UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 2

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

Herramienta para la creación de datos sintéticos en problemas de predicción con salidas múltiples integrado en MULAN

AUTOR

Ronald Rodríguez Reyes

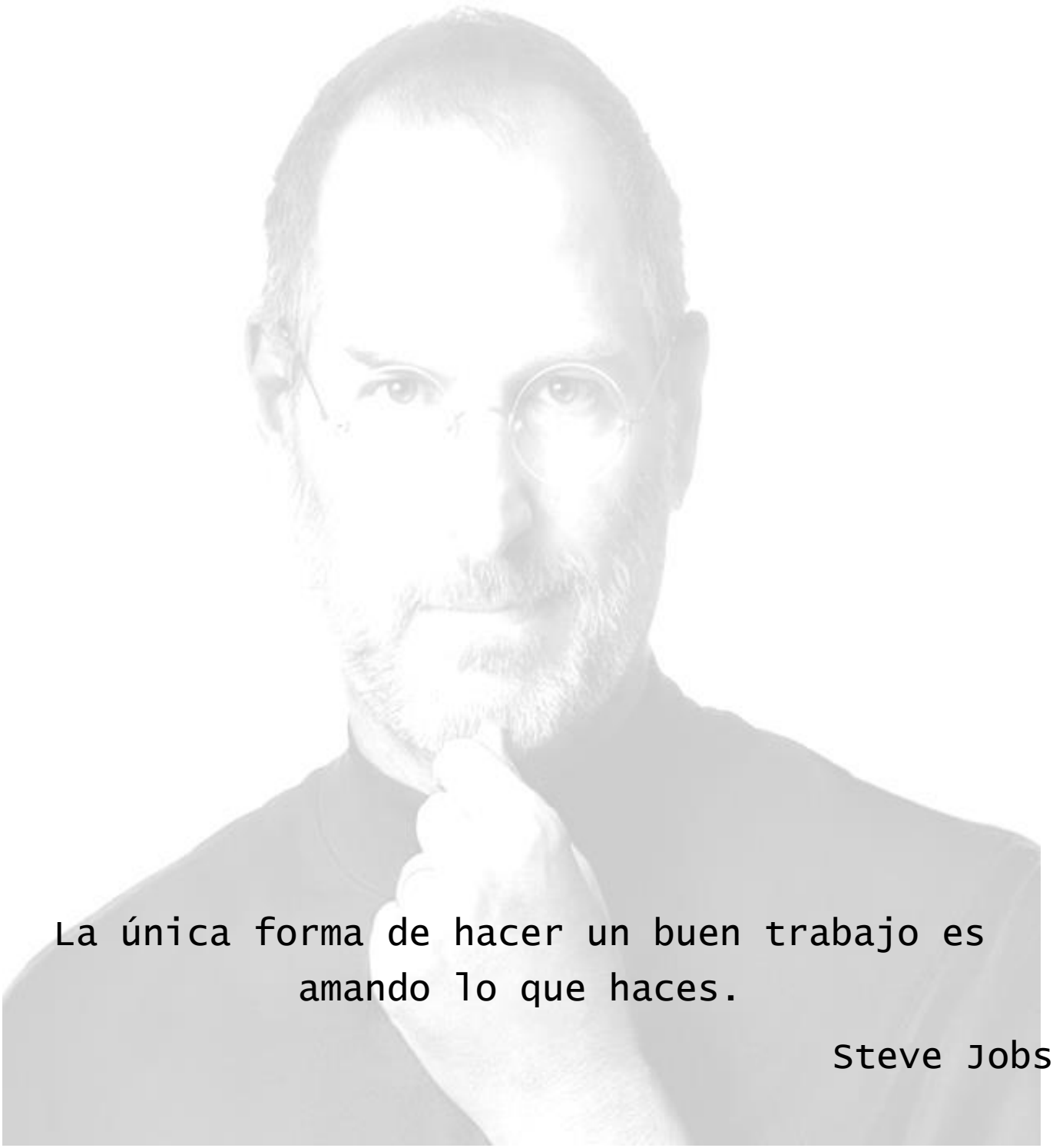
TUTOR

Ms. C. Héctor R. González Diez

La Habana, junio de 2018

"Año 60 de la Revolución

Pensamiento



La única forma de hacer un buen trabajo es amando lo que haces.

Steve Jobs

DECLARACIÓN DE AUTORÍA

Declaro ser el único autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

Ronald Rodríguez Reyes
(Autor)

Ms. C. Héctor R. González Diez
(Tutor)

DATOS DE CONTACTO

Ms. C. Héctor R. González Diez (hglez@uci.cu):

Ronald Rodríguez Reyes (rrreyes@estudiantes.uci.cu):

AGRADECIMIENTOS

Les agradezco a mis compañeros de apartamento por la amistad que hemos forjado a lo largo de estos cinco años. Al piquete de la aceituna por cada momento que compartimos y pasamos súper bien juntos. A mis amigos del voleibol, Todos ustedes los considero como los hermanos y hermanas que no tuve. Agradecerles además a todas esas personas que siempre estuvieron al tanto de mí. A Anita por tu apoyo en estos 5 años y por todo lo que has hecho por mi mamá. Y por último y no menos importantes a mi familia que han inculcado buenos valores en mí, que me apoyaron y siempre estuvieron ahí. Los quiero

DEDICATORIA

Le dedico esta tesis a mi familia y en especial a mis padres por siempre apoyarme, por educarme como lo hicieron, gracias a ellos soy una mejor persona.

RESUMEN

En muchas aplicaciones prácticas del aprendizaje supervisado, la tarea implica la predicción de múltiples variables objetivo a partir de un conjunto común de variables de entrada. Cuando los objetivos de predicción son binarios, la tarea se denomina Multi-Label Classification, mientras que cuando los objetivos son continuos, la tarea se denomina Multi-Target Regression. En la actualidad solo existen 18 conjuntos de datos destinados a esta última tarea, todos muy similares, en su mayoría con pocos datos, y siempre con menos atributos que instancias. Para contrarrestar esta problemática se desarrolla una herramienta informática para la creación de datos sintéticos en problemas de predicción con salidas múltiples integrado en MULAN. Para ello se utilizó como IDE Netbeans v8.2, como lenguaje de programación JAVA 8. Además, se realizaron las pruebas de Friedman y Bonferroni-Dunn utilizando como medida el Average Relative RMSE.

Los resultados de una evaluación experimental llevada a cabo en una diversa colección de conjuntos de datos muestran que existen diferencias significativas entre los algoritmos que resuelven problemas de tipo Multi-Target Regression cuando los conjuntos de datos tienen más atributos que instancias.

Palabras clave: Multi-Target Regression, conjuntos de datos, datos sintéticos

Índice

ÍNDICE DE FIGURAS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

En la actualidad, el desarrollo de las tecnologías exige nuevos conocimientos por parte de quienes requieren su uso y aplicación, sobretodo, para dirigir una organización. El manejo eficiente de los sistemas de información en las organizaciones da soporte a las operaciones empresariales, la gestión empresarial y la toma de decisiones; en este último se analiza toda la información existente. “El análisis de información forma parte del proceso de adquisición y apropiación de los conocimientos latentes acumulados en distintas fuentes de información. El análisis busca identificar la información “útil", es decir, aquella que interesa al usuario, a partir de una gran cantidad de datos” (Sarduy, 2007).

Existen varias técnicas para realizar el análisis de la información, una de ellas es la inteligencia artificial, la cual para convertir los datos en conocimiento utiliza el proceso de extracción del conocimiento (KDD) por sus siglas en inglés donde se plantea que los datos deben transitar por cuatro etapas. La primera etapa es pre-procesamiento, donde se realiza la limpieza, integración, selección y transformación de los datos. Le sigue minería de datos, aquí se aplican métodos inteligentes para extraer patrones de datos. Luego se aplica la evaluación de patrones, esta se usa para identificar los patrones realmente interesantes que representan el conocimiento basado en algunas medidas de interés. Por último, la presentación de conocimiento, que es donde las técnicas de visualización y representación del mismo se utilizan para presentar el conocimiento minado al usuario (Han et al., 2011).

Los datos que transitan por el proceso KDD son almacenados en un *dataset*, también conocido como conjunto de datos. “Un *dataset* es un conjunto o colección de valores que toman las variables o campos de una entidad junto con la estructura descriptiva de los mismos, organizados en forma de una matriz rectangular (filas y columnas) y que puede ser reconocida y procesada por una herramienta informática” (Rodríguez, 2014a). Estos conjuntos de datos varían su estructura en dependencia de la herramienta de análisis de información que los utilice. Algunas de estas herramientas son: *Waikato Environment for Knowledge Analysis* (WEKA), es una colección de algoritmos de aprendizaje automático(AA) para tareas de minería de datos (University of Waikato, 2018); *The Machine Learning Library* (MLlib), es la biblioteca de aprendizaje de la máquina escalable de *Apache Spark* (Meng et al., 2015) y *Orange Data Mining Fruitful & Fun*, es una máquina de aprendizaje de código abierto que presenta visualización de datos para principiantes y expertos y flujos de trabajo de análisis de datos interactivos (University of Ljubljana, 2018).

Las herramientas antes mencionadas son capaces de utilizar el aprendizaje automático sobre los conjuntos de datos que ellas cargan. Los algoritmos de AA se dividen en dos grandes enfoques, convencionales y no convencionales. El primer enfoque, es aquel que tiene como salida una sola variable; el segundo, contiene salidas múltiples. En el último enfoque mencionado uno de los problemas que se solucionan es Multi - Target

Regression (MTR). Un problema MTR consiste en la predicción de múltiples variables objetivo a partir de un conjunto común de variables, estas variables objetivos deben de estar contenidas en un dominio continuo (Spyromitros-Xioufis et al., 2016).

Existen varias herramientas que permiten ejecutar algoritmos que resuelven tareas MTR. Algunas de ellas son: MULAN, es una biblioteca de código abierto de JAVA para aprender de conjuntos de datos de etiquetas múltiples que utiliza como base WEKA (Tsoumakas et al., 2009); ShoGun, es una biblioteca de aprendizaje automático de código abierto que ofrece una amplia gama de métodos de aprendizaje de máquina eficientes y unificados (Strathmann et al., 2018); MOA, es el marco de código abierto más popular para la minería de flujo de datos, incluye una colección de algoritmos de aprendizaje automático, además está escrito en JAVA (MOA, 2018). De estas herramientas la más utilizada es MULAN debido a que ShoGun ha sido menos desarrollada que esta por la complejidad del lenguaje, y MOA está destinada principalmente a trabajar con algoritmos online.

Para ejecutar algoritmos contenidos en la herramienta MULAN que den solución a los problemas MTR es necesario utilizar datos almacenados en data sets. Porque MULAN no posee un algoritmo que permita generar los datos necesarios. Actualmente solo se cuenta con 18 conjuntos de datos reales para este tipo de problemas, todos ellos muy similares, en su mayoría con pocos datos y siempre contienen más instancias que atributos(Spyromitros-Xioufis et al., 2016). Pese a esto no se puede evaluar adecuadamente los algoritmos cubriendo todas las situaciones posibles.

Por lo antes expuesto se identifica el siguiente **problema a resolver**: ¿Cómo favorecer la experimentación científica de los modelos de aprendizaje con salidas múltiples apoyados en datos simulados con determinadas características? Este problema está enmarcado en el **objeto de estudio:** Proceso de generación de datos sintéticos para problemas de aprendizaje automático. Centrándose en el **campo de acción:** Generación de datos sintéticos en problemas de aprendizaje automático con salidas múltiples en la herramienta MULAN. Por lo que se tiene como **objetivo general:** desarrollar una herramienta informática para la creación de datos sintéticos en problemas de predicción con salidas múltiples integrado en MULAN.

Para dar cumplimiento al objetivo general se realizan las siguientes **tareas de investigación**:

1. Elaboración del marco teórico metodológico de la investigación referente a la creación de datos sintéticos en problemas de predicción con salidas múltiples.
2. Análisis de las herramientas informáticas y tecnologías que generan datos sintéticos en problemas de aprendizaje automático con salidas múltiples para identificar características o funcionalidades asociadas a la situación problemática.

3. Desarrollo de una herramienta informática de generación de datos sintéticos en problemas de predicción con salidas múltiples integrado en MULAN.
4. Validación de la herramienta a partir de las pruebas definidas en la investigación.

Para la presente investigación se utilizó como **métodos de investigación**:

Métodos Teóricos:

- **Histórico - lógico:** se utilizó para realizar la valoración de las herramientas que generan datos sintéticos, existentes a nivel internacional.
- **Analítico – sintético:** se utilizó para procesar la información referente a los algoritmos que generan datos sintéticos y arribar a conclusiones de la investigación.

Métodos Empíricos:

- **Observación:** se realizó una observación sobre los algoritmos que generan datos sintéticos para proponer una ecuación con el mismo objetivo para la herramienta MULAN.
- **Análisis documental:** se utilizó en la revisión de la literatura especializada para consultar la información necesaria en el proceso de investigación.
- **Experimentación:** Se realizó un experimento con los conjuntos de datos generados por la propuesta de solución para ver si existían diferencias significativas entre los algoritmos que resuelven problemas MTR.

El documento está estructurado en tres capítulos:

Capítulo 1. Fundamentación teórica de la investigación referente a la generación de datos sintéticos. Contiene una descripción de los ficheros ARFF y XML, también se hace un análisis del estado del arte, a nivel internacional, de las herramientas que generan datos para problemas de aprendizaje automático con múltiples salidas. Además, se describen las herramientas usadas en la solución.

Capítulo 2. Herramienta informática para la generación de datos sintéticos. En este capítulo se realiza la descripción de la herramienta, la arquitectura que esta presenta, algoritmos implementados, así como donde se incluirán estos algoritmos.

Capítulo 3. Validación de la herramienta informática para la creación de datos sintéticos. En este capítulo se documentan las pruebas que describe la metodología, que son, como se usan y que resultados se obtuvo, realizadas a la herramienta informática para la creación de datos sintéticos para la herramienta MULAN.

CAPÍTULO 1: Fundamentación teórica de la investigación referente a la generación de datos sintéticos

En este capítulo se realiza un estudio sobre herramientas que generan datos sintéticos para problemas de AA con múltiples salidas, también se describen los ficheros con extensión XML y ARFF. Además, se detallan las herramientas y tecnologías propuestas para la implementación de la solución del problema planteado.

1.1. Formatos de datos

Un formato de un archivo es una forma particular de codificar información para almacenarla (ALEGSA, 2018). En los siguientes sub-epígrafes se describen los formatos usados por la herramienta MULAN para cargar los conjuntos de datos. Debido a que la propuesta de solución debe generar un conjunto de datos en uno de estos formatos para su uso.

1.1.1. Extensible Markup Language

Extensible Mark-up Language (XML) es un formato de texto simple y muy flexible derivado de Estándar Generalized Mark-up Language (SGML). XML es un meta-lenguaje que nos permite definir lenguajes de marcado adecuados a usos determinados. Esta extensión de fichero presenta algunas características como (Exes, 2018):

- XML es un subconjunto de SGML que incorpora las tres características más importantes de este:
  - Extensibilidad
  - Estructura
  - Validación
- Basado en texto.
- Orientado a los contenidos no presentación.
- Las etiquetas se definen para crear los documentos, no tienen un significado preestablecido.
- No es sustituto de HTML.
- No existe un visor genérico de XML.

XML presenta como ventajas (Exes, 2018):

- Fácilmente procesable
- Separa radicalmente el contenido y el formato de presentación

- Diseñado para cualquier lenguaje y alfabeto. (encoding)

En un documento XML existen los siguientes componentes (Exes, 2018):

- Elementos: Pieza lógica del marcado, se representa con una cadena de texto encerrada entre etiquetas. Pueden existir elementos:
  - vacíos (
).
  - Los elementos pueden contener atributos.
- Instrucciones: Ordenes especiales para ser utilizadas por la aplicación que procesa <?xml-stylesheet type="text/css" href="estilo.css">
  - Comienzan por <? Y terminan por ?>
- Comentarios: Información que no forma parte del documento.
  - Comienzan por <!-- y terminan por ->.
- Declaraciones de tipo: Especifican información acerca del documento:
  - <!DOCTYPE persona SYSTEM "persona.dtd">
- Secciones CDATA: Se trata de un conjunto de caracteres que no deben ser interpretados por el procesador:
  - <![CDATA[Aquí se puede meter cualquier carácter, como <, &, >, ... Sin que sean interpretados como marcación]]>

El archivo XML debe de tener una estructura para su utilización en MULAN como la presentada a continuación. Además se puede expresar jerarquía entre las etiquetas (Tsoumakas et al., 2018):

```
<?xml version="1.0" encoding="utf-8"?>
<labels xmlns="http://mulan.sourceforge.net/labels">
    <label name="sports">
            <label name="football"></label>
            <label name="basketball"></label>
    </label>
    <label name="arts">
            <label name="sculpture"></label>
            <label name="photography"></label>
    </label>
</labels>
```

Figura 1 Ejemplo de estructura de archivo XML

1.1.2. Attribute-Relation File Format

Los archivos ARFF tienen formato de texto plano en *American Standard Code for Information Interchange* (ASCII), por lo que pueden ser visualizados y modificados desde cualquier editor de texto, siguiendo unas normas básicas(txikiboo, 2014):

- Para escribir comentarios en ellos, se debe empezar la línea con %.
- Para las declaraciones de relación, atributos y datos, se usa @ al comienzo.
- Todo texto que incluya espacios debe ir entrecomillado (excepto en los comentarios). Esto se aplica a toda información de tipo String, tanto datos como nombres.

Este tipo de fichero es usado por algunas herramientas dedicadas al análisis de la información. Aunque varían en su estructura, siempre tiene bien definida su cabecera, el área donde se declaran los atributos y los datos. Lo anterior se sustenta a que *Multilabel Extension to WEKA* (MEKA) usa atributos múltiples, uno para cada objetivo o etiqueta, en lugar de un solo atributo de clase. El número de atributos de destino se especifica con -C o -c; a diferencia de WEKA, donde la bandera -c indica la posición del índice de la clase. MEKA utiliza la referencia a classIndex internamente para indicar el número de atributos de destino. Dado que el número de atributos objetivo tiende a variar con cada conjunto de datos, para mayor comodidad, MEKA permite que esta opción se almacene en el nombre @relación de un archivo ARFF, donde se usan dos puntos (:) para separar el nombre del conjunto de datos y las opciones («Tutorial. Meka 1.9.0», 2015).

