instancias externas tales como: MICITT o CONICIT. |
| Categoría Gestión. Que al socializar el marco de trabajo con el sector industria, no haya una apertura por parte de las empresas, en brindarnos no solo los permisos de ingreso, sino de recolección de errores de sus proyectos con el fin de realizar los experimentos pertinentes. | Como plan de contigencia, tendríamos contar con al menos unas tres opciones de organizaciones paraasí tener más opciones de probar el producto propuesto. |
| | |

| | |
|---|---|
| Categoría Gestión. Que no se asigne de forma correcta en tiempo y forma la participación de los estudiantes. | Como plan de contigencia a este riesgo, sería gestionar con anticipación la participación de los estudiantes, rigiéndonos por los tiempos de los objetivos y del cumplimiento de estos. |
| Categoría Gestión. Que no se pueda brindar el tiempo (jornada) solicitado, por parte de la escuela al investigador principal. | Como plan de contigencia, se tiene ver la opción de reeplantear objetivos y/o actividades en miras de la reducción del proyecto y del tiempo de ejecución del mismo. |
| Categoría Investigación. Que los objetivos planteados, no sean cumplidos en los tiempos establecidos y que esto implique solicitar más tiempo (extensión del proyecto). | Como plan de contigencia en este riesgo, sería ir cumpliendo, cada actividad según el cronograma planteado, sin salirse de tiempo de cada actividad. |
| Categoría Investigación. Que no se tengan los productos esperados en el tiempo establecido (siempre relacionado al punto anterior). | Como plan de contigencia en este riesgo es tener claro el alcance de cada entregable, y de ser necesario reducirlo, sin afectar el producto final,con el fin de cumplir con el tiempo establecido. |

## 3 Presupuesto

No se ha definido el presupuesto.

### 3.1 Justificación

No se ha descrito una justificación para el proyecto.

## 4 Información para indicadores y afines

### 4.1 Palabras clave

- ALGORITMOS
- Actividad científica
- MODELOS

### 4.2 Grupos meta

| Primer nivel | Último nivel |
|---|---|
| SEGÚN ORGANIZACIÓN | TICS |

### 4.3 Sector de aplicación u objetivo socioeconómico

Investigación no orientada

### 4.4 Áreas de la ciencia

| Área | Subárea |
|---|---|
| Ingeniería y tecnología | Ingeniería eléctrica, electrónica y de la información |

### 4.5 Área de influencia del proyecto

| Región | País | Región cioeconómica | Provincia | Cantón | Distrito | Localidad |
|---|---|---|---|---|---|---|
| Nacional | COSTA RICA | Región Central | SAN JOSÉ | SAN JOSÉ | CATEDRAL | Zona central de San José |

| Latitud | Longitud |
|---|---|
| 0.0 | 0.0 |

### 5 Observaciones

1. Se va a trabajar en conjunto con la Facultad de Ingeniería de Software e Inteligencia Artificial de la Universidad Complutense de Madrid, España, específicamente con el Dr. Rubén Fernandez- Fuentes como investigador invitado.
2. Por cada objetivo del proyecto se tendrá participación estudiantil. No se discriminará la participación de estudiantes de diferentes géneros, etnias, orientaciones sexuales, discapacidades u otras características que puedan dar lugar a perspectivas diversas y representativas en el proyecto.
3. Se busca fomentar la inclusión y diversidad en la selección y participación del estudiante asistente, promoviendo la igualdad de oportunidades y evitando cualquier forma de discriminación.
4. Los logros y avances del proyecto pueden ser utilizados en futuras investigaciones, aplicaciones prácticas o proyectos relacionados. Los resultados podrían tener un impacto significativo en la industria del software, promoviendo su aceptación y escalabilidad. Al hacerlo, se fortalece la viabilidad y sostenibilidad del proyecto, asegurando que sus resultados trasciendan más allá de su duración y generen un impacto perdurante.

### 6 Bibliografía

# Bibliografía:

[1] G. Singh, D. Singh, and V. Singh, A study of software metrics. IJCEM International Journal of Computational Engineering & Management, vol. 11, pp. 22-27, 2011.

[2] M. Shepperd, Q. Song, Z. Sun, and C. Mair, Data quality: Some comments on the nasa software defect datasets? Software Engineering, IEEE Transactions on, vol. 39, no. 9, pp. 1208-1215, 2013.

