UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Facultad 2

Algoritmos de adaptación ML-C4.5 y RF-C4.5 para problemas de aprendizaje multi-etiquetas integrado a scikit-multilearn

Autor: Raydel Barbaro Cardoso Bocourt

Tutor: Ing. Vladimir Milián Núñez

La Habana, julio de 2018

Declaración de Autoría:

Declaro ser el autor de la presente tesis de título "**Algoritmos de adaptación ML-C4.5 y RF-C4.5 para problemas de aprendizaje multi-etiquetas integrado a scikit-multilearn**" y entrego a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ___ días del mes de julio del año 2018

_____

Raydel Barbaro Cardoso Bocourt

_____

Vladimir Milián Núñez

Agradecimientos:

Quiero agradecerle primero que todo a mis padres, a quienes les debo mucho y doy las gracias por todo lo que me han enseñado hasta ahora y por todo el sacrificio que han tenido que hacer para que yo llegara hasta aquí. También quiero agradecerles a mis hermanos por siempre estar ahí cuando los necesitaba, por preocuparse siempre por mí, por enseñarme como hacer las cosas y quiero que sepan que por mucho que diga, no existen palabras para expresar cuanto los quiero. Agradecer a mi tutor, al oponente y al tribunal por la ayuda que me han brindado en este tiempo. Darle las gracias a la gente del grupo 1, porque son muchos recuerdos los que me llevo y sé que aunque a muchos no los volveré a ver, quiero que sepan que siempre los llevaré conmigo. Agradecerle a Suan, Hassan, Carlos, José Ernesto, Antonio, El Chino