Para que este fichero sea usado en WEKA se define una estructura. La misma debe empezar en la cabecera con @relation <nombre>, el <nombre> debe ser el nombre del conjunto de datos, esta fila debe estar separada de las siguientes con un retorno de carro, luego se declaran los atributos; ejemplo: @attribute <nombre> <Tipo> o @attribute <nombre> {<valor>, …, <valor>}. Además, la etiqueta <valor> toma los posibles valores de ese atributo, la diferencia es que el primero se usa para datos de tipo entero o flotantes y el segundo para datos que contengan caracteres como fechas, entre otras. Luego separado por otro retorno de carro viene @data, significa que lo que viene a continuación son todos los datos del fichero, los datos reales que deben cumplir con las declaraciones anteriores. Dichos datos deben de estar separados por coma. A continuación, se muestra un ejemplo.

```
@relation 'football-2ndlevel-trainning'

@attribute vectorlengthL0 numeric
@attribute residual16x16 numeric
@attribute MBtypeAVC {0,1,2,3,8,9,10,11}
@attribute meansofvariances4x4 numeric
@attribute varianceofmeans4x4 numeric

@attribute class {0,1}

@data
16.40,1687.00,8,33.80,57.87,1
0.71,1715.00,2,95.19,68.55,1
1.00,132.00,1,1.86,0.73,0
```

Figura 2 Ejemplo de estructura del fichero ARFF

MULAN maneja el formato de ARFF utilizado por WEKA. Además, las etiquetas deben especificarse como atributos nominales con dos valores "0" y "1" que indican la ausencia o existencia de la etiqueta, respectivamente (Tsoumakas et al., 2018).

1.1.3. Comma Separated Values

El formato de valores separados por comas (CSV) se ha utilizado para intercambiar y convertir datos entre varios programas de hojas de cálculo durante bastante tiempo (Shafranovich, 2005). Alguna de sus ventajas son (GitHub, 2018):

- Los editores de texto como el bloc de notas pueden abrir o editar archivos CSV.
- En almacenes de datos, CSV sigue un esquema bastante plano y simple.
- Generar un archivo CSV en cualquier lenguaje de programación es fácil.
- CSV es seguro y puede diferenciar claramente entre los valores numéricos y el texto. CSV no manipula los datos y los almacena tal cual.
- En CSV, escribe encabezados de columna solo una vez donde, como en Excel, debe tener una etiqueta de inicio y una etiqueta de finalización para cada columna en cada fila.
- La importación de archivos CSV puede ser mucho más rápida y también consume menos memoria.
- Es fácil manipular un archivo CSV ya que, después de todo, son archivos de texto simples.

A continuación, se muestra un ejemplo de la estructura que contiene un fichero de extensión CSV.

```
Cup, Blue, 0.98
Saucer, Blue, 0.45
Jug, Clear Glass, 3.75
Dish, Patterned, 2.99
Mug, Plain White, 0.99
Mug, White with Logo, 1.75
```

Figura 3 Ejemplo de fichero CSV

1.2. Algoritmos y modelos para generar datos sintéticos

En este epígrafe se caracterizan y comparan los algoritmos existentes a nivel internacional que realizan el proceso de generación de datos sintéticos. Como el objetivo es desarrollar un generador de datos sintéticos para problemas MTR se desechó aquellas soluciones con una sola salida. Además, se analizaron algunos algoritmos que generan datos sintéticos para problemas Multi-Label Classification debido a que es sencillo extender el código para generar datos sintéticos para problemas MTR.

1.2.1. Datos simulados para el algoritmo DMLKNN

El conjunto de datos simulado contiene 1019 instancias en R^2 que pertenecen a tres clases posibles, $\mathrm{Y} = \{\omega_1, \omega_2, \omega_3\}$. Los datos fueron generados a partir de siete distribuciones gaussianas con medias (0,0), (1,0), (0.5,0), (0.5,1), (0.25,0.6), (0.75,0.6), (0.5,0.5), respectivamente, y matriz de covarianza equivalente a matriz de identidad de 2 x 2 en cuanto la distribución geométrica de los datos de Gauss, las instancias de cada conjunto se asignaron respectivamente a la (s) etiqueta (s) $\{\omega_1\}, \{\omega_2\}, \{\omega_3\}, \{\omega_1, \omega_2\}, \{\omega_1, \omega_3\}, \{\omega_2, \omega_3\}, \{\omega_1, \omega_2, \omega_3\}$ (Member et al., 2011).

1.2.2. Datos simulados para el algoritmo MLNB

Los datos sintéticos se crean de la siguiente manera: se supone que hay una hiperesfera HS con radio r ubicado en un espacio de características d-dimensional. Generando aleatoriamente Q hiperesfera internas hs_l $(\mathrm{l} \in \{1, 2, \ldots, \mathrm{Q}\})$ todas incrustadas en la hiperesfera externa HS, donde cada hs_l corresponde a una clase de concepto a aprender. En base a esto, los puntos de datos (instancias) se generan aleatoriamente en HS, cada punto x se asocia con un conjunto de etiquetas $\mathrm{Y} = \{\mathrm{l} \mid \mathrm{x\ cubierto\ por}\ hs_l, \mathrm{l} \in \{1, 2, \ldots, \mathrm{Q}\}\}$ (Zhang et al., 2009).

1.2.3. Estrategia de Hiperesferas

La estrategia de Hiperesferas tiene 9 parámetros de entrada para generar conjuntos de datos con las características $M = M_{rel} + M_{irr} + M_{red}$ (Torres et al., 2014).

Parámetros:

- M_{rel}: Number of relevant features.
- M_{irr}: Number of irrelevant features.
- M_{red}: Number of redundant features.
- q: Number of labels in the dataset.
- N: Number of instances in the dataset.
- maxR: Maximum radius of the small hyperspheres.
- minR: Minimum radius of the small hyperspheres.
- name: Name of the relation in the header of the ARFF file

Los pasos principales para generar un conjunto de datos sintéticos son (Torres et al., 2014):

1. Generación de la hiperesfera HS en $R^{M_{rel}}$.
    - Se crea un HS de hiperesfera en $R^{M_{rel}}$, centrado en el origen del sistema de coordenadas cartesianas y con un radio $r_{HS} = 1$.
2. Generando las q hiperesferas pequeñas hs_i, i $= 1..$ q dentro de HS.

Tabla 1 Generación de pequeñas hiperesferas

Algorithm 1 Generation of the small hyperspheres
1: **for** i = 1 → q **do**
2: $C_i \leftarrow \emptyset$
3: $r_i \leftarrow$ random(minR, maxR)
4: maxC ← (1 - r_i)
5: minC ← - (1 - r_i)
6: **for** j = 1 → M_{rel} (j randomly defined) **do**
7: $c_{ij} \leftarrow$ random(minC, maxC)
8: $C_i \leftarrow C_i \cup \{ c_{ij} \}$
9: minC ← updateminC (minC)
10: maxC ← updatemaxC (maxC)
11: **end for**
12: $hs_i \leftarrow (r_i , C_i)$
13: **end for**
14: **return** { $hs_1, hs_2, \dots , hs_q$}

3. Generación de los N puntos (instancias) en R^{M}.

Tabla 2 Generación de puntos dentro de la hiperesfera

Algorithm 2 Generation of the points inside the hyperspheres
1: **for** k = 1 → N **do**
2: $x_k \leftarrow \emptyset$
3: maxX ← c_{ij} - r_i
4: minX ← c_{ij} + r_i
5: **for** j = 1 → M_{rel} (j randomly defined) **do**
6: x_{kj} ← random(minX, maxX)
7: $x_k \leftarrow x_k \cup \{x_{kj}\}$
8: minX ← updateminX (minX)
9: maxX ← updatemaxX (maxX)
10: **end for**
11: **end for**
12: **return** $\{x_1, x_2, \ldots, x_N\}$

4. Generación de etiquetas múltiples basadas en las q hiperesferas pequeñas e inserción del nivel o niveles de ruido μ
   - El procedimiento para asignar la etiqueta y_i a la etiqueta múltiple Y_k de $x_k \forall k \in [1..N]$ se implementa como se define en la ecuación $\sqrt{(x_{kj} - c_{ij})^2} \leq r_i, \forall i \in [1..q], \forall j \in [1..M_{rel}], \forall k \in [1..N]$. Solo deben tenerse en cuenta las características de M_{rel}.
   - Para generar los datos con ruidos cada instancia Ei, el procedimiento para insertar ruido en el conjunto de datos construido invierte la etiqueta $y_j \in Y_i$, j = 1..q, con probabilidad μ. En otras palabras, si la etiqueta y_j está en la etiqueta múltiple Y_i, la tapa quitará y_j del sello múltiple Y_i con probabilidad μ. De lo contrario, la etiqueta y_j se insertará con probabilidad μ.

1.2.4. Estrategia Hipercubos

La estrategia Hipercubos también tiene 9 parámetros y las restricciones correspondientes para generar data sets sintéticos sin y con ruido. Sin embargo, a diferencia de la estrategia de Hiperesferas, maxR y minR denotan, respectivamente, los medios-bordes máximos y mínimos de los pequeños hipercubos. Los principales pasos considerados para generar conjuntos de datos sintéticos de acuerdo con la estrategia Hipercubos son similares a los requeridos por Hiperesferas (Torres et al., 2014).

1. Generación del hipercubo HC en $R^{M_{rel}}$.

- Se crea un hipercubo HC en $R^{M_{rel}}$, centrado en el origen del sistema de coordenadas cartesianas con medio borde $e_{HC} = 1$.

2. Generando los pequeños hipercubos q hc_i, i = 1..q dentro de HC.

Tabla 3 Generación de pequeños hipercubos

Algorithm 3 Generation of the small hypercubes
for i = 1 → q **do**
$C_i \leftarrow \emptyset$
$e_i \leftarrow$ random(minR, maxR)
maxC ← (1 - e_i)
minC ← - (1 - e_i)
for j = 1 → M_{rel} **do**
$c_{ij} \leftarrow$ random(minC, maxC)
$C_i \leftarrow C_i \cup \{ c_{ij} \}$
end for
$hc_i \leftarrow (e_i , C_i)$
end for
return { $hc_1, hc_2, \dots , hc_q$}

3. Generación de los N puntos (instancias) en R^M.

Tabla 4 Generación de los puntos dentro de los hipercubos

Algorithm 4 Generation of the points inside the hypercubes
for k = 1 → N **do**
$x_k \leftarrow \emptyset$
maxX ← c_{ij} - e_i
minX ← c_{ij} + e_i
for j = 1 → M_{rel} (j randomly defined) **do**
$x_{kj} \leftarrow$ random(minX, maxX)
$x_k \leftarrow x_k \cup \{ x_{kj} \}$
end for
end for
return { $x_1, x_2, \dots , x_N$}

4. Generación de etiquetas múltiples basadas en q pequeños hipercubos e inserción de niveles de ruido μ

- De forma similar a Hiperesferas, cualquier instancia x_k ∀k ∈ [1..N] tiene la etiqueta y_i , i = 1..q, en su Y_k de etiqueta múltiple, si x_k está dentro del hipercubo hc_i. El Y_k multi-label final se compone de todas las etiquetas que cumplen esta condición, que se puede verificar fácilmente según la distancia entre x_k y cada centro C_i, i = 1..q. Si esta distancia es menor que la mitad del borde e_i, entonces x_k está dentro de hc_i y $y_i \in Y_k$; de lo contrario, $y_i \notin Y_k$. El procedimiento para asignar la etiqueta y_i al Y_k de etiqueta múltiple de x_k ∀k ∈ [1..N] se

implementa como se define en la ecuación $|(x_{kj} - c_{ij})| \leq e_i, \forall i \in [1..q]\, e\, \forall j \in [1..Mrel]$. Tenga en cuenta que solo se deben tener en cuenta las características de M_{rel}.

1.2.5. Datos simulados para el algoritmo BRRR

Los datos sintéticos se generan de acuerdo a $Y = (X\Psi + \alpha\Omega)\,\Gamma + (1 - \alpha)H\Lambda + E$ donde $\mathrm{vec}(E) \sim N(0, I_{NK})$ y los parámetros $\alpha \in [0,1]$ define la proporción de varianza atribuida al ruido latente frente al ruido independiente. Los parámetros Γ y Λ están ortogonalizados usando la ortogonalización Gramm-Schmidt. Los parámetros se escalan para que las covariables X expliquen el 3% de la varianza de Y a $X\Psi\Gamma$, el ruido gaussiano diagonal $N(0, I_{NK})$ explica el 20% de la varianza total de Y, y el ruido estructurado $\alpha\Omega\Gamma + (1 - \alpha)H\Lambda$ explica el 77% restante de la varianza total de Y (Gillberg et al., 2016).

1.2.6. Fuzzy c-regression(FCR) y Fuzzy c-means(FCM)

FCM

El algoritmo FCM hace una partición difusa de un conjunto dado de elementos. Desde un punto de vista conceptual, las categorías de datos subyacentes se consideran difusas. Luego, con un conjunto de objetos $X = \{x_1, x_2, \dots, x_n\}$ evaluado en términos de atributos $A = \{A_1, A_2, \dots, A_m\}$ fuzzy c-means hace una partición difusa de los objetos X. Por lo tanto, considerando las categorías c $C = \{C_1, C_2, \dots, C_c\}$, el problema es la determinación de las funciones de pertenencia c $\mu_1, \mu_2, \dots, \mu_c$ donde μ_i es la función de pertenencia correspondiente a C_i. μ_i son tales que para cada objeto x su membresía a toda la categoría C se agrega a uno. Además, se requiere al menos un elemento con una membresía distinta de cero para cada categoría. Por lo tanto, las funciones de membresía deben cumplir las dos condiciones siguientes (Cano et al., 2009):

- $\sum_{i=1}^{c} \mu_i(x) = 1$ para toda $x \in X$
- $0 < \sum \mu_i(x)_{x \in X} < N$ para toda $C_i \in C$

Una vez que se tiene las restricciones de la membresía, el problema puede formularse de la siguiente manera. Los parámetros a minimizar son μ y P, donde μ es igual que el anterior y P son los centroides P_k de los conglomerados $\mathrm{k} = 1 \dots \mathrm{c}$. Minimizar $FO(\mu, P) = \sum_{k=1}^{c} \sum(\mu_k(x))^m{}_x \,||\, A(x) - \mathrm{p}_k||^2$ prohibido para: $\mu \in \mathrm{M}_f = \{(\mu_k(x)) | \mu_k(x) \in [0,1], \sum_{k=1}^{c} \mu_k(x) = 1, \forall x \in \mathrm{X}\}$. Donde c es un valor constante que representa el número de categorías difusas permitidas. El otro valor constante m, que debe ser mayor que 1, es el grado

de difuminado de las categorías. Cuanto mayor es el valor de m, más difusas son las categorías. Cuando $m \to \infty$, todas las categorías cubren todos los puntos. Por el contrario, cuanto menor es m, menos confusas son las categorías. Cuando m = 1 corresponde al algoritmo c-means El algoritmo FCM construye una solución factible del problema anterior de la siguiente manera (Cano et al., 2009):

1. Se define una partición inicial μ y se calcula los centroides P. Esto se puede hacer al azar.
2. En este segundo paso, para cada elemento x_i actualice la membresía de x_i a cada categoría C_k de la siguiente manera:
   - Si $|| x_i - p_k||^2 > 0$ entonces cada categoría de C_k: $\mu_k(x_i) = \left[\sum_{j=1...c} \left(\frac{|| x_i - p_k||^2}{|| x_i - p_j||^2} \right)^{\frac{1}{(m-1)}} \right]^{-1}$
   - Si hay alguna categoría $C_k$para la cual $|| x_i - p_k||^2 = 0$ significa que x_i tiene el mismo valor que algún centroide p_k. Por lo tanto, en este caso, la membresía de x_i debe ser compartida al azar con todos los centroides que coincidan con x_i
3. El siguiente paso es actualizar el valor del centroide. Entonces, para cada categoría C_k, su centroide se define de la siguiente manera:
   - $p_k = \frac{\sum_{i=1}^{n} (\mu_k(x_i))^m A(x_i)}{\sum_{i=1}^{n} (\mu_k(x_i))^m}$ y para el componente j-ésimo, esto se define como:
   - $A_j(p_k) = \frac{\sum_{i=1}^{n} (\mu_k(x_i))^m A_j(x_i)}{\sum_{i=1}^{n} (\mu_k(x_i))^m}$
4. Deténgase cuando se exceda el umbral de convergencia; de lo contrario, vaya al paso 2. El umbral de convergencia se puede definir como la comparación de las funciones de membresía de dos iteraciones consecutivas. Teniendo en cuenta un umbral λ y también μ' y μ ya que la membresía funciona a partir de dos iteraciones consecutivas, la condición de detención se puede definir de la siguiente manera:
   - $\lambda > \max_{k=1}(\max_{x \in X} | \mu'_k(x) - \mu_k(x)|)$

FCR

Los modelos difusos de regresión c (FCRM) son una familia de funciones objetivas que se pueden usar para ajustar los modelos de regresión de conmutación a datos mixtos numéricos y continuos. Para un c dado (el

número de conglomerados, 1 <c <n), el algoritmo de regresión c difusa es capaz de obtener una estimación de los parámetros de los modelos de regresión c, junto con una partición difusa c de los datos (Cano et al., 2009).

Se considera un conjunto de datos de objeto de tamaño n, $S = \{(x_1, y_1), \dots, (x_n, y_n)\}$ donde cada vector de característica (x_i, y_i) tiene una observación dependiente $y_i \in R^t$ correspondiente a una determinada observación independiente $x_i \in R^s$. La principal diferencia entre los modelos de regresión c difusa y los problemas de ajuste de datos más simples es que los segundos suponen que una relación funcional única entre x y y se cumple para todos los datos mientras que los primeros suponen que los datos se ahogan de los modelos c: $y = f_i(\mathrm{x}, \beta_i) + \varepsilon$, $1 \leq i \leq c$ cada $\beta_i \in \Omega_i \subset R^{k_i}$, y cada ε_i es un vector aleatorio con el vector medio $\mu_i = 0 \in R^t$ y la matriz de covarianza Σi. Se debe decir que S no está etiquetado, por lo tanto, para un vector de características dado (x_i, y_i), no se sabe qué modelo de 1 se aplica. Hathaway y Bezdek publicaron en una solución factible para este problema. Su enfoque se basa en técnicas de agrupamiento difusas y es capaz de producir buenas estimaciones de $\{\beta_1, \dots, \beta_c\}$ mientras se etiqueta con un vector etiqueta fuzzy cada dato en S. El problema de etiquetado se resuelve por medio de la agrupación difusa asignando vectores de etiquetas restringidas que representan la pertenencia de cada objeto (x_i, y_i) a cada una de las clases c (Cano et al., 2009).

El algoritmo para construir los FCRM tiene pasos similares a los de FCM (Cano et al., 2009):

1. Dado un conjunto de datos de objeto $S = \{(x_1, y_1), \dots, (x_n, y_n)\}$. Se establece m> 1 (una elección razonable es m = 1.5), se especifica los modelos 1 de regresión y se elige una medida del error $E = \{E_{ik}\}$ para que $E_{ik}(\beta_i) \geq 0$ para que i y k satisfaga también la propiedad del minimizador. Se elige un umbral de terminación > 0 (una elección para ε en el rango de 0.0001 a 0.00001 generalmente produce buenas estimaciones) y una partición inicial $U^{(0)} \in M_f$. A continuación, se establece un umbral para r_{max}, el número máximo de iteraciones, de modo que $\mathrm{r} = 1, \dots, r_{max}$.
2. Se actualiza los valores para los parámetros del modelo c $\beta_i = \beta_i{}^{(\mathrm{r})}$ y luego la medida del error $E_{ik}(\beta_i)$ en $f_i(x_k;\, \beta_i)$ que minimice globalmente (sobre $\Omega_1 \times \Omega_2 \times \dots \times \Omega_c$) la función restringida: $\psi(\beta_1, \dots, \beta_c) \equiv E_m(U^{(r)}, \beta_1, \dots, \beta_c)$. El ejemplo más común para la medida del error $E_{ik}(\beta_i)$ es la norma del vector al cuadrado $E_{ik}(\beta_i) = \left|\left| f_i(x_k; \beta_i) - y_k \right|\right|^2$. El segundo paso puede especificarse fijando $\Omega_c = R^s$, $f_i(x_k; \beta_i) = (x_k)^T \beta_i$ y 1 ≤ i ≤ c, por lo tanto, la función objetivo $E_m(U^{(r)}, \beta_1, \dots, \beta_c)$ se convierte en una extensión multimodel difusa del criterio de mínimos cuadrados para la

adaptación del modelo: $E_{ik}(\beta_i) = (y_k - (x_k)^T \beta_i)^2$. Además, los nuevos valores para los parámetros del modelo de regresión β_i (r), 1 ≤ i ≤ c se pueden calcular utilizando la siguiente fórmula explícita si las columnas de X son linealmente independientes y $U_{ik}{}^{(r)} > 0$ para 1 ≤ k ≤ n: β_i (r) = $[X^{\mathrm{T}} D_i X]^{-1} X^{\mathrm{T}} D_i Y$ donde X denota la matriz en $R^{\mathrm{n}\times\mathrm{s}}$ teniendo x_k como su k-ésima fila. Y denota el vector en R^{n} que tiene y_k como su componente k-ésima, y D_i denota la matriz diagonal en $R^{\mathrm{n}\times\mathrm{n}}$ que tiene $(U_{ik}{}^{(r)})^{\mathrm{m}}$ como su k-ésimo elemento diagonal.

3. El objetivo de este paso es actualizar $U^{(r)} \rightarrow U^{(r+1)} \in M_f$, interpretando U_{ik} como la importancia o el peso asociado a la medida en que el valor del modelo $f_i(x_k; \beta_i)$ concuerda con y_k (membresía difusa en todos los modelos c). La actualización se realiza mediante la siguiente fórmula:

   a. $U_{ik} = \left[\sum_{j=1}^{c} \left(\frac{E_{ik}}{E_{jk}} \right)^{\frac{1}{(\mathrm{m}-1)}} \right]^{-1}$, si $E_{ik} > 0$ para 1 ≤ i ≤ c

      En caso de que se encuentre alguna $E_{ik} = 0$, su valor puede ser reemplazado por la adición de un pequeño número positivo.

4. Si $U^{(r)} - U^{(r+1)}$ correspondiente a dos iteraciones consecutivas es mayor que el umbral de terminación, o $r \leq r_{max}$ entonces r: = r + 1 y se va al paso 2. De lo contrario, se detiene.

Para generar datos sintéticos utilizando los algoritmos Fuzzy c- Means y Fuzzy c-Regression es necesario seguir los siguientes pasos (Cano et al., 2009):

1. El primer paso es dividir el conjunto de datos en x independientes y atributos y dependientes.
2. El siguiente paso es ejecutar el algoritmo Fuzzy c-regression arrancándolo con Fuzzy c-Means para calcular la partición inicial $U^{(0)} \in M_f$.
3. Una vez que FCRM finaliza, aparece el paso de generación de datos sintéticos. Para cada vector de función $(x_s, y_s) \in S$, $1 \leq s \leq n$, seleccione el i-ésimo clúster con membresía máxima. Es decir, seleccione el $\arg max^{c}{}_{i=1}\, U_{is}$. Ahora se sabe que el centroide p_i se aproxima mejor a cada vector de características. Por lo tanto, se puede usar el modelo de regresión β_i correspondiente a cada centroide p_i para pronosticar el valor sintético y'_s para reemplazar el y_s original

1.2.7. Algoritmo para generar datos para regresiones múltiples

Primero, se genera una matriz definida positiva aleatoria Σ^{-1} que actuará como la matriz de covarianza inversa de la tarea. A continuación, se genera una matriz V de tamaño D × K con cada entrada muestreada

a partir de una distribución normal de media cero y $1 / \mathrm{D}$ de varianza. Se calcula la raíz cuadrada S de Σ (= SS, donde S es también una matriz definida positiva simétrica), y S se usa para generar la matriz de peso final W como W = VS. Está claro que, para un W generado de esta manera, tendremos $\mathrm{E}\left[\mathrm{W}^{\mathrm{T}}\mathrm{W}\right] = SS = \Sigma$. Este proceso genera W de manera que sus columnas (y por lo tanto los vectores de ponderación para diferentes salidas) están correlacionados. Un vector de tendencia b de tamaño K se genera aleatoriamente a partir de una distribución normal de varianza de unidad media cero. Luego se genera una matriz definida aleatoria dispersa Ω^{-1} que actúa como la matriz de covarianza inversa condicional en el ruido de salida haciendo que las salidas estén correlacionadas (dadas las entradas). A continuación, se generan las variables de salida multivariantes correspondientes se generan como $y_i = Wx_i + \mathrm{b} + \varrho_i \, \forall \mathrm{i} = 1, 2, \ldots, \mathrm{N}$, donde ϱ_i es el vector de ruido correlacionado muestreado aleatoriamente a partir de una distribución normal media cero con la matriz de covarianza Ω (Rai et al., 2012).

1.2.8. Datos sintéticos generados por Massimiliano Pontil

Se crea un conjunto de datos sintéticos mediante la generación de T = 200 parámetros de tareas w_t a partir de una distribución Gaussiana de 5 dimensiones con media cero y covarianza igual a Diag (1, 0.25, 0.1, 0.05, 0.01). Estas son las dimensiones relevantes que deseamos aprender. A estos seguimos sumando hasta 20 dimensiones irrelevantes que son exactamente cero. Los conjuntos de entrenamiento y prueba se seleccionaron al azar de $[0, 1]^{25}$ y contenían 5 y 10 ejemplos por tarea, respectivamente. Las salidas y_{ti} se calcularon a partir de w_t y x_{ti} como $y_{ti} = (\mathrm{w}_t, x_{ti}) + \nu$, donde ν es ruido gaussiano de media cero con una desviación estándar igual a 0,1 (Massimiliano Pontil et al., 2007).

1.2.9. Comparación de algoritmos

Para la sucesiva comparación se tomó en cuenta los siguientes aspectos:

- Tipo de problema
- Propósito
- Existe correlación entre las salidas

Tabla 5 Comparación entre los algoritmos

Algoritmos	Tipo de problema	Propósito	Dependencia entre salidas

Datos simulados para el algoritmo DMLKNN	Multi-Label	General	No
Datos simulados para el algoritmo MLNB	Multi-Label	General	No
Estrategia de Hiperesferas	Multi-Label	General	No
Estrategia Hipercubos	Multi-Label	General	No
Datos simulados para el algoritmo BRRR	Multi-Target	Modelos de ruido	Si
FCR y FCM	Multi-Target	General	No
Algoritmo para generar datos para regresiones múltiples	Multi-Target	General	Si
Datos sintéticos generados por Massimiliano Pontil	Multi-Task	General	No

Luego de analizar los distintos algoritmos que generan datos. Se concluye que el único que cumple con los aspectos de propósito general y crear dependencia entre las salidas es el algoritmo para generar datos para regresiones múltiples (Rai et al., 2012). Por lo tanto, se tomará como base para la implementación de la propuesta de solución.

1.3. Ambiente de desarrollo

En este epígrafe se caracterizan las herramientas y metodología seleccionadas para la implementación de la propuesta de solución.

1.3.1. MULAN

MULAN es una biblioteca de código abierto de Java para aprender de conjuntos de datos de etiquetas múltiples. Esta es en realidad la naturaleza de muchos problemas del mundo real, como la anotación semántica de imágenes y video, la categorización de páginas web, el marketing directo, la genómica funcional y la categorización de la música en géneros y emociones. Actualmente, la biblioteca incluye una variedad de algoritmos de última generación para realizar las siguientes tareas principales de aprendizaje con múltiples etiquetas (Tsoumakas, G. et al., 2018):

- Clasificación: Esta tarea se refiere a la salida de una bipartición de las etiquetas en relevantes e irrelevantes para una instancia de entrada determinada.
- Ranking: Esta tarea se refiere a generar un orden de las etiquetas, de acuerdo con su relevancia para un elemento dado.
- Clasificación y ranking: Una combinación de las dos tareas mencionadas arriba.

Además, la biblioteca ofrece las siguientes características (Tsoumakas, G. et al., 2018):

- Selección de características. Los métodos básicos simples son actualmente compatibles.

- Evaluación. Clases que calculan una gran variedad de medidas de evaluación a través de evaluación de resistencia y validación cruzada.

Cada versión de MULAN requiere una versión diferente de WEKA y JAVA para ejecutar. Además, se requiere una versión específica de JUnit para ejecutar pruebas. Además MULAN no posee un generador de conjuntos de datos para problemas MTR (Tsoumakas, G. et al., 2018). En la presente investigación se utilizará la versión 1.5.0 de MULAN, WEKA en su versión 3.7.10, JAVA 8, y jUnit en su versión 4.10.

MULAN utiliza el patrón arquitectónico modular. La Arquitectura modular se refiere al diseño de sistemas compuestos por elementos separados que pueden conectarse preservando relaciones proporcionales y dimensionales. La belleza de la arquitectura modular se basa en la posibilidad de reemplazar o agregar cualquier componente sin afectar al resto del sistema (Meneses Gómez, Mateo, 2017).

1.3.2. Visual Paradigm v8.0

Visual Paradigm ofrece a los desarrolladores de software una plataforma de desarrollo innovadora para crear aplicaciones de calidad de manera más rápida, mejor y más económica. Facilita una excelente interoperabilidad con otras herramientas CASE y la mayoría de los IDE líderes que sobresalen todo su proceso de desarrollo Model-Code-Deploy.(Scribd Inc, 2018)

Las ventajas que proporciona Visual Paradigm for UML son:

- Dibujo. Facilita el modelado de UML, ya que proporciona herramientas específicas para ello. Esto también permite la estandarización de la documentación, ya que la misma se ajusta al estándar soportado por la herramienta.
- Corrección sintáctica. Controla que el modelado con UML sea correcto.
- Coherencia entre diagramas. Al disponer de un repositorio común, es posible visualizar el mismo elemento en varios diagramas, evitando duplicidades.
- Integración con otras aplicaciones. Permite integrarse con otras aplicaciones, como herramientas ofimáticas, lo cual aumenta la productividad.
- Trabajo multiusuario. Permite el trabajo en grupo, proporcionando herramientas de compartición de trabajo.

- Reutilización. Facilita la reutilización, ya que disponemos de una herramienta centralizada donde se encuentran los modelos utilizados para otros proyectos.
- Generación de código. Permite generar código de forma automática, reduciendo los tiempos de desarrollo y evitando errores en la codificación del software.

Generación de informes. Permite generar diversos informes a partir de la información introducida en la herramienta.

1.3.3. Netbeans v8.2

NetBeans Integrated Development Environment (IDE): Es un entorno de desarrollo. Una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Está escrito en Java, pero puede servir para cualquier otro lenguaje de programación. *NetBeans* IDE es un producto libre y gratuito sin restricciones de uso (Oracle Corporation, 2017b).

NetBeans IDE 8.2 proporciona analizadores y editores de código listos para usar para trabajar con las últimas tecnologías Java 8: Java SE 8, Java SE *Embedded* 8 y Java ME *Embedded* 8. El IDE también tiene una gama de nuevas herramientas para HTML5 / JavaScript, en particular para Node.js, *KnockoutJS* y AngularJS; mejoras que mejoran aún más su soporte para *Maven* y Java EE con *PrimeFaces*; y mejoras al soporte de PHP y C / C ++ (Oracle Corporation, 2017b).

Este IDE tiene como ventajas(Sornoza et al., 2015):

- Desarrollo para cualquier tipo de aplicativo
- Reutilización de módulos
- Instalación y actualización no compleja
- Módulo centro de actualización
- Es de código abierto.

Y presenta como desventajas:

- Pesado, en el caso de que existe muchos proyectos
- *Plug-in* escasos

- Falta de apoyo *webapps*

1.3.4. Lenguajes de desarrollo

Lenguaje de programación es aquella estructura que, con una cierta base sintáctica y semántica, imparte distintas instrucciones a un programa de computadora.

UML

Es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; está respaldado por el OMG (Object Management Group). Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio, funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y compuestos reciclados (SCHOLARIUM SAS, 2017).

JAVA

El lenguaje Java™ fue creado por Sun Microsystems Inc. en un proceso por etapas que arranca en 1990, año en el que Sun creó un grupo de trabajo, liderado por James Gosling, para desarrollar un sistema para controlar electrodomésticos e incluso PDAs o Asistentes Personales (pequeños ordenadores) que, además, permitiera la conexión a redes de ordenadores. Se pretendía crear un hardware polivalente, con un Sistema Operativo eficiente (SunOS) y un lenguaje de desarrollo denominado Oak (roble), el precursor de Java. El proyecto finalizó en 1992 y resultó un completo fracaso debido al excesivo coste del producto, con relación a alternativas similares, tras lo cual el grupo se disolvió. Este lenguaje proporciona numerosas comprobaciones en compilación, tiempo de ejecución e interpretación, lo que lo ha llevado a ubicarse entre los lenguajes de preferencias entre la comunidad de desarrolladores a nivel internacional.(KEN ARNOLD; JAMES GOSLING; DAVID HOLMES, 2001)

Tiene como características (KEN ARNOLD; JAMES GOSLING; DAVID HOLMES, 2001):

- Es intrínsecamente orientado a objetos.
- Funciona perfectamente en red.
- Aprovecha características de la mayoría de los lenguajes modernos evitando sus inconvenientes. En particular los del C++.

- Tiene una gran funcionalidad gracias a sus librerías (clases).
- No tiene punteros manejables por el programador, aunque los maneja interna y transparentemente.
- El manejo de la memoria no es un problema, la gestiona el propio lenguaje y no el programador.
- Genera aplicaciones con pocos errores posibles.
- Incorpora Multi-Threading (para permitir la ejecución de tareas concurrentes dentro de un mismo programa).

JAVA 8 incluye nuevas características, mejoras y correcciones de bugs para mejorar la eficacia en el desarrollo y la ejecución de programas Java. Estas características son (Oracle Corporation, 2017a):

- Nueva API para Date y Time.
- Varias mejoras de seguridad.
- Se incorporan en la máquina virtual *Java HotSpot* características que estaban en *JRockit*, convergiendo ambas máquinas virtuales. Se incorpora Mission Control.
- Mejoras en JDBC la base para el acceso en bases de datos en Java.

1.3.5. Biblioteca

En ciencias de la computación, una biblioteca (del inglés library) es un conjunto de subprogramas utilizados para desarrollar software. Las bibliotecas contienen código y datos, que proporcionan servicios a programas independientes, es decir, pasan a formar parte de estos. Esto permite que el código y los datos se compartan y puedan modificarse de forma modular. Algunos programas ejecutables pueden ser a la vez programas independientes y bibliotecas, pero la mayoría de estas no son ejecutables. Ejecutables y bibliotecas hacen referencias (llamadas enlaces) entre sí a través de un proceso conocido como enlace, que por lo general es realizado por un software denominado enlazador (Google Sites, 2018).

JFreeChart:

Una biblioteca de gráficos libre de Java gratuita que facilita a los desarrolladores la visualización de gráficos de calidad profesional en sus aplicaciones. JFreeChart admite gráficos circulares (2D y 3D), gráficos de barras (horizontales y verticales, regulares y apilados), gráficos de líneas, diagramas de dispersión, diagramas de series de tiempo, diagramas alto-bajo-abierto-cerrado, diagramas de velas, diagramas

combinados, diagramas de Gantt, termómetros, diales y más. JFreeChart se puede usar en aplicaciones del lado del cliente y del lado del servidor. Este proyecto es mantenido por David Gilbert. El extenso conjunto de características de JFreeChart incluye (Viklund , 2017):

- Una API consistente y bien documentada que admite una amplia gama de tipos de gráficos
- Un diseño flexible que es fácil de extender, y se dirige tanto a aplicaciones del lado del servidor como del lado del cliente
- Soporte para muchos tipos de salida, incluidos los componentes Swing y JavaFX, archivos de imagen (incluidos PNG y JPEG) y formatos de archivos de gráficos vectoriales (incluidos PDF, EPS y SVG);
- JFreeChart es de software libre. Se distribuye bajo los términos de la Licencia Pública General Reducida (LGPL) de GNU, que permite el uso en aplicaciones propietarias.

JAMA:

JAMA es un paquete de álgebra lineal básica para Java. Proporciona clases de nivel de usuario para construir y manipular matrices reales y densas. Está destinado a proporcionar suficiente funcionalidad para problemas de rutina, empaquetados de forma natural y comprensible para los no expertos. La librería actual solo trata con matrices reales.

JAMA está compuesta por las clases:

- Matrix
- CholeskyDecomposition
- LUDecomposition
- QRDecomposition
- SingularValueDecomposition
- EigenvalueDecomposition.

Commons Math

Commons Math es una biblioteca de componentes livianos y autónomos de matemáticas y estadística que abordan los problemas más comunes que no están disponibles en el lenguaje de programación Java o Commons Lang (The Apache Software Foundation, 2016).

Características de Commons Math (The Apache Software Foundation, 2016):

- Este paquete enfatiza componentes pequeños y fácilmente integrados en lugar de bibliotecas grandes con dependencias y configuraciones complejas.
- Todos los algoritmos están completamente documentados y siguen las mejores prácticas generalmente aceptadas.
- En situaciones donde existen algoritmos estándar múltiples, se usa un patrón de Estrategia para admitir implementaciones múltiples.
- Dependencias limitadas Sin dependencias externas más allá de los componentes de Commons y la plataforma central de Java (al menos Java 1.3 hasta la versión 1.2 de la biblioteca, al menos Java 5 que comienza con la versión 2.0 de la biblioteca).

1.4. Conclusiones del capítulo

- Se utilizará el formato ARFF para generar los conjuntos de datos en la propuesta de solución ya que es fácil de crear y además es uno de los formatos más usados por las herramientas que contienen algoritmos de aprendizaje automático.
- La herramienta MULAN no posee un generador de datos para problemas de salidas múltiples que permita evaluar todas las opciones. Por ejemplo, variables con ruido, relaciones entre las salidas, entre otras.
- La descripción de la herramienta MULAN permitió definir lenguajes de programación y librerías compatibles a usar con la versión identificada de la misma en el desarrollo de la propuesta de solución.
- De los algoritmos y modelos que generan datos considerados en el estado del arte se tuvo en cuenta el proceso que realiza el algoritmo para regresiones múltiples por lo que se propone un nuevo modelo descrito en el próximo capítulo.

CAPÍTULO 2: Herramienta informática para la generación de datos sintéticos

En el presente capítulo se describen algoritmos usados en la implementación de la solución y donde se incluirán dentro de la herramienta MULAN. También se caracterizan los patrones de diseño utilizados en la solución. Así como los estándares de codificación usados en la misma.

2.1. Propuesta de solución

Se propone implementar un algoritmo similar al algoritmo para generar datos para regresiones múltiples, que realice una generación de datos sintéticos distribuidos normalmente. Para ello se tiene en cuenta el ruido en las variables de salida, la dependencia entre las entradas y las salidas, y la dependencia entre las salidas. La expresión utilizada para ello es:

$$y = (\mathrm{x} * W \ + \ b)\Omega$$

Primeramente, se genera una matriz **X** de tamaño n x p, sus datos siguen una distribución normal de media 0 y varianza 1; luego se genera una matriz **MCI** de tamaño p x p, los datos son generados de manera aleatoria entre 0 y 1; después se calcula una matriz **S** al hallar la inversa de **MCI**; También se genera una matriz **V** de tamaño p x q, sus datos siguen una distribución normal de media 0 y varianza 1/p; luego se calcula una matriz **W** a través de la expresión.

$$W = \mathrm{V} * S$$

Esta matriz se encarga de la correlación entre las variables de entradas y salidas; luego se genera una matriz **Bias** de tamaño n x q, sus datos siguen una distribución normal de media 0 y varianza 1; a continuación, se genera una matriz **Omega** de tamaño q x q, sus datos siguen una distribución normal de media 0 y varianza 1, además esta matriz se encarga de correlacionar las variables de salidas e insertar el ruido. Para mejor entendimiento véase la imagen a continuación.

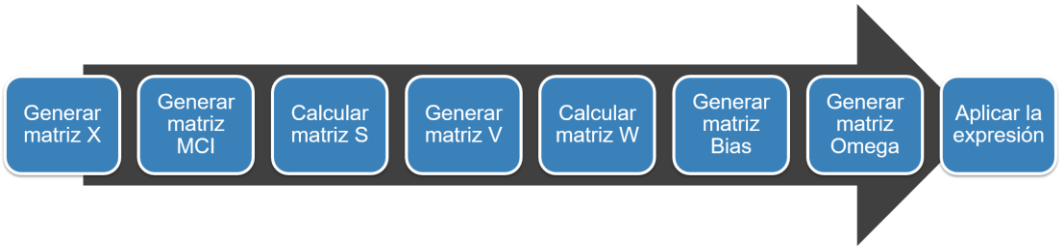


Figura 4 Proceso de generación del conjunto de datos

Como la mayoría de los datos generados siguen una distribución normal también conocida como distribución gaussiana se puede calcular la probabilidad de que varios valores ocurran dentro de ciertos rangos o intervalos. Sin embargo, la probabilidad exacta de un valor particular dentro de una distribución continua, como la distribución normal, es cero. Se llama distribución normal a aquella que queda perfectamente descrita por su media aritmética σ y su desviación típica s. Las distribuciones normales, se denotan por la expresión N (σ, s). La expresión para calcular la densidad de la distribución normal se muestra a continuación (hiru, 2018).

$$f(x) = \frac{e^{-\frac{(x-\bar{x})^2}{2\sigma^2}}}{\sigma\sqrt{2\pi}}$$

Figura 5 Expresión de la función de densidad de la distribución normal

El cálculo de las probabilidades asociadas a una distribución normal por medio de integrales resulta, en general, complejo. Por ello, se utiliza una función de distribución de apoyo cuya media es 0 y cuya desviación típica es la unidad. Tal función se denomina distribución normal tipificada, y se expresa como N (0,1). Se llama tipificación a la operación consistente en cambiar de una variable aleatoria X a otra variable Z de distribución tipificada, por medio de la expresión siguiente (hiru, 2018):

$$Z = \frac{X - \bar{x}}{\sigma}$$

Figura 6 Expresión de la distribución normal tipificada

Es importante que el algoritmo a implementar tenga en cuenta la dependencia entre las variables de entrada y salida porque no existe ningún modelo que pueda predecir las variables de salida si no tiene esta dependencia. Además, si no se tiene en cuenta la dependencia entre las salidas simplemente se genera un modelo para cada salida y se estaría trabajando con los problemas de clasificación clásicos y no con los MTR.

Para la creación de los datos sintéticos solo se necesita la cantidad de variables de entradas, la cantidad de variables de salida, la cantidad de instancias a generar y un porcentaje de dispersión que se utiliza en la

matriz que crea la correlación entre las variables de salida. Se generaron los siguientes métodos para la composición del algoritmo:

- generateMatrixX()
- generateMatrixV()
- generateMatrixCovarianceInverse()
- generateMatrixS(Matrix mci)
- generateMatrixW(Matrix S, Matrix V)
- generateMatrixOmega()
- generateMatrixBias()
- generateOutput(Matrix X, Matrix W, Matrix B, Matrix omega)
- GenerateDataOutput()

Este algoritmo se incluyó en un paquete llamado **mulan.syntetic.data.generator** dentro de la herramienta MULAN conservando la estructura modular y por paquetes de la misma e implementó la interfaz **DataGenerator**. Véase en la siguiente figura. Para ejecutar este algoritmo hay que crear un objeto de la clase **DataSynteticGeneratorMTR** el cual contiene todos los métodos mencionados anteriormente. Además, el uso de la librería JAMA en la implementación del algoritmo facilitó el trabajo con matrices.

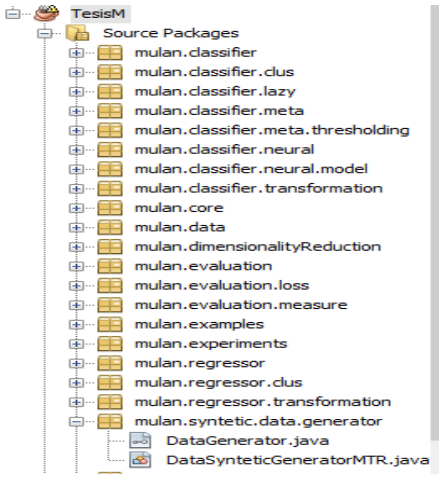


Figura 7 Ubicación de la clase que contiene el algoritmo

Además, se creó un paquete llamado **mulan.view** en el cual se añadió varias clases de interfaz visual. Una para poder recoger los datos necesarios para usar el algoritmo para generar datos sintéticos y otra a

través de la cual se carga el conjunto de datos al cual se le realiza el experimento y también se muestra los resultados del mismo mediante un gráfico. Para crear este gráfico se usó la librería JFreeChart. A continuación, se muestra una imagen con la relación entre los paquetes y algunas clases contenidas en los mismos.

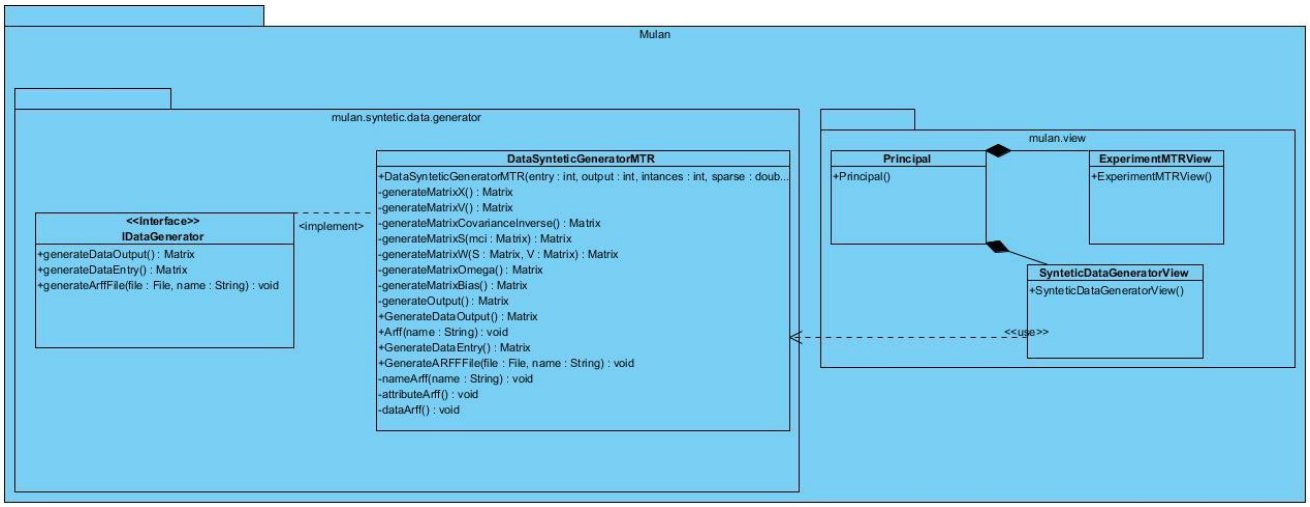


Figura 8 Relación entre paquetes y clases

2.2. Algoritmos usados en la propuesta de solución

Cualquier problema de computación puede resolverse ejecutando una serie de acciones en un orden específico. Un procedimiento para resolver un problema en términos de las acciones a ejecutar y el orden en el que se ejecutan estas acciones se conoce como un algoritmo (DEITEL et al., 2008). A continuación, los métodos definidos por el algoritmo están descritos.

2.2.1. generateMatrixX()

Este método genera una matriz con los datos de las variables de entradas de tamaño intances x entry. Estos datos son generados siguiendo una distribución normal de media 0 y covarianza 1. Para ello se creó un objeto de tipo NormalDistribution llamado ndi, este objeto se encargará de que los datos generados sigan la distribución comentada anteriormente. Luego se crea un objeto de tipo Matrix llamado c de tamaño intances x entry que contendrá los datos generados. Después se recorre esta matriz insertando los datos generados por ndi a través del método sample(). Por último, se devuelve la variable c. A continuación, se muestra una imagen para mayor entendimiento.

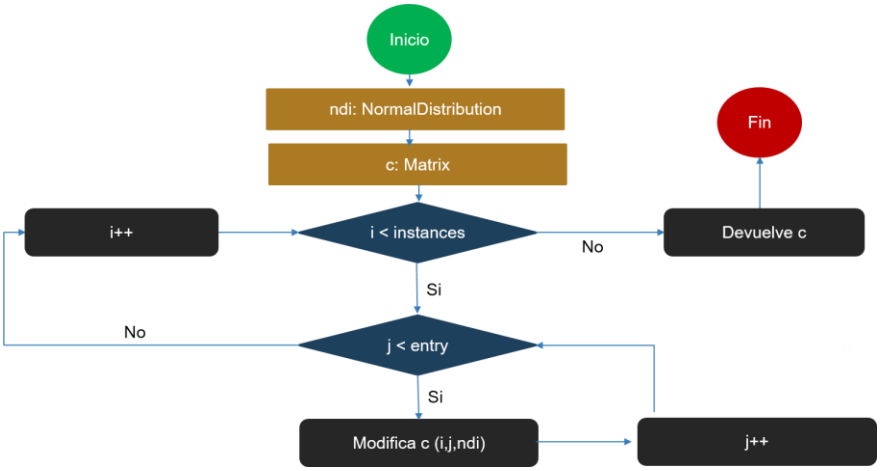


Figura 9 Método generateMatrixX()

2.2.2. generateMatrixV()

Este método genera una matriz de tamaño entry x output que sigue una distribución normal de media 0 y covarianza $\frac{1}{entry}$. Para ello se creó un objeto de tipo NormalDistribution llamado ndi pasándole por parámetro la media 0 y la desviación estándar $(\frac{1}{entry})^2$. Este objeto se encargará de generar datos con la distribución antes mencionada. Luego se declara un objeto de tipo Matrix llamado c que almacenara los datos generados. Después se recorre la matriz creada y se insertan en cada posición los datos generados por ndi a través del método sample(). Por último, se devuelve la variable c. A continuación, se muestra una imagen para mayor comprensión.

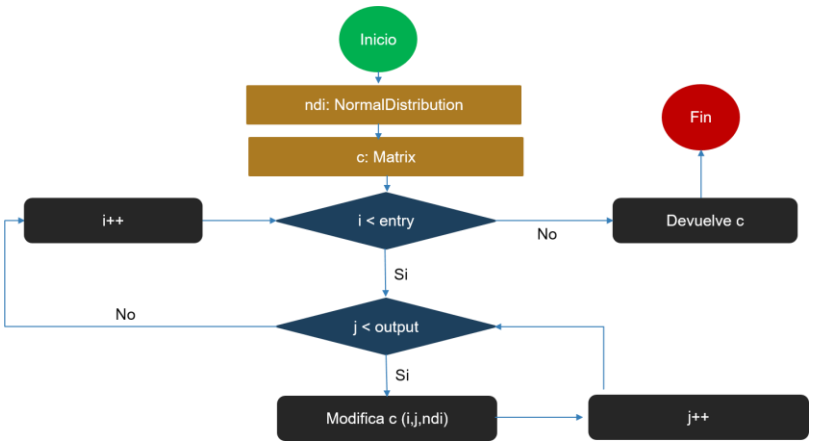


Figura 10 Método generateMatrixV()

2.2.3. generateMatrixCovarianceInverse()

Este método genera una matriz de covarianza inversa de manera aleatoria entre 0 y 1 de tamaño entry x entry. Para ello se declara e inicializa la variable r de tipo Random. La cual generara los valores aleatorios. Luego se crea la variable c que devolverá el método de tipo Matrix pasándole como parámetros la variable entry declarada como atributo de la clase. Después se itera la matriz añadiéndole en cada posición el valor generado de la variable r a través del método nextDouble() si las variables i y j son distintas, estas variables son de tipo entero y a su vez son las que controlan los ciclos. En caso que sean iguales se inserta en esa posición el valor 1. Por último, se devuelve la variable c. A continuación, se muestra una figura para mayor entendimiento.

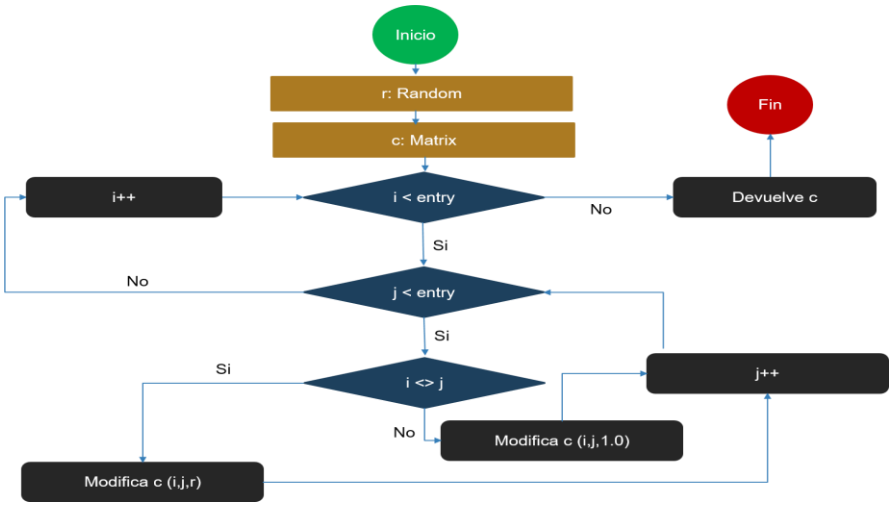


Figura 11 Método generateMatrixCovarianceInverse()

2.2.4. generateMatrixS(Matrix mci)

Este método genera una matriz de las mismas dimensiones que la matriz de covarianza inversa pasada por parámetro. Para ello se crea una variable s de tipo Matrix que será la variable que se devuelva al final del método pasándole por parámetros las dimensiones de la matriz mci pasada por parámetro a través de los métodos getRowDimension() y getColumnDimension(). Luego se recorre la matriz s y se inserta en cada posición recorrida el valor obtenido de hallar la raíz cuadrada a 1 entre cada valor de la matriz de covarianza inversa pasada por parámetro en la misma posición. Por último, se devuelve s. Para mayor comprensión se muestra la siguiente imagen.

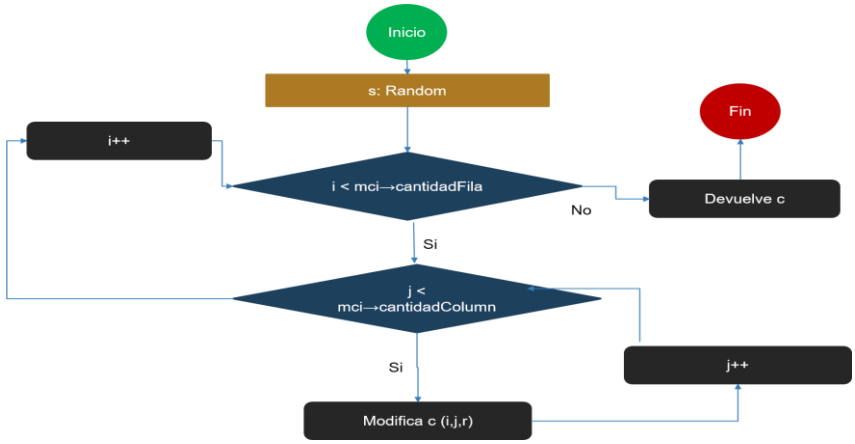


Figura 12 Método generateMatrixS()

2.2.5. generateMatrixW(Matrix S, Matrix V)

Este método genera una matriz de tamaño entry x output. Para ello crea una variable c de tipo Matrix en la cual se almacena el resultado de una multiplicación de la matriz s y v pasadas por parámetros a través del método de la clase Matrix, times(), el cual se le pasa una matriz por parámetro. Por último, se devuelve c. A continuación, se muestra una imagen para mayor entendimiento.