[3] T. McCabe, A complexity measure. IEEE Transactions on Software Engineering, vol. 2, no. 4, pp. 308-320, December 1976.

[4] Q. Song, Z. Jia, M. Shepperd, S. Ying, and J. Liu. A general software defect-proneness prediction framework. Software Engineering, IEEE Transactions on, vol. 37, no. 3, pp. 356? 370- 2011.

[5] J. Murillo and M. Jenkins. A software defect-proneness prediction framework: A new approach using genetic algorithms to generate learning schemes. In 27th International Conference on Software Engineering and Knowledge Engineering. SEKE, 2015, pp. 60-66.

[6] M. Host and P. Runeson. Checklists for software engineering case study research. In Empirical Software Engineering and Measurement, 2007. ESEM 2007. First International Symposium on, Sept 2007, pp. 479-481.

[7] R. Malhotra. Comparative analysis of statistical and machine learning methods for predicting faulty modules. Applied Soft Computing, vol. 21, pp. 286-297, 2014.

[8] Y. Jiang, B. Cukic, T. Menzies, and J. Lin. Incremental development of fault prediction models. International Journal of Software Engineering and Knowledge Engineering, vol. 23, no. 10, pp. 1399-1425, 2013.

[9] H. Lu and B. Cukic. An adaptive approach with active learning in software fault prediction. In Proceedings of the 8th International Conference on Predictive Models in Software Engineering, ser. PROMISE ?12. New York, NY, USA: ACM, 2012, pp. 79-88. [Online]. Available: http://doi.acm.org/10.1145/2365324.2365335

[10] P. Singh and S. Verma. Empirical investigation of fault prediction capability of object oriented metrics of open

source software. In Computer Science and Software Engineering (JCSSE), 2012 International Joint Conference on. IEEE, 2012, pp. 323-327.

[11] A. B. de Carvalho, A. Pozo, and S. R. Vergilio. A symbolic fault- prediction model based on multiobjective particle swarm optimization,? J. Syst. Softw., vol. 83, no. 5, pp. 868?882, May 2010. [Online]. Available: http://dx.doi.org/10.1016/j.jss.2009.12-23

[12] N. Seliya, T. Khoshgoftaar, and J. Van Hulse. Predicting faults in high assurance software. in High-Assurance Systems Engineering (HASE), 2010 IEEE 12th International Symposium on, Nov 2010, pp. 26-34.

[13] I. Gondra. Applying machine learning to software fault-proneness prediction. Journal of Systems and Software, vol. 81, no. 2, pp. 186-195, 2008.

[14] Y. Jiang, B. Cuki, T. Menzies, and N. Bartlow. Comparing design and code metrics for software quality prediction. In Proceedings of the 4th international workshop on Predictor models in software engineering. ACM, 2008, pp. 11-18.

[15] K. Gao and T. Khoshgoftaar. A comprehensive empirical study of count models for software fault prediction. Reliability, IEEE Transactions on, vol. 56, no. 2, pp. 223?236, June 2007.

[16] T. Gyimothy, R. Ferenc, and I. Siket? Empirical validation of object-oriented metrics on open source software for fault prediction. Software Engineering, IEEE Transactions on, vol. 31, no. 10, pp. 897-910, Oct 2005.

[17] T. Hall, S. Beecham, D. Bowes, D. Gray, and S. Counsell. A systematic literature review on fault prediction performance in software engineering Software Engineering, IEEE Transactions on, vol. 38, no. 6, pp. 1276?1304, Nov 2012.

[18] E. Arisholm, L. C. Briand, and E. B. Johannessen. A systematic and comprehensive investigation of methods to build and evaluate fault prediction models. Journal of Systems and Software, vol. 83, no. 1, pp. 2-17, 2010.

[19] B. T. Compton and C. Withrow. Prediction and control of ada software defects. Journal of Systems and Software, vol. 12, no. 3, pp. 199-207, 1990.

[20] J. Munson and T. Khoshgoftaar. Regression modelling of software quality: empirical investigation. Information and Software Technology, vol. 32, no. 2, pp. 106-114, 1990.

[21] N. B. Ebrahimi. On the statistical analysis of the number of errors remaining in a software design document after inspection. Software Engineering, IEEE Transactions on, vol. 23, no. 8, pp. 529-532, 1997.

[22] S. A. Vander Wiel and L. G. Votta. Assessing software designs using capture-recapture methods. Software Engineering, IEEE Transactions on, vol. 19, no. 11, pp. 1045-1054, 1993.