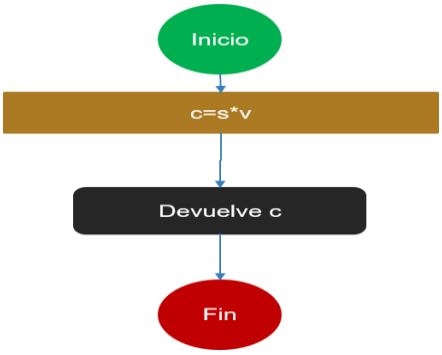


Figura 13 Método generateMatrixW(Matrix S, Matrix V)

2.2.6. generateMatrixOmega()

El método genera una matriz con un porciento de dispersión dado de tamaño output x output siguiendo una distribución normal de media 0 y covarianza 1. Para ello se crea el objeto ndi e tipo NormalDistribution. Luego se crea el objeto de tipo Matrix que se devolverá llamado o de tamaño output x output. Después se recorre la matriz añadiendo en cada posición de la diagonal de la matriz un valor generado que sigue la

distribución normal a través del método sample() del objeto ndi. Si la posición no pertenece a la diagonal de la matriz entonces se añade 0. Luego se crea una variable de tipo Random llamada r. Luego se declara y calcula la variable totalp de la siguiente forma output x output x sparse / 100 y después se redondea dicho número y se almacena el resultado en la variable total. Luego se itera hasta la variable total y se generan dos valores aleatorios entre 0 y output con la variable r a través del método nextInt(). Para que sean las posiciones donde se insertará el elemento según el porciento de esparcidad. Si estos valores generados son iguales y señalan a una posición que contiene un valor distinto de 0 entonces se decrementa la variable del ciclo i, si no, se añade en la matriz en la posición generada un valor siguiendo una distribución normal generado por ndi a través del método sample(). Por último, se devuelve la variable o. Véase las siguientes figuras.

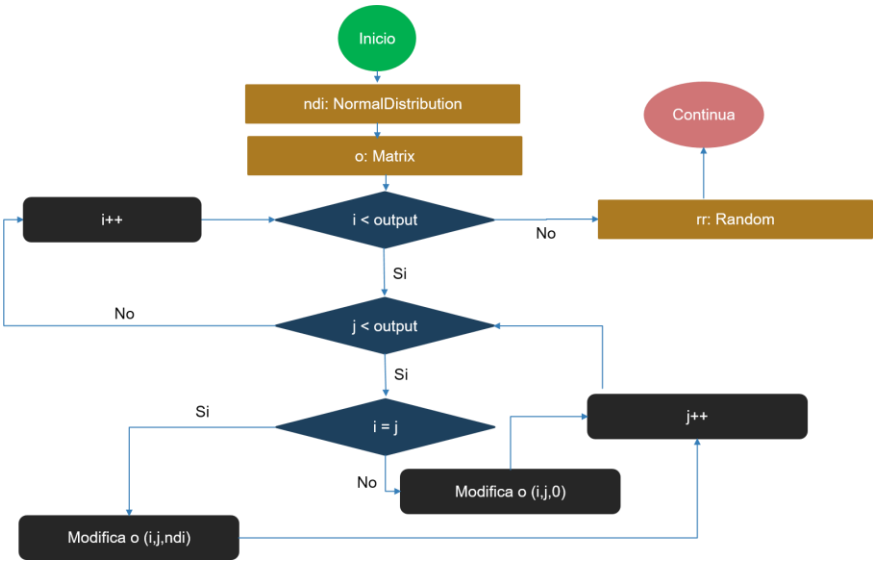


Figura 14 Método generateMatrixOmega() parte 1

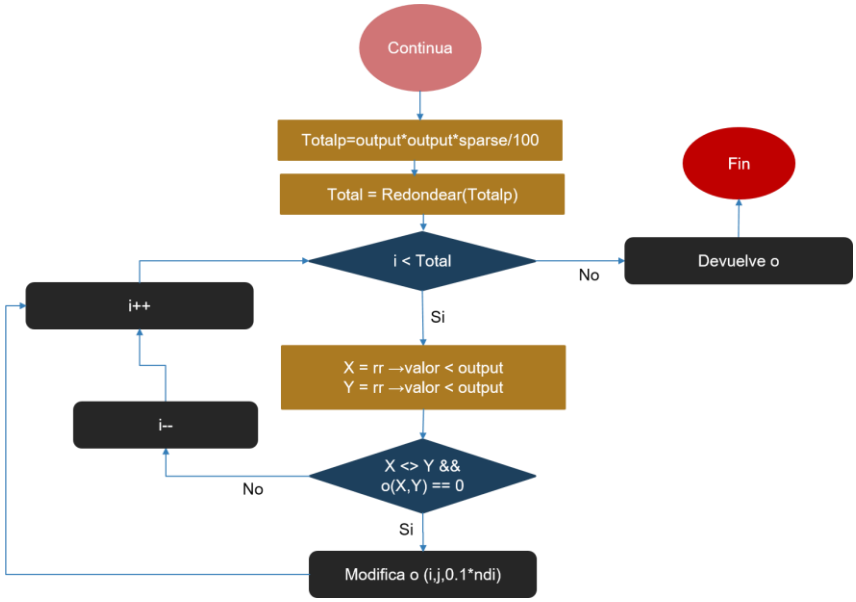


Figura 15 Método generateMatrixOmega() parte 2

2.2.7. generateMatrixBias()

Este método genera una matriz siguiendo una distribución normal de tamaño intances x intances. Para ello se declara un objeto llamado ndi de tipo NormalDistribution. Luego se crea un objeto de tipo Matrix llamado c de tamaño antes mencionado. Luego se itera la matriz y se añade en cada posición el numero generado aleatoriamente siguiendo una distribución normal con el método sample() del objeto ndi. Por último, se devuelve la variable c. Véase la siguiente figura.

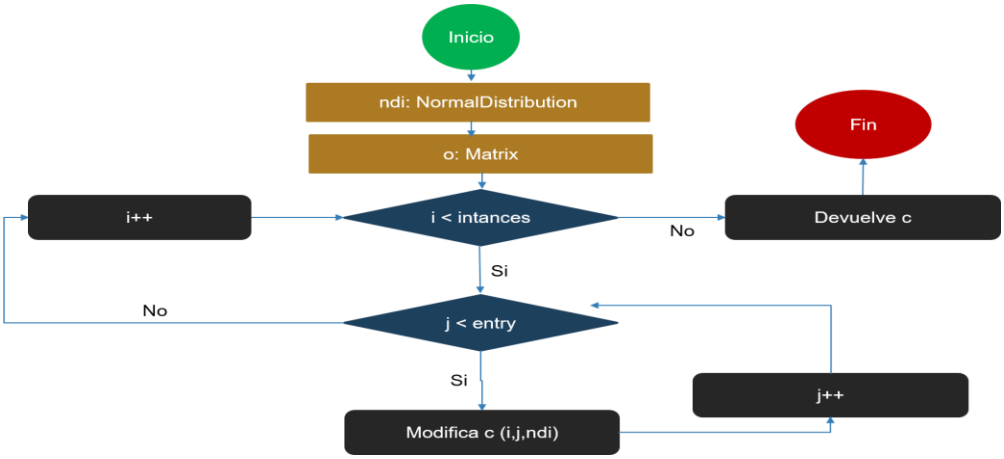


Figura 16 Método generateMatrixBias

2.2.8. generateOutput(Matrix X, Matrix W, Matrix B, Matrix omega)

En este método devuelve las variables objetivo a través de las matrices pasadas por parámetro y la siguiente expresión. Se multiplica la matriz X por la matriz W de esta forma se crea la correlación entre las variables de entradas y las de salida. Luego se suma la matriz B al resultado de la multiplicación. Después se multiplica a ese resultado la matriz omega asegurando la correlación entre las variables de salida. Esa expresión se almacena en la variable y que es una matriz de tamaño intances x output. Por último, se devuelve la variable y. Véase la siguiente figura.

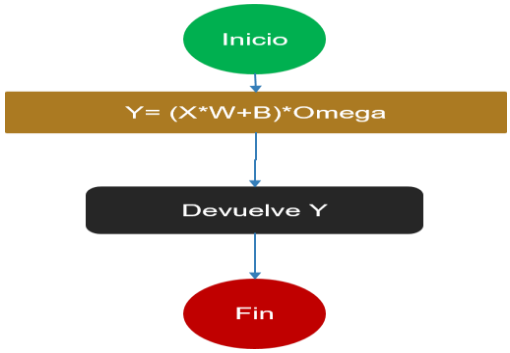


Figura 17 Método generateOutput(Matrix X, Matrix W, Matrix B, Matrix omega)

2.2.9. GenerateDataOutput()

En este método se llama al método generateMatrixX() y se almacena en la variable X si X es igual a null. Luego se usa el método generateMatrixV() y se almacena en la variable V. Después se utiliza el método

generateMatrixCovarianceInverse() y se almacena en la variable mci. A continuación se llama al método generateMatrixS(Matrix mci) el cual se le pasa la variable mci y se almacena el resultado en la variable S. Luego se llama al método generateMatrixW(Matrix S, Matrix V) y se le pasa por parámetro las variables ya calculadas S y V. Se guarda el resultado en la variable W. Después se almacena en la variable B el resultado de invocar al método generateMatrixBias(). A continuación se almacena en la variable omega el resultado de llamar al método generateMatrixOmega(). Después se utiliza el método generateOutput(Matriz X, Matriz W, Matriz B, Matriz omega) y se almacena en la variable y el resultado. Por último, se devuelve la variable y. Véase la siguiente figura para mayor entendimiento.

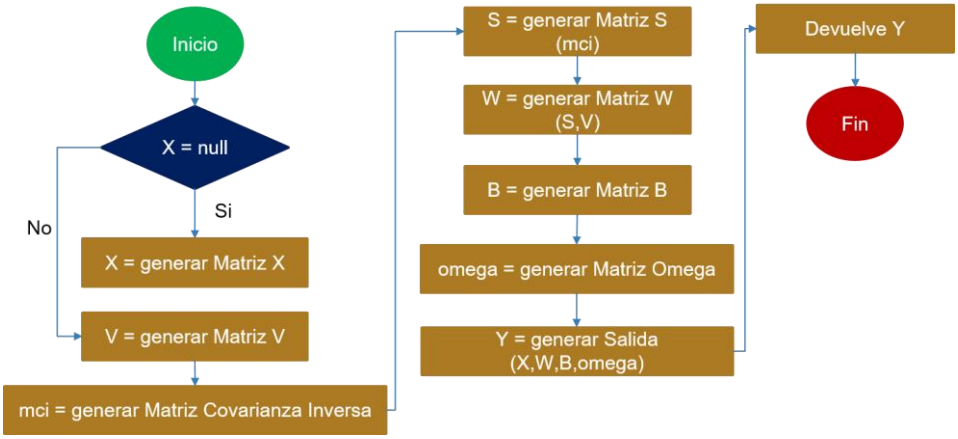


Figura 18 Método GenerateDataOutput()

2.3. Patrones de diseños utilizados

Un patrón de diseño resulta ser una solución a un problema de diseño. Para que una solución sea considerada un patrón debe poseer ciertas características. Una de ellas es que debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores. Otra es que debe ser reutilizable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias (Gamma et al., 2015). En la herramienta a implementar se evidencian algunos patrones de diseño *General Responsibility Assignment Software Patterns* (GRASP) y *Gang of Four* (GoF). En los sub-epígrafes siguientes se presenta dicha evidencia.

2.3.1. Experto

Experto es un patrón que suele utilizarse en el diseño orientado a objetos. Este patrón sugiere asignar

la responsabilidad al objeto que posea la información necesaria para desempeñarla. Con su utilización se conserva el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide. El comportamiento se distribuye entre las clases que cuentan con la información requerida, alentando con ello definiciones de clases sencillas y más cohesivas que son más fáciles de comprender y mantener. Se evidencia en la clase DataSynteticGeneratorMTR, que se responsabiliza de crear un objeto Random llamado r, el cual se encargará de generar números aleatoriamente para su posterior uso. Véase en la siguiente imagen.

```
78          Random r = new Random();
79          Matrix c = new Matrix(entry, entry);
80          for (int i = 0; i < entry; i++) {
81              for (int j = 0; j < entry; j++) {
82
83                  if (i != j) {
84                      c.set(i, j, r.nextDouble());
85                  } else {
86                      c.set(i, j, 1.0);
87                  }
88              }
89          }
```

Figura 19 Ejemplo del patrón de diseño experto

2.3.2. Creador

El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos. El propósito fundamental de este patrón es encontrar un creador que se debe conectar con el objeto producido en cualquier evento. Brinda un soporte a un bajo acoplamiento, lo que supone menos dependencias respecto al mantenimiento y mejores oportunidades de reutilización. Está evidenciado en la clase DataSynteticGeneratorMTR que crea la instancia de la clase NormalDistribution. Véase la siguiente imagen.

```
54          NormalDistribution ndi = new NormalDistribution(0, Math.pow(1.0 / entry, 2));
```

Figura 20 Ejemplo del patrón de diseño creador

2.3.3. Alta cohesión

La información que almacena una clase debe de ser coherente y debe estar, en la medida de lo posible, relacionada con la clase. Esto se evidencia en la clase DataSynteticGeneratorMTR que solo contiene información del algoritmo para generar datos. Véase la siguiente imagen.

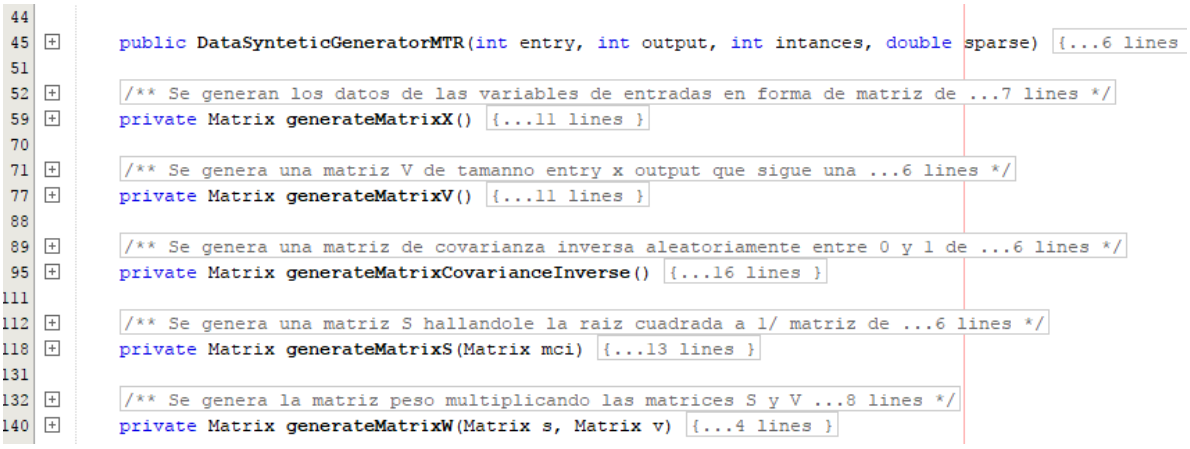


Figura 21 Ejemplo del patrón de diseño alta cohesión

2.3.4. Bajo acoplamiento

Este patrón consiste en tener las clases lo menos ligadas entre sí que se pueda, de tal forma que, en caso de producirse una modificación en alguna de ellas, tenga la mínima repercusión posible en el resto de ellas, potenciando la reutilización y disminuyendo la dependencia entre clases. Esto se evidencia en la clase DataSynteticGeneratorMTR que no se comunica con ninguna clase y solo implementa una interfaz. Véase la siguiente imagen.

```
23  /**...4 lines */
27  public class DataSynteticGeneratorMTR implements IDataGenerator {
28
29      private JTextArea console;
30
        private int entry;
        private int output;
        private int intances;
        private double sparse;
35      private Matrix x;
36      private Matrix v;
37      private Matrix mci;
38      private Matrix s;
39      private Matrix w;
40      private Matrix b;
41      private Matrix omega;
42      private Matrix y;
43
44      public DataSynteticGeneratorMTR(int entry, int output, int intances, double sparse)
```

Figura 22 Ejemplo del patrón de diseño bajo acoplamiento

2.3.5. Abstract factory

Este patrón consiste en proporcionar una interfaz para crear familias de objetos relacionados o dependientes sin especificar sus clases concretas. Esto se evidencia en la clase en la clase ExperimentMTR donde se crea un objeto de tipo Classifier llamado stLearner el cual se le asigna la variable svr de tipo SMOreg la cual hereda de Classifier. Véase la siguiente imagen

```
else if(stLearnerName.equals("svr")){
    SMOreg svr = new SMOreg();

    stLearner = svr;
}
```

Figura 23 Ejemplo del patrón de diseño Abstract factory

2.3.6. Template method

El patrón Template method es una definición abstracta de un algoritmo. Define el algoritmo paso a paso. Cada paso invoca una operación abstracta o una operación primitiva. Una subclase desarrolla el algoritmo definiendo las operaciones abstractas. Esto se evidencia en la clase DataSynteticGeneratorMTR que implementa de la interfaz DataGenerator implementando algunos algoritmos definidos por la misma. Véase la siguiente figura

```
170     @Override
        public Matrix GenerateDataOutput(){
172
173         X = generateMatrixX();
174         V = generateMatrixV();
175         mci = generateMatrixCovarianceInverse();
176         S = generateMatrixS(mci);
177         W = generateMatrixW(S, V);
178         B = generateMatrixBias();
179         omega = generateMatrixOmega();
180         y = generateOutput(X, W, B, omega);
181         return y;
182     }
```

Figura 24 Ejemplo del patrón de diseño Template method

2.4. Estándares de codificación

Un estándar de código es un conjunto de convenciones establecidas de ante mano para la escritura de código. Estos estándares varían dependiendo del lenguaje de programación elegido y además varían en cobertura (Ohmyroot, 2017).

2.4.1. Nomenclatura de identificadores

Las convenciones de nombres de identificadores permiten que los programas sean más fáciles de leer y por tanto más comprensibles. También proporcionan información sobre la función que desempeña el identificador dentro del código, es decir, si es una constante, una variable, una clase o un paquete, entre otros (Flower, 2010).

Paquetes

Se escribirán siempre en letras minúsculas para evitar que entren en conflicto con los nombres de clases o interfaces. El prefijo del paquete siempre corresponderá al nombre mulan. El resto de componentes del paquete se nombrarán de acuerdo a la función por la cual fue creado ese paquete. Véase en la siguiente figura (Flower, 2010):

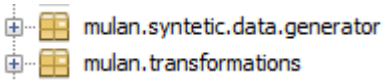


Figura 25 Ejemplo de la nomenclatura de los paquetes

Clases e interfaces

Los nombres de clases deben ser sustantivos y deben tener la primera letra en mayúsculas. Si el nombre es compuesto, cada palabra componente deberá comenzar con mayúsculas. Los nombres serán simples y descriptivos. Debe evitarse el uso de acrónimos o abreviaturas, salvo en aquellos casos en los que dicha abreviatura sea más utilizada que la palabra que representa (URL, HTTP, etc.) (Flower, 2010).

Las interfaces se nombrarán siguiendo los mismos criterios que los indicados para las clases. Como norma general toda interfaz se nombrará con el prefijo "I" para diferenciarla de la clase que la implementa (que tendrá el mismo nombre sin el prefijo "I") (Flower, 2010). Véase las siguientes imágenes.

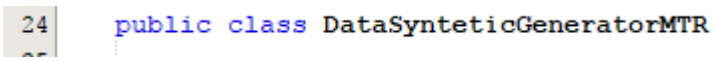


Figura 26 Ejemplo de nomenclatura de una clase

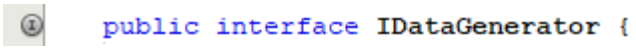


Figura 27 Ejemplo de nomenclatura de una interfaz

Métodos

Los métodos deben ser verbos escritos en minúsculas. Cuando el método esté compuesto por varias palabras cada una de ellas tendrá la primera letra en mayúsculas (Flower, 2010). Véase las imágenes a continuación:

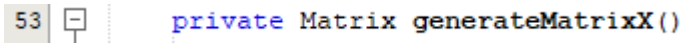


Figura 28 Ejemplo 1 de la nomenclatura de los métodos

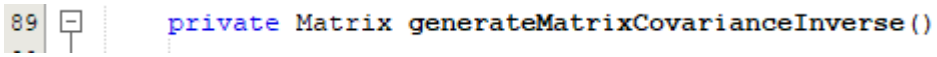


Figura 29 Ejemplo 2 de la nomenclatura de los métodos

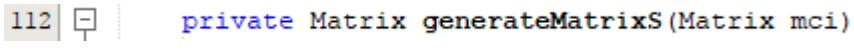


Figura 30 Ejemplo 3 de la nomenclatura de los métodos

Variables

Las variables se escribirán siempre en minúsculas. Las variables compuestas tendrán la primera letra de cada palabra componente en mayúsculas. Ellas nunca podrán comenzar con el carácter "_" o "$". Los

nombres de variables deben ser cortos y sus significados tienen que expresar con suficiente claridad la función que desempeñan en el código. Debe evitarse el uso de nombres de variables con un sólo carácter, excepto para variables temporales (Flower, 2010). Véase la imagen a continuación.

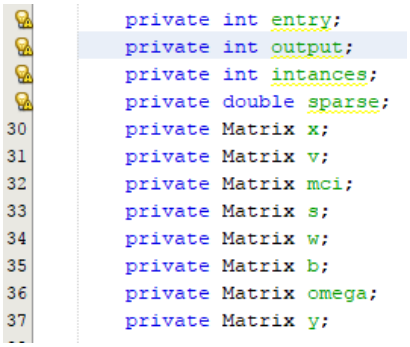


Figura 31 Ejemplos de la nomenclatura de una variable

2.4.2. Longitud de línea

La longitud de línea no debe superar los 80 caracteres por motivos de visualización e impresión (Flower, 2010).

División de líneas

Cuando una expresión ocupe más de una línea, esta se podrá romper o dividir en función de los siguientes criterios (Flower, 2010):

- Tras una coma
- Antes de un operador
- Se recomienda las rupturas de nivel superior a las de nivel inferior
- Alinear la nueva línea con el inicio de la expresión al mismo nivel que la línea anterior
- Si las reglas anteriores generan código poco comprensible, entonces estableceremos tabulaciones de 8 espacios

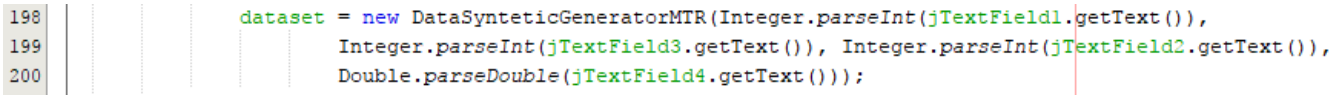


Figura 32 Ejemplo de división de líneas

2.4.3. Comentarios

Se distingue dos tipos de comentarios: los comentarios de implementación y los de documentación.

Comentarios de implementación

Estos comentarios se utilizan para describir el código ("el cómo"), y en ellos se incluye información relacionada con la implementación, tales como descripción de la función de variables locales, fases lógicas de ejecución de un método, captura de excepciones (Flower, 2010).

Hay tres tipos de comentarios de implementación:

- Comentarios de bloque:
  - Permiten la descripción de ficheros, clases, bloques, estructuras de datos y algoritmos.

```
46      /**
47       * Se generan los datos de las variables de entradas en forma de matriz de
48       * tamanno intances x entry, estos datos son generados siguiendo una
49       * distribucion normal
50       *
51       * @return
52       */
```

Figura 33 Ejemplo de comentarios de bloque

- Comentarios de línea:
  - Son comentarios cortos localizados en una sola línea y tabulados al mismo nivel que el código que describen. Si ocupa más de una línea se utilizará un comentario de bloque. Deben estar precedidos por una línea en blanco.

```
382        // Se crea la grafica con todos sus componentes;
```

Figura 34 Ejemplo de comentario de línea

- Comentario al final de línea
  - Comentario situado al final de una sentencia de código y en la misma línea.

```
204            if(jTextField5.getText().matches("[A-Za-z0-9]*")){// Se Valida el campo
```

Figura 35 Ejemplo de comentario al final de una sentencia de código

Comentarios de documentación

Los comentarios de documentación, también denominados "comentarios javadoc", se utilizan para describir la especificación del código, desde un punto de vista independiente de la implementación, de forma que pueda ser consultada por desarrolladores que probablemente no tengan acceso al código fuente (Flower, 2010).

2.5. Conclusiones del capítulo

- El desarrollo de los métodos propuestos permitió la implementación del algoritmo de solución. Obteniendo así una herramienta capaz de crear conjuntos de datos para la experimentación con problemas Multi-Target Regression.
- Con la implementación de algunos patrones GRASP y GoF se aseguran las buenas prácticas recomendadas para el diseño de la propuesta de solución.
- Con el uso de los estándares de codificación se logró una mejor homogeneidad en el código para mejor entendimiento por parte de los programadores en futuras implementaciones o modificaciones de la solución.

CAPÍTULO 3: Validación de la herramienta informática para la creación de datos sintéticos.

A continuación, se describen las distintas pruebas usadas en la investigación, además se mostrarán los resultados obtenidos al comparar cómo se comportan los algoritmos contenidos en la herramienta MULAN que resuelven problemas Multi-Target Regression con respecto a los distintos conjuntos de datos generados a través de la propuesta de solución. Además, se mostrarán los resultados obtenidos tras aplicar la prueba de Friedman.

3.1. Resultados

A continuación, se mostrará una tabla donde se aprecia las distintas características con las que fueron creados los conjuntos de datos sintéticos con la herramienta propuesta. Estas características comprenden nombre del conjunto de datos, cantidad de entradas, cantidad de salidas, porciento de esparcidad, instancias.

Tabla 6 Características de los data set generados

Conjunto de datos	Cantidad de entradas	Cantidad de salidas	% de esparcidad	Instancias
dataset1	20	5	5	100
dataset2	20	5	10	100
dataset3	20	5	15	100
dataset4	500	5	5	100
dataset5	500	5	10	100
dataset6	500	5	15	100
dataset7	1000	5	5	100
dataset8	1000	5	10	100
dataset9	1000	5	15	100
dataset10	20	10	5	100
dataset11	20	10	10	100
dataset12	20	10	15	100
dataset13	500	10	5	100
dataset14	500	10	10	100
dataset15	500	10	15	100
dataset19	20	50	5	100
dataset20	20	50	10	100
dataset21	20	50	15	100
dataset22	500	50	5	100
dataset23	500	50	10	100
dataset24	500	50	15	100
dataset28	5	10	5	1000
dataset29	5	10	10	1000
dataset30	5	10	15	1000

A estos conjuntos de datos se le aplicaron algunos métodos de evaluación de problemas MTR como son Single-Target Regressor (ST), Multi-Target Stacking (MTS), Ensemble of Regressor Chains (ERC), Multi-Objective Random Forest (MORF), Multi-Target Stacking Corrected (MTSC), Ensemble of Regressor Chains Corrected (ERCC). Para ello se utilizó validación cruzada como tipo de evaluación, además se utilizaron como regresores base reptree-bag, KNN y SVR, se iteró una sola vez para cada algoritmo. A continuación, se mostrará una tabla por cada regresor base con los resultados del comportamiento de los algoritmos MTR ante los conjuntos de datos generados con respecto al Average Relative RMSE.

Tabla 7 Comparación entre los algoritmos utilizando reptree-bag como regresor base

Conjuntos de datos	ST	MTS	MTSC	ERC	MORF	ERCC
dataset1	1.017	1.037	1.021	1.014	**1.008**	1.016
dataset2	1.019	1.021	1.019	1.017	**1.012**	1.019
dataset3	1.015	**1.009**	1.011	1.014	1.02	1.012
dataset4	1.008	1.006	1.009	1.007	**1.001**	1.008
dataset5	1.01	1.01	1.01	1.01	**0.995**	1.01
dataset6	1.001	**1.0**	**1.0**	1.001	1.004	**1.0**
dataset7	1.009	1.009	1.008	1.008	**0.999**	1.008
dataset8	**1.003**	1.014	**1.003**	1.006	1.012	**1.003**
dataset9	1.01	1.01	1.011	1.01	**1.005**	1.01
dataset10	1.014	1.046	1.016	1.013	**1.009**	1.014
dataset11	1.008	1.014	1.008	**1.002**	1.01	1.005
dataset12	1.024	1.02	1.024	1.019	**1.014**	1.024
dataset13	1.008	1.008	**1.006**	1.007	1.013	1.007
dataset14	1.004	1.016	**1.003**	1.007	1.004	1.004
dataset15	**1.0**	1.003	1.001	**1.0**	1.003	**1.0**
dataset16	1.017	1.06	1.017	1.02	**1.01**	1.016
dataset17	1.0	1.015	1.0	1.004	**0.999**	1.0
dataset18	1.006	1.006	1.006	1.006	**1.002**	1.006
dataset19	1.011	1.021	1.011	1.007	1.009	**1.006**
dataset20	1.006	1.009	1.006	**1.0**	1.003	1.001
dataset21	1.015	1.031	1.013	**1.007**	1.014	1.009
dataset22	**1.006**	1.015	**1.006**	1.007	1.007	**1.006**
dataset23	**1.008**	1.027	**1.008**	1.01	**1.008**	**1.008**
dataset24	**1.002**	1.006	1.004	**1.002**	1.003	**1.002**
dataset28	0.988	0.987	0.988	0.984	0.996	**0.983**
dataset29	0.989	1.002	0.987	0.984	0.999	**0.982**
dataset30	0.964	0.968	0.962	0.959	0.973	**0.958**

Como se puede observar en la tabla anterior un 4% de los conjuntos de datos alcanzó los mejores resultados con el algoritmo MTS, el 7% con el algoritmo MTSC, el 11% con el ERC, el 44% con el algoritmo MORF y el 15% con el ERCC. Además, el 22% de los mejores resultados fue compartido entre varios algoritmos. Se puede decir que el algoritmo MORF se comportó mejor ante conjuntos de datos con características como las descritas anteriormente usando RepTree-Bag como regresor base.

Tabla 8 Comparación entre algoritmos utilizando KNN como regresor base

Conjuntos de datos	ST	MTS	MTSC	ERC	MORF	ERCC
dataset1	1.02	1.045	1.04	1.025	**1.008**	1.024
dataset2	1.011	1.025	1.023	1.012	1.012	**1.01**
dataset3	1.008	1.018	1.03	**1.004**	1.02	1.01
dataset4	1.011	1.005	1.014	1.009	**1.001**	1.008
dataset5	1.045	1.043	1.043	1.045	**0.995**	1.046
dataset6	1.017	1.016	1.012	1.011	**1.004**	1.012
dataset7	1.026	1.037	1.025	1.028	**0.999**	1.026
dataset8	1.009	1.011	1.008	1.01	1.012	**1.007**
dataset9	1.026	1.023	1.023	1.022	**1.005**	1.02
dataset10	1.02	1.03	1.031	1.018	**1.009**	1.013
dataset11	1.011	1.017	1.009	1.005	1.01	**1.004**
dataset12	1.03	1.037	1.027	1.028	**1.014**	1.023
dataset13	1.028	1.028	1.038	1.029	**1.013**	1.031
dataset14	1.022	1.024	1.024	1.015	**1.004**	1.017
dataset15	1.029	1.028	1.022	1.025	**1.003**	1.022
dataset16	1.029	1.032	1.034	1.027	**1.01**	1.026
dataset17	1.021	1.026	1.031	1.019	**0.999**	1.017
dataset18	1.033	1.03	1.036	1.036	**1.002**	1.03
dataset19	1.026	1.037	1.032	1.017	**1.009**	1.01
dataset20	1.024	1.027	1.023	1.012	**1.003**	1.009
dataset21	1.029	1.034	1.029	1.019	1.014	**1.009**
dataset22	1.036	1.045	1.04	1.033	**1.007**	1.029
dataset23	1.03	1.029	1.031	1.023	**1.008**	1.02
dataset24	1.023	1.024	1.026	1.018	**1.003**	1.016
dataset28	0.996	0.999	0.999	0.988	0.996	**0.987**
dataset29	0.996	0.999	0.999	0.989	0.999	**0.985**
dataset30	0.968	0.972	0.972	0.962	0.973	**0.96**

En la tabla anterior un 26% de los conjuntos de datos alcanzó los mejores resultados con el algoritmo ERCC, un 4% con el ERC y un 70% con el algoritmo MORF. Por lo que se puede concluir que MORF se comportó mejor que los otros algoritmos sobre conjuntos de datos con características como las descritas anteriormente usando como regresor base KNN.

Tabla 9 Comparación entre algoritmos utilizando SVR como regresor base

Conjuntos de datos	ST	MTS	MTSC	ERC	MORF	ERCC
dataset1	1.219	1.223	1.192	1.2	**1.008**	1.184
dataset2	1.086	1.087	1.112	1.086	**1.012**	1.093
dataset3	1.242	1.253	1.271	1.241	**1.02**	1.249
dataset4	1.104	1.104	1.106	1.104	**1.001**	1.105
dataset5	1.135	1.135	1.122	1.135	**0.995**	1.127
dataset6	1.12	1.12	1.12	1.12	**1.004**	1.12
dataset7	1.047	1.047	1.045	1.047	**0.999**	1.046
dataset8	1.059	1.059	1.06	1.059	**1.012**	1.06

dataset9	1.026	1.026	1.027	1.026	**1.005**	1.027
dataset10	1.167	1.176	1.186	1.151	**1.009**	1.151
dataset11	1.144	1.15	1.188	1.131	**1.01**	1.143
dataset12	1.188	1.197	1.229	1.183	**1.014**	1.187
dataset13	1.204	1.204	1.181	1.203	**1.013**	1.189
dataset14	1.115	1.115	1.094	1.115	**1.004**	1.102
dataset15	1.121	1.121	1.095	1.121	**1.003**	1.107
dataset16	1.089	1.089	1.073	1.089	**1.01**	1.081
dataset17	1.044	1.044	1.044	1.044	**0.999**	1.044
dataset18	1.076	1.076	1.071	1.076	**1.002**	1.073
dataset19	1.195	1.223	1.274	1.154	**1.009**	1.159
dataset20	1.177	1.205	1.269	1.14	**1.003**	1.162
dataset21	1.184	1.205	1.267	1.153	**1.014**	1.156
dataset22	1.132	1.132	1.099	1.132	**1.007**	1.102
dataset23	1.127	1.127	1.095	1.127	**1.008**	1.099
dataset24	1.105	1.105	1.086	1.105	**1.003**	1.085
dataset28	**0.978**	**0.978**	**0.978**	**0.978**	0.996	**0.978**
dataset29	0.98	0.98	0.98	0.979	0.999	**0.979**
dataset30	**0.947**	**0.947**	0.948	**0.947**	0.973	0.948

Como se puede apreciar en la tabla anterior un 89% de los conjuntos de datos alcanzó los mejores resultados con el algoritmo MORF, un 4% con el ERCC y un 7% fue compartido entre varios conjuntos. Por esto se concluye que MORF se comportó mejor ante los conjuntos de datos con las características descritas anteriormente y usando el regresor base SVR.

3.2. Metodologías estadísticas para la comparación de algoritmos

La evaluación estadística de resultados experimentales es una parte esencial para la validación de los métodos de aprendizaje automático. Para la comparación entre múltiples modelos sobre varias bases de datos, (Vázquez et al., 2001) y (Pizarro et al., 2002) utilizan la Prueba ANOVA y la Prueba de Friedman

A continuación se muestra un diagrama donde se evidencia qué tipo de metodología escoger en dependencia de los resultados que se quiera obtener y en las características de las bases de datos con las que se debe trabajar. Por ejemplo si la distribución de los datos es normalizada, se cumple la propiedad de esfericidad de los datos y son de baja dimensionalidad, lo recomendable es usar la prueba paramétrica ANOVA, en caso contrario se recomienda la Prueba de Friedman. Ambas pruebas determinan si existen diferencias entre los algoritmos y cuentan con un conjunto de pruebas post-hoc para determinar si estas diferencias son significativas o no.

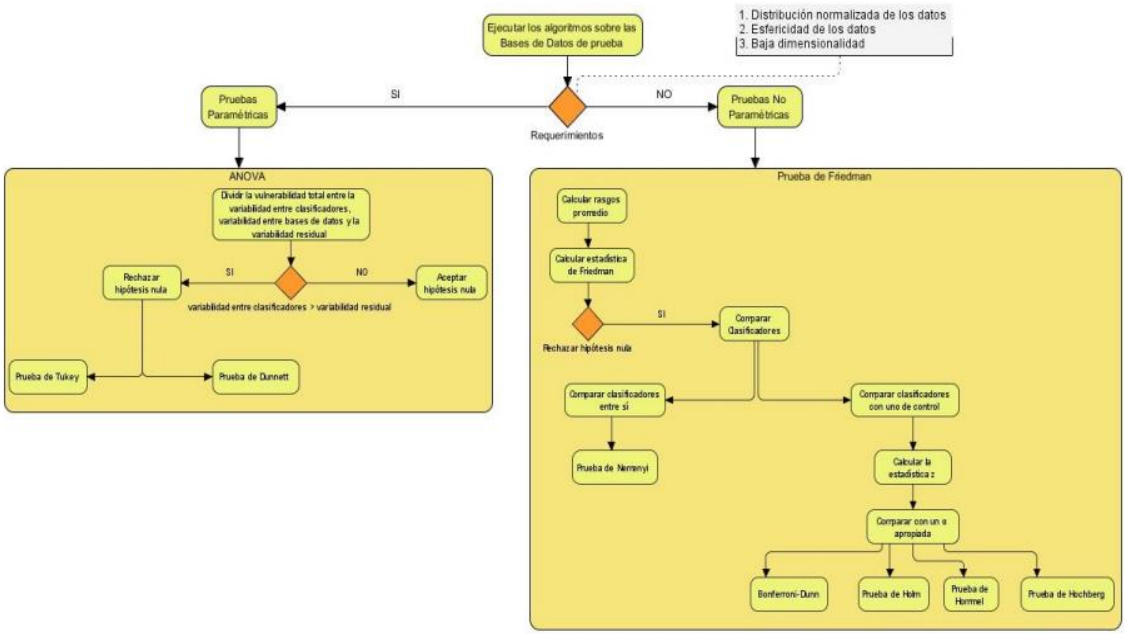


Figura 36 Diagrama de Selección de metodología

3.2.1. Comparación de múltiples algoritmos

El método estadístico común para la comparación de múltiples algoritmos es ANOVA (conocido también como medidas- repetidas) (Fisher, 1959) . Una de las desventajas de ANOVA es que está basada en asunciones, que, al analizar el desempeño de los algoritmos de aprendizaje automático, son posiblemente violadas. Primeramente, ANOVA asume que las muestras son extraídas de distribuciones normalizadas cuando no hay garantía de ello, aunque esto no es un gran problema y muchos estadísticos no se oponen a su uso con la excepción de que las distribuciones fueran, para dar ejemplo, claramente bimodales (Hamilton, 1990). La segunda y más importante asunción de ANOVA es la igualdad de varianza. Debido a la naturaleza de los algoritmos de aprendizaje y las colecciones de datos, es una propiedad que no se puede dar por hecho. Violaciones de estas asunciones tiene un efecto aún mayor en las Pruebas Post-hoc, por lo que en los problemas de aprendizaje automático no se garantiza el cumplimiento de los supuestos paramétricos para aplicar la prueba ANOVA (Demšar, 2006).

Un equivalente no paramétrico de ANOVA es la Prueba de Friedman (Friedman, 1937) (Friedman, 1940), la cual jerarquiza los algoritmos por cada colección de datos separadamente y en caso de empate asigna un rango promedio. Tomando r_i^j como el rango del algoritmo j de k algoritmos en la $i-$ é$sima$ colección de datos de N colecciones de datos, la Prueba de Friedman compara los rasgos promedios de los algoritmos

$R_j = {}^{1}/_{\mathrm{N}} \sum_i {r_i}^j$. En esta, la hipótesis nula afirma que todos los algoritmos son equivalentes y por eso sus rangos R_j deberían ser iguales. Bajo esta hipótesis nula, la estadística de la Prueba de Friedman se calcula:

$$X^2F = \frac{12N}{k(k+1)} \left[\sum_j {R_j}^2 - \frac{k(k+1)^2}{4} \right]$$

y es distribuida de acuerdo a ${X_F}^2$ con $k - 1$ grados de libertad cuando k y N son lo suficientemente grandes ($k > 5$, $N > 10$) y en caso contrario (Sheskin, 2003) calcularon valores críticos exactos. En (Iman et al., 1980) demostraron que la ${X_F}^2$ de Friedman no es lo deseadamente conservativa y derivaron una mejor estadística:

$$F_F = \frac{(N-1){X_F}^2}{N(k-1) - {X_F}^2}$$

la cual es distribuida de acuerdo a la distribución F con $k - 1$ y $(k - 1)(N - 1)$ grados de libertad. Teóricamente la Prueba de Friedman no paramétrica tiene menos poder que el ANOVA paramétrico cuando las asunciones de ANOVA se encuentran, en caso contrario no necesariamente. (Friedman, 1940) experimentalmente los comparó en 56 problemas independientes y demostró que los dos coincidían generalmente. Cuando uno encontraba significancia en $p < 0.01$ el otro la muestra en al menos $p < 0.05$ respectivamente. Solamente en dos casos ANOVA encontró significancia que era insignificante para Friedman, mientras que lo contrario ocurrió en 4 casos. (Demšar, 2006)

Si la hipótesis nula es rechazada se puede proceder con una Prueba de Post-hoc. La Prueba de Nemenyi (Nemenyi, 1963) es similar a la de Tukey para ANOVA y es usada cuando los clasificadores son comparados entre sí. El desempeño de los mismos es significantemente diferente si el rango ordinario correspondiente difiere al menos la diferencia crítica (CD) donde los valores críticos $q\alpha$ están basados en la estadística del rango Studentized dividida por √2. Esta diferencia crítica se calcula de la siguiente manera:

$$CD = {}^{q\alpha}\sqrt{\frac{k(k+1)}{6N}}$$

Cuando todos los clasificadores son comparados con un clasificador de control, se puede hacer uso de algún procedimiento general en lugar de la Prueba de Nemenyi para controlar el error FamilyWise (FWE) en la prueba de múltiples hipótesis como la corrección de Bonferroni u otros similares. El FWE o como también se conoce la inflación alfa o error acumulativo tipo I representa la probabilidad de que una de un conjunto de comparaciones es un error de tipo I. Aunque estos métodos se pueden considerar generalmente

conservativos, en este caso son más efectivos que el de Nemenyi, ya que este ajusta el valor crítico para realizar $\frac{k(k-1)}{2}$ comparaciones mientras que al comparar con un clasificador de control solo se hacen $k - 1$ comparaciones. La estadística para comparar los clasificadores i y j utilizando estos métodos es:

$$z = \frac{(R_i - R_j)}{\sqrt{\frac{k(k+1)}{6N}}}$$

Este valor es usado para encontrar la probabilidad correspondiente para compararla con un α apropiada. Las pruebas difieren en la forma que ajustan el valor de α para compensar por comparaciones múltiples (Demšar, 2006).

El test de Bonferroni-Dunn (Dunn, 1961) controla la tasa del FWE al dividir α por la cantidad de comparaciones realizadas, o calcular la CD usando la ecuación de Nemenyi pero usando los valores críticos para $\frac{\alpha}{(k-1)}$ .Como contraste del test de Bonferroni-Dunn, los procedimientos de paso-adelante y paso-atrás prueban secuencialmente las hipótesis ordenadas por su significancia, denotando los valores ordenados p por $p_1, p_2, \ldots, p_{k-1}$ de manera que $p_1 < p_2 < p_{k-1}$. Ambos comparan cada p_i con $\frac{\alpha}{(k-i)}$ pero difieren en el orden de las pruebas. El procedimiento paso-atrás de Holm (Holm, 1979) comienza con el valor p más significante. Si p_1 es menor que $\frac{\alpha}{(k-1)}$ la hipótesis correspondiente es rechazada y se permite la comparación de p_2 y así sucesivamente. En el momento en que alguna hipótesis nula no pueda ser rechazada, las restantes son retenidas también. El procedimiento de paso-adelante de Hochberg trabaja de manera opuesta, comparando el valor más grande de p con α, el segundo con α/2, y así consecutivamente hasta que encuentre una hipótesis que pueda rechazar. Todas las hipótesis con valores p más pequeños son rechazadas también (Demšar, 2006).

El procedimiento de Hommel (Hommel, 1988) encuentra el valor más grande j para el cual $p_{n-j+k} > \frac{k\alpha}{j}$ para todas las k = 1. Si tal j no existe se rechazan todas las hipótesis, en caso contrario se rechazan todas para las cuales $p_i \leq \frac{\alpha}{j}$. El procedimiento de Holm es más potente que el de Bonferroni-Dunn y no realiza asunciones adicionales sobre la hipótesis probada. La única ventaja del test de Bonferroni-Dunn es que es más fácil de describir y visualizar porque utiliza la misma CD para todas las comparaciones. Los métodos de Hochberg y Hommel rechazan más hipótesis que el de Holm pero en algunas circunstancias exceden el FWE prescrito ya que se basan en las conjeturas de Simes las cuales todavía se encuentran bajo investigación. Las diferencias entre todos estos métodos en la práctica son pequeñas, así que el método más complejo de Hommel no ofrece gran ventaja sobre el más sencillo de Holm (Holland, B., 1991). Aunque

estos procedimientos se proponen como Pruebas Post-hoc para la Prueba de Friedman, también pueden usarse para controlar el FWE cuando múltiples hipótesis, de posiblemente varios tipos, son probadas. Aunque algunas veces la Prueba de Friedman reporta una diferencia significante, las Pruebas Post-hoc fallan en detectarla debido al poco poder que poseen (Demšar, 2006).

3.3. Pruebas de software

La prueba es un conjunto de actividades que pueden planearse por adelantado y realizarse de manera sistemática. Por esta razón, durante el proceso de software, debe definirse una plantilla para la prueba del software: un conjunto de pasos que incluyen métodos de prueba y técnicas de diseño de casos de prueba específicos (ROGER S. PRESSMAN, 2010).

3.3.1. Prueba unitaria

Se realizó una prueba unitaria con el fin de comprobar si a partir de un caso de prueba el método seleccionado devolvía el resultado esperado. Se seleccionó el método generateOutput debido a que los demás generan sus salidas de manera aleatoria o siguiendo una distribución normal por lo que se hace imposible predecir el resultado final. Para la realización de esta prueba se pasó por parámetro una matriz X, una matriz W, una matriz B, una matriz Omega y además, el resultado esperado como se muestra a continuación.

Tabla 10 Matriz X entrada por parámetro

0.100	0.200
0.500	0.600
0.900	0.100

Tabla 11 Matriz W entrada por parámetro

0.100	0.200
0.500	0.400

Tabla 12 Matriz B entrada por parámetro

0.100	0.200
0.500	0.900
0.800	0.100

Tabla 13 Matriz Omega entra por parámetro

0.100	0.200
0.500	0.400

Tabla 14 Resultado esperado

0.171	0.162
0.705	0.666
0.254	0.316

La prueba realizada albergó resultados satisfactorios. Validando así la solución implementada. Estos resultados se muestran a continuación.

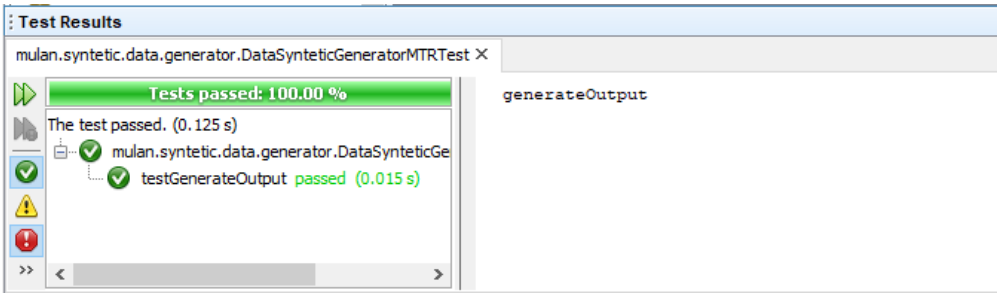


Figura 37 Resultados de la prueba

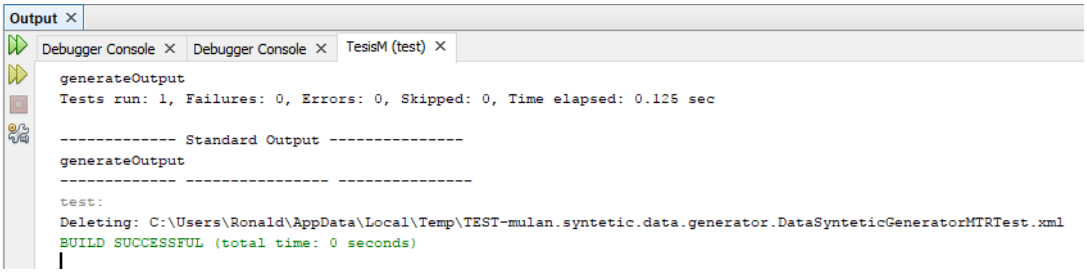


Figura 38 Salida de la prueba

3.3.2. Prueba de tiempo de ejecución

A los conjuntos de datos generados con las características mencionadas en el epígrafe 3.1 se les midió el tiempo en milisegundos de ejecución de los métodos con los cuales fueron creados. Se obtuvo los siguientes resultados mostrados de forma tabular. Estos métodos son:

1. generateMatrixX
2. generateMatrixV
3. generateMatrixCovarianceInverse
4. generateMatrixS
5. generateMatrixW
6. generateMatrixBias
7. generateMatrixOmega

8. generateOutput

Tabla 15 Tiempo de creación de los conjuntos de datos

Conjunto de datos	1	2	3	4	5	6	7	8	Total
dataset1	64	1	1	0	0	0	1	0	67
dataset2	61	1	1	0	0	1	0	0	64
dataset3	61	1	1	0	0	0	0	1	64
dataset4	129	2	49	24	25	1	0	3	233
dataset5	126	1	46	23	27	1	6	3	233
dataset6	109	2	50	24	20	0	1	3	209
dataset7	144	4	93	31	34	1	0	1	308
dataset8	135	4	95	31	36	0	0	2	303
dataset9	165	4	101	33	41	0	1	1	346
dataset10	94	1	1	1	0	1	1	1	100
dataset11	103	1	1	1	0	2	1	2	111
dataset12	86	1	1	0	0	1	0	2	91
dataset13	116	5	37	22	34	1	0	5	220
dataset14	115	3	48	26	28	1	0	3	224
dataset15	110	3	50	23	36	1	0	6	229
dataset16	121	5	97	29	48	0	1	2	303
dataset17	143	6	96	34	40	0	1	2	322
dataset18	143	7	93	32	41	1	0	3	320
dataset19	82	2	1	0	1	3	1	10	100
dataset20	93	2	1	0	1	4	1	14	116
dataset21	82	1	1	0	2	2	2	9	99
dataset22	106	15	46	23	59	2	0	6	257
dataset23	115	17	43	45	56	2	1	6	285
dataset24	114	9	44	21	86	2	0	6	282
dataset28	105	0	1	0	0	6	1	7	120
dataset29	89	0	1	0	0	8	0	9	107
dataset30	91	0	1	0	0	7	1	6	106

Como se puede observar en la tabla, los resultados no exceden el segundo. Por lo que, para ser un proceso de generación de datos, que además otros procesos no dependen de él, son buenos resultados.

3.4. Prueba de Friedman

Para la realización de esta prueba se tomó las tablas 9,10,11 donde se muestra el Average Relative RMSE para determinar si existen diferencias significativas entre los algoritmos. Se obtuvo un $\chi^2 = 5.6222$, $\chi^2 = 14.873$, $\chi^2 = 9.9834$ con un *p_value* = 3.044e-11, *p_value* = 5.945e-08, con un *p_value* = 2.2e-16 respectivamente menor que $\alpha = 0.05$. Esto significa que se rechaza la hipótesis nula, es decir, existen diferencias significativas entre los algoritmos. Finalmente se obtiene un ranking de los algoritmos como se puede observar en las siguientes imágenes.

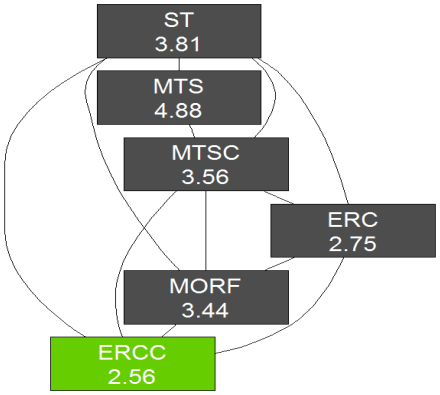


Figura 39 Ranking de Friedman para la tabla 9

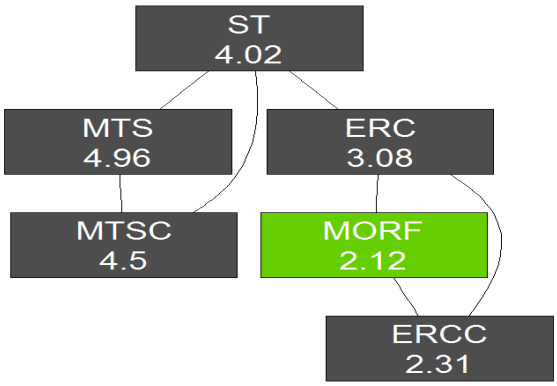


Figura 40 Ranking de Friedman para la tabla 10

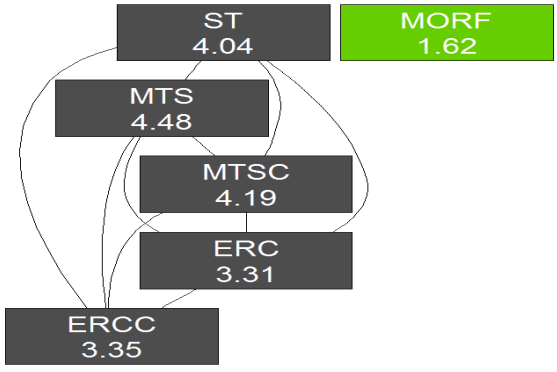


Figura 41 Ranking de Friedman para la tabla 11

Tras obtener los rankings para las distintas tablas se procedió a realizar la prueba de Bonferroni-Dunn, que los compara con uno de control y utiliza la ecuación de diferencia crítica planteada por Nemenyi. Esta prueba post-hoc plantea que el desempeño de los clasificadores es significantemente diferente si el rango ordinario correspondiente difiere al menos la diferencia crítica (CD), obteniéndose los siguientes resultados para $\alpha = 0.05$.

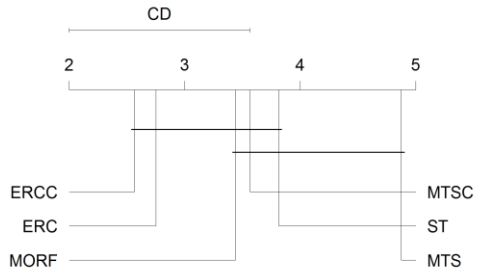


Figura 42 Prueba de Bonferroni-Dunn para la tabla 9

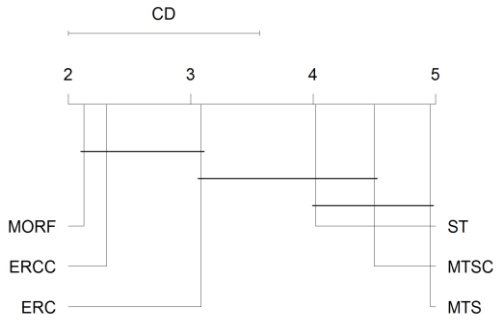


Figura 43 Prueba de Bonferroni-Dunn para la tabla 10

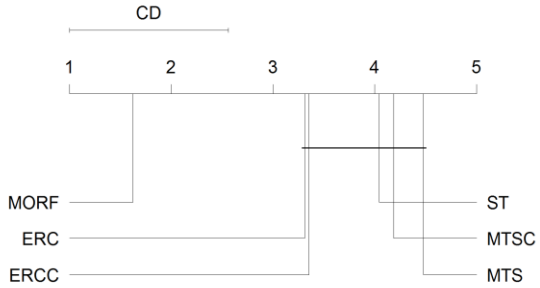


Figura 44 Prueba de Bonferroni-Dunn para la tabla 11

Como se puede apreciar en las imágenes anteriores la distancia entre los algoritmos tras ser aplicados en conjuntos de datos que presentan más atributos que instancias si supera la diferencia critica establecida por Nemenyi, por tanto, se puede afirmar que si existen diferencias significativas entre ellos con un 95% de certeza. Incluso, la familia de algoritmos basados en apilamiento no tiene un buen desempeño a diferencia de la experimentación (Spyromitros-Xioufis et al., 2016b) donde los algoritmos ERC y ERCC tuvieron buenos resultados, además estos últimos se comportaron mejor que el MORF para aquellos conjuntos de datos, no siendo así para los generados en esta investigación.

Además, se realizó la prueba de Friedman para determinar un ranking entre los regresores base obteniendo como resultado un $\chi^2 = 98.023$ con un *p_value* = 2.2e-16 menor que $\alpha = 0.05$. Esto significa que se rechaza la hipótesis nula, es decir, existen diferencias significativas entre los regresores base. A continuación, se muestra una imagen con el ranking entre los regresores base.

Figura 45 Ranking de Friedman entre los regresores base

Tras obtener el ranking para los regresores base se procedió a realizar la prueba de Bonferroni-Dunn obteniéndose como resultados para $\alpha = 0.05$ los presentados en la figura 36. Por lo tanto, se concluye que si existe diferencias significativas entre los regresores base.

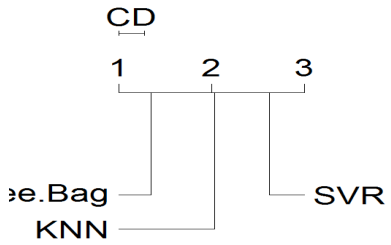


Figura 46 Prueba de Bonferroni-Dunn para los regresores base

3.5. Conclusiones del capítulo

- Se generaron conjuntos de datos con diferentes características que ayudaron a obtener un amplio abanico de pruebas para los distintos algoritmos que trae implementado MULAN para problemas Multi-Target Regression
- Se seleccionó la prueba de Friedman para realizar las pruebas estadísticas debido a que los problemas de aprendizaje automático no se garantiza el cumplimiento de los supuestos paramétricos para aplicar la prueba ANOVA.
- Se demostró la calidad de la herramienta creada a través de los resultados obtenidos en las pruebas de software.
- Se realizaron las pruebas estadísticas de Friedman y el test post-hoc de Bonferroni-Dunn que arrojó como resultados que si existe diferencias significativas entre los algoritmos cuando se usan conjuntos de datos que contienen más atributos que instancias con un 95% de certeza. Además, el mejor algoritmo como regresor base es RepTree-Bag.

CONCLUSIONES

Con la realización del presente trabajo de diploma se ha cumplido con el objetivo general propuesto, así como con las tareas de la investigación obteniéndose las siguientes conclusiones:

- Durante la fundamentación teórica de la investigación se analizó el algoritmo para generar datos para regresiones múltiples asentando así las bases para dar solución a las necesidades descritas en la problemática actual.
- El empleo de netbeans en su versión 8.2, MULAN en su versión 1.5.0, Visual Paradigm en su versión 8.0 y JAVA 8 permitió que la implementación de la propuesta de solución se realizara satisfactoriamente.
- Se implementó un generador de datos sintéticos para problemas con salidas múltiples integrados en MULAN, el cual favoreció la experimentación en problemas Multi-Target Regression.
- La validación de los resultados aportó que los algoritmos implementados en MULAN si poseen diferencias significativas cuando se trata con conjuntos de datos que poseen muchos más atributos que instancias.

RECOMENDACIONES

Para futuras investigaciones se recomienda:

- Extender la herramienta desarrollada para modelos no lineales la cual permita obtener conjuntos de datos con las mismas características.

REFERENCIAS BIBLIOGRÁFICAS

1. ALEGSA, 2018. DICCIONARIO DE INFORMÁTICA Y TECNOLOGÍA. .

2. CANO, ISAAC y VICENC ¸ TORRA, 2009. *Generation of Synthetic Data by means of Fuzzy c-Regression* [en línea]. 2009. S.l.: s.n. [Consulta: 3 abril 2018]. Disponible en: http://www.iiia.csic.es/files/pdfs/FUZZIEEE09.pdf.

3. DEITEL, PAUL J y DEITEL, HARVEY M., 2008. *CÓMO PROGRAMAR EN JAVA*. Séptima edición. México: PEARSON EDUCACIÓN. ISBN 978-970-26-1190-5.

4. DEMŠAR, J., 2006. Statistical Comparisons of Classifiers over Multiple Data Sets. *Journal of Machine Learning Research*, vol. 7, no. Jan, pp. 1-30. ISSN ISSN 1533-7928.

5. DUNN, O.J., 1961. Multiple Comparisons among Means. *Journal of the American Statistical Association*, vol. 56, no. 293, pp. 52-64. ISSN 0162-1459. DOI 10.1080/01621459.1961.10482090.

6. EXES, 2018. Cursos oracle,cursos java,master oracle,master java,formación en informatica. [en línea]. [Consulta: 31 marzo 2018]. Disponible en: http://www.mundolinux.info/que-es-xml.htm.

7. FISHER, S.R.A., 1959. *Statistical Methods and Scientific Inference*. S.l.: Hafner.

8. FLOWER, 2010. Java Foundations: Java - Estándares de programación. *Java Foundations* [en línea]. Disponible en: http://javafoundations.blogspot.com/2010/07/java-estandares-de-programacion.html.

9. FRANCISCO MIGUEL RODRÍGUEZ SÁNCHEZ, 2014. *Herramientas para Big Data: Entorno Hadoop.* [en línea]. septiembre 2014. S.l.: s.n. [Consulta: 2 febrero 2018]. Disponible en: http://repositorio.upct.es/bitstream/handle/10317/4402/tfg482.pdf?sequence=1&isAllowed=y.

10. FRIEDMAN, M., 1937. The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance. *Journal of the American Statistical Association*, vol. 32, no. 200, pp. 675-701. ISSN 0162-1459. DOI 10.1080/01621459.1937.10503522.

11. FRIEDMAN, M., 1940. A Comparison of Alternative Tests of Significance for the Problem of m Rankings. *The Annals of Mathematical Statistics*, vol. 11, no. 1, pp. 86-92. ISSN 0003-4851. DOI 10.2307/2235971.

12. GAMMA, ERICH, HELM, RICHARD, JOHNSON, RALPH y VLISSIDES, JOHN, 2015. *Design Patterns Elements of Reusable Object-Oriented Software*. S.l.: Pearson. ISBN 978-93-325-5540-2.

13. GILLBERG, J., MARTTINEN, P., PIRINEN, M., KANGAS, A.J., SOININEN, P., ALI, M., HAVULINNA, A.S., JÄRVELIN, M.-R., ALA-KORPELA, M. y KASKI, S., 2016. Multiple Output Regression with Latent Noise. *Journal of Machine Learning Research*, vol. 17, no. 122, pp. 1-35.

14. GITHUB, 2018. prakhargvp/csvWithCplusplus. *GitHub* [en línea]. [Consulta: 10 junio 2018]. Disponible en: https://github.com/prakhargvp/csvWithCplusplus.

15. GOOGLE SITES, 2018. Biblioteca informática - LOGICA DE PROGRAMACION. [en línea]. [Consulta: 4 abril 2018]. Disponible en: https://sites.google.com/site/israelcortess/documentos/biblioteca-informatica-3.

16. HAMILTON, L.C., 1990. *Modern Data Analysis: A First Course in Applied Statistics*. S.l.: Brooks/Cole Publishing Company. ISBN 978-0-534-12846-3.

17. HAN, J., PEI, J. y KAMBER, M., 2011. *Data Mining: Concepts and Techniques*. S.l.: Elsevier. ISBN 978-0-12-381480-7.

18. HIRU, 2018. Distribución normal - hiru. [en línea]. [Consulta: 3 mayo 2018]. Disponible en: https://www.hiru.eus/es/matematicas/distribucion-normal.

19. HOLLAND, B., 1991. On the application of three modified Bonferroni procedures to pairwise multiple comparisons in balanced repeated measures designs. *Computational Statistics Quarterly*, pp. 219-231.

20. HOLM, S., 1979. A Simple Sequentially Rejective Multiple Test Procedure. *Scandinavian Journal of Statistics*, vol. 6, no. 2, pp. 65-70. ISSN 0303-6898.

21. HOMMEL, G., 1988. A stagewise rejective multiple test procedure based on a modified Bonferroni test. *Biometrika*, vol. 75, no. 2, pp. 383-386. ISSN 0006-3444, 1464-3510. DOI 10.1093/biomet/75.2.383.

22. IMAN, R.L. y DAVENPORT, J.M., 1980. Approximations of the critical region of the fbietkan statistic. *Communications in Statistics - Theory and Methods*, vol. 9, no. 6, pp. 571-595. ISSN 0361-0926. DOI 10.1080/03610928008827904.

23. JIMENA TORRES TOMÁS, NEWTON SPOLAOR, EVERTON ALVARES CHERMAN y MARIA CAROLINA MONARD, 2014. *A Framework to Generate Synthetic Multi-label Datasets*. 2014. S.l.: s.n.

24. KEN ARNOLD; JAMES GOSLING; DAVID HOLMES, 2001. *EL LENGUAJE DE PROGRAMACION JAVA TM*. S.l.: ADDISON-WESLEY. ISBN 978-84-7829-045-1.

25. MASSIMILIANO PONTIL, ANDREAS ARGYRIOU y THEODOROS EVGENIOU, 2007. Multi-Task Feature Learning. , pp. 41-48.

26. MEMBER, Z.Y. (EURASIP, ABDALLAH, F., DENOEUX, T. y SNOUSSI, H., 2011. A Dependent Multilabel Classification Method Derived from the <Emphasis Type=«Italic»>k</Emphasis>-Nearest Neighbor Rule. *EURASIP Journal on*

Advances in Signal Processing, vol. 2011, no. 1, pp. 645964. ISSN 1687-6180. DOI 10.1155/2011/645964.

27. MENESES GÓMEZ, MATEO, 2017. Espacios adaptables a través del diseño modular. [en línea]. Universidad Católica de Pereira, 2017. [Consulta: 8 mayo 2018]. Disponible en: http://200.21.98.67:8080/jspui/handle/10785/4361.

28. MENG, X., BRADLEY, J., YAVUZ, B., SPARKS, E., VENKATARAMAN, S., LIU, D., FREEMAN, J., TSAI, D., AMDE, M., OWEN, S., XIN, D., XIN, R., J. FRANKLIN, M., ZADEH, R., ZAHARIA, M. y TALWALKAR, A., 2015. MLlib: Machine Learning in Apache Spark. *JMLR*, vol. 17.

29. MOA, 2018. MOA. *MOA* [en línea]. [Consulta: 30 marzo 2018]. Disponible en: https://moa.cms.waikato.ac.nz/.

30. NEMENYI, P., 1963. *Distribution-free Multiple Comparisons*. S.l.: s.n.

31. OHMYROOT, 2017. Estándares de codificación - ¡Mejora tu código! *Ohmyroot!* [en línea]. [Consulta: 9 mayo 2018]. Disponible en: https://www.ohmyroot.com/buenas-practicas-legibilidad-del-codigo/.

32. ORACLE CORPORATION, 2017a. Java Platform, Standard Edition What's New in Oracle JDK 8. [en línea]. [Consulta: 8 noviembre 2017]. Disponible en: https://docs.oracle.com/javase/8/.

33. ORACLE CORPORATION, 2017b. NetBeans IDE. [en línea]. [Consulta: 8 noviembre 2017]. Disponible en: https://netbeans.org/.

34. PIZARRO, J., GUERRERO, E. y GALINDO, P.L., 2002. Multiple comparison procedures applied to model selection. *Neurocomputing*, vol. 48, no. 1-4, pp. 155-173. ISSN 09252312. DOI 10.1016/S0925-2312(01)00653-1.

35. RAI, P., KUMAR, A. y DAUME, H., 2012. Simultaneously Leveraging Output and Task Structures for Multiple-Output Regression. En: F. PEREIRA, C.J.C. BURGES, L. BOTTOU y K.Q. WEINBERGER (eds.), *Advances in Neural Information Processing Systems 25* [en línea]. S.l.: Curran Associates, Inc., pp. 3185–3193. [Consulta: 7 abril 2018]. Disponible en: http://papers.nips.cc/paper/4501-simultaneously-leveraging-output-and-task-structures-for-multiple-output-regression.pdf.

36. ROGER S. PRESSMAN, 2010. *Ingenieria del Software. Un Enfoque Practico. Septima Edicion*. S.l.: Mc Graw Hill. ISBN 978-607-15-0314-5.

37. SCHOLARIUM SAS, 2017. UML | SCHOLARIUM SAS - Servicios Integrales de optimización, gestión y sistematización de procesos de negocio. [en línea]. [Consulta: 12 mayo 2017]. Disponible en: http://scholarium.info/uml/.

38. SCRIBD INC, 2018. Visual Paradigm - UML | Object Relational Mapping | Databases. *Scribd* [en línea]. [Consulta: 26 febrero 2018]. Disponible en: https://es.scribd.com/document/8748836/Visual-Paradigm-UML.

39. SHAFRANOVICH, Y., 2005. Common Format and MIME Type for Comma-Separated Values (CSV) Files. [en línea]. [Consulta: 10 junio 2018]. Disponible en: https://tools.ietf.org/html/rfc4180.

40. SHESKIN, D.J., 2003. *Handbook of Parametric and Nonparametric Statistical Procedures: Third Edition*. S.l.: CRC Press. ISBN 978-1-4200-3626-8.

41. SORNOZA, V. y OSWALDO, C., 2015. *DESARROLLO E IMPLEMENTACIÓN DE LA PLATAFORMA TECNOLÓGICA PARA LA COORDINACIÓN DE SEGUIMIENTO A GRADUADOS DE LAS CARRERAS INGENIERÍA EN SISTEMAS COMPUTACIONALES Y NETWORKING* [en línea]. Thesis. S.l.: Universidad de Guayaquil Facultad de Ciencias Matemáticas y Física Carrera de Ingeniería en Sistemas Computacionales. [Consulta: 10 noviembre 2017]. Disponible en: http://repositorio.ug.edu.ec/handle/redug/10114.

42. SPYROMITROS-XIOUFIS, E., TSOUMAKAS, G., GROVES, W. y VLAHAVAS, I., 2016. Multi-target regression via input space expansion: treating targets as inputs. *Machine Learning*, vol. 104, no. 1, pp. 55-98. ISSN 0885-6125, 1573-0565. DOI 10.1007/s10994-016-5546-z.

43. STRATHMANN, HEIKO, GAL, VIKTOR y LISITSYN, SERGEY, 2018. Shogun. [en línea]. [Consulta: 30 marzo 2018]. Disponible en: http://www.shogun-toolbox.org/.

44. THE APACHE SOFTWARE FOUNDATION, 2016. Math – Commons Math: The Apache Commons Mathematics Library. [en línea]. [Consulta: 9 abril 2018]. Disponible en: http://commons.apache.org/proper/commons-math/.

45. TOM M. MITCHELL, 1997. *Machine Learning*. S.l.: McGraw-Hill Science. ISBN 0-07-042807-7.

46. TSOUMAKAS, G., KATAKIS, I. y VLAHAVAS, I., 2009. Mining Multi-label Data. En: DOI: 10.1007/978-0-387-09823-4_34, *Data Mining and Knowledge Discovery Handbook* [en línea]. S.l.: Springer, Boston, MA, pp. 667-685. [Consulta: 30 marzo 2018]. Disponible en: https://link.springer.com/chapter/10.1007/978-0-387-09823-4_34.

47. TSOUMAKAS, G., KATAKIS, I. y VLAHAVAS, I., 2018. Data Format. [en línea]. [Consulta: 31 marzo 2018]. Disponible en: http://mulan.sourceforge.net/format.html.

48. *Tutorial. Meka 1.9.0* [en línea], 2015. noviembre 2015. S.l.: s.n. [Consulta: 21 enero 2018]. Disponible en: https://www.researchgate.net/...MEKA.../Tutorial.pdf.

49. TXIKIBOO, 2014. Archivos ARFF (Weka). *Txikiboo* [en línea]. [Consulta: 10 noviembre 2017]. Disponible en: https://txikiboo.wordpress.com/2014/01/16/archivos-arff-weka/.

50. UNIVERSITY OF LJUBLJANA, 2018. Orange – Data Mining Fruitful & Fun. [en línea]. [Consulta: 26 febrero 2018]. Disponible en: https://orange.biolab.si/.

51. UNIVERSITY OF WAIKATO, 2018. Machine Learning Project at the University of Waikato in New Zealand. [en línea]. [Consulta: 11 marzo 2018]. Disponible en: https://www.cs.waikato.ac.nz/ml/index.html.

52. VÁZQUEZ, E.G., ESCOLANO, A.Y., RIAÑO, P.G. y JUNQUERA, J.P., 2001. Repeated Measures Multiple Comparison Procedures Applied to Model Selection in Neural Networks. En: DOI: 10.1007/3-540-45723-2_10, *Bio-Inspired Applications of Connectionism* [en línea]. S.l.: Springer, Berlin, Heidelberg, pp. 88-95. [Consulta: 8 abril 2018]. Disponible en: https://link.springer.com/chapter/10.1007/3-540-45723-2_10.

53. VIKLUND, ANDREAS, 2017. JFreeChart. [en línea]. [Consulta: 4 abril 2018]. Disponible en: http://www.jfree.org/jfreechart/.

54. YANETSYS SARDUY DOMÍNGUEZ, 2007. El análisis de información y las investigaciones cuantitativa y cualitativa. [en línea], [Consulta: 22 enero 2018]. Disponible en: http://bvs.sld.cu/revistas/spu/vol33_3_07/spu20207.htm#cargo.

55. ZHANG, M.-L., PEÑA, J.M. y ROBLES, V., 2009. Feature selection for multi-label naive Bayes classification. *Information Sciences*, vol. 179, no. 19, pp. 3218-3229. ISSN 00200255. DOI 10.1016/j.ins.2009.06.010.

ANEXOS