[23] P. Runeson and C. Wohlin. An experimental evaluation of an experience-based capture-recapturemethod in software code inspections. Empirical Software. Engineering, vol. 3, no. 4, pp. 381-406, Dec. 1998. [Online]. Available: http://dx.doi.org/10.1023/A:1009728205264

[24] K. El Emam and O. Laitenberger. Evaluating capture-recapture models with two inspectors, Software Engineering, IEEE Transactions on, vol. 27, no. 9, pp. 851-864, Sep 2001.

[25] Q. Song, M. Shepperd, M. Cartwright, and C. Mair. Software defect association mining and defect correction effort prediction. Software Engineering, IEEE Transactions on, vol. 32, no. 2, pp. 69-82, Feb 2006.

[26] H. Wang, T. M. Khoshgoftaar, and A. Napolitano. Software measurement data reduction using ensemble techniques. Neurocomputing, vol. 92, pp. 124-132, 2012.

[27] V. R. Basili, L. C. Briand, and W. L. Melo. A validation of object- oriented design metrics as quality indicators. Software Engineering, IEEE Transactions on, vol. 22, no. 10, pp. 751-761, 1996.

[28] T. M. Khoshgoftaar, E. B. Allen, J. P. Hudepohl, and S. J. Aud. Application of neural networks to software quality modeling of a very large telecommunications system. Neural Networks, IEEE Transactions on, vol. 8, no. 4, pp. 902-909, 1997.

[29] T. M. Khoshgoftaar, E. B. Allen, W. D. Jones, and J. P. Hudepohl. Classification tree models of software quality over multiple releases. In Software Reliability Engineering, 1999. Proceedings. 10th International Symposium on. IEEE, 1999, pp. 116-125.

[30] Y. Jiang, J. Lin, B. Cukic, and T. Menzies. Variance analysis in software fault prediction models. In Software Reliability Engineering, 2009. ISSRE-09. 20th International Symposium on. IEEE, 2009, pp. 99-108.

[31] N. Gayatri, S. Nickolas, A. Reddy, and R. Chitra. Performance analysis of datamining algorithms for software quality prediction. In Advances in Recent Technologies in Communication and Computing, 2009. ART- Com-09. International Conference on. IEEE, 2009, pp. 393-395.

[32] C. Catal and B. Diri. Unlabelled extra data do not always mean extra performance for semi-supervised fault prediction. Expert Systems, vol. 26, no. 5, pp. 458-471, 2009.

[33] R. Malhotra, A. Kaur, and Y. Singh. Empirical validation of object- oriented metrics for predicting fault proneness at different severity levels using support vector machines. International Journal of System Assu- rance Engineering and Management, vol. 1, no. 3, pp. 269-281, 2010.

[34] R. Moser, W. Pedrycz, and G. Succi. A comparative analysis of the efficiency of change metrics and static code attributes for defect prediction. In Proceedings of the 30th international conference on Software engineering, ser. ICSE-08. New York, NY, USA: ACM, 2008, pp. 181-190. [Online]. Available: http://www.siduna.una.ac.cr: 2257/10.1145/1368088.1368114

[35] Y. Singh, A. Kaur, and R. Malhotra. Empirical validation of object-oriented metrics for predicting fault proneness models. Software quality journal, vol. 18, no. 1, pp. 3-35, 2010.

[36] J. Murillo, C. Quesada, and M. Jenkins. Software fault prediction: A systematic mapping study. In XVIII Ibero-American Conference on Software Engineering. CibSE, 2015, pp. 50-64.

[37] T. Menzies, J. Greenwald, and A. Frank. Data mining static code at- tributes to learn defect predictors. Software Engineering, IEEE Transactions on, vol. 33, no. 1, pp. 2-13, Jan 2007.

[38] S. H. Kan, Metrics and Models in Softwa re Quality Engineering, 2nd ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2002.

[39] T. McCabe. A complexity measure. IEEE Transactions on Software Engineering, vol. 2, no. 4, pp. 308-320, December 1976.

[40] M. Halstead, Elements of Software Science. Elsevier, 1977.

[41] I. H. Witten and E. Frank, Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann, 2005.

[42] R. Malhotra. A systematic review of machine learning techniques for software fault prediction. Applied Soft Computing, vol. 27, pp. 504-518, 2015.

[43] K. Gao, T. M. Khoshgoftaar, and H. Wang. An empirical investigation of filter attribute selection techniques for software quality classification. In Proceedings of the 10th IEEE international conference on Information Reuse & Integration, ser. IRI?09. Piscataway, NJ, USA: IEEE Press, 2009, pp. 272?277. [Online]. Available: http://dl.acm.org/citation.cfm?id=1689250.1689300

[44] D. Whitley. A genetic algorithm tutorial. Statistics and Computing, vol. 4, pp. 65-85, 1994.

[45] A. Schroter. Failure preventing recommendations. In Software Engineering, 2010 ACM/IEEE 32nd International Conference on, vol. 2, 2010, pp. 397-400.

[46] D. Lo, H. Cheng, J. Han, S.-C. Khoo, and C. Sun. Classification of software behaviors for failure detection: a discriminative pattern mining approach. In Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, ser. KDD-09. New York, NY, USA: ACM, 2009, pp. 557?566. [Online]. Available: http://doi.acm.org/10.1145/1557019.1557083

[47] A. B. De Carvalho, A. Pozo, and S. R. Vergilio. A symbolic fault-prediction model based on multiobjective particle swarm optimization, Journal of Systems and Software, vol. 83, no. 5, pp. 868-882, 2010.

[48] T. M. Khoshgoftaar, K. Gao, and N. Seliya. Attribute selection and imbalanced data: Problems in software defect prediction. In Tools with Artificial Intelligence (ICTAI), 2010 22nd IEEE International Conference on, vol. 1. IEEE, 2010, pp. 137-144.

[49] A. E. Hassan and T. Xie. Mining software engineering data. In

Proc. 34th International Conference on Software Engineering (ICSE 2012), Tutorial, June 2012. [Online]. Available: http://www.csc.ncsu. edu/faculty/xie/publications.htm

[50] A. E. Hassan. Predicting faults using the complexity of code changes. In Proceedings of the 31st International Conference on Software Engineering, ser. ICSE ?09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 78-88. [Online]. Available: http://dx.doi.org/10.1109/ICSE.2009.5070510

[51] A. E. Hassan and T. Xie. Software intelligence: the future of mining software engineering data. In Proceedings of the FSE/SDP workshop on Future of software engineering research, ser. FoSER 10. New York, NY, USA: ACM, 2010, pp. 161-166. [Online]. Available: http://doi.acm.org/10.1145/1882362.1882397

[52] C. Wohlin, P. Runeson, M. Host, M. C. Ohlsson, B. Regnell, and A. Wesslen, Experimentation in software engineering. Springer Science & Business Media, 2012.

[53] M. Halstead, Elements of Software Science. Elsevier, 1977.

[54] F. Berzal and N. Mat. Data mining: Concepts and techniques by jiawei han and micheline kamber? SIGMOD Rec., vol. 31, no. 2, pp. 66?68, Jun. 2002. [Online]. Available: http://doi.acm.org/10.1145/565117.565130

[55] S. Beniwal and J. Arora. Classification and feature selection techniques in data mining. International Journal of Engineering Research and Technology (IJERT), vol. 1, no. 6, 2012.

[56] T. Menzies, B. Caglayan, Z. He, E. Kocaguneli, J. Krall, F. Peters, and B. Turhan. (2012, June) The promise repository of empirical software engineering data. [Online]. Available: http://promisedata. googlecode. com

[57] L. Araujo and C. Cervigon, Algoritmos evolutivos: un enfoque practico. Alfaomega, 2009.

[58] B. Kitchenham and S. Charters, Guidelines for performing systematic literature reviews in software engineering. 2007.

[59] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsso n. Systematic mapping studies in software engineering. In 12th International Conference on Evaluation and Assessment in Software Engineering, vol. 17, 2008, p. 1.

[60] B. Kitchenham, O. P. Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, Systematic literature reviews in software engineering a systematic literature review. Information and software technology, vol. 51, no. 1, pp. 7-15, 2009.

[61] J. Biolchini, P. G. Mian, A. C. C. Natali, and G. H. Travassos. Systematic review in software engineering. System Engineering and Compu- ter Science Department COPPE/UFRJ, Technical Report ES, vol. 679, no. 05, p. 45, 2005.

[62] D. C. Montgomery, Design and analysis of experiments. John Wiley & Sons, 2010.

Meerschaert, Mark M. Mathematical modeling. Academic press, 2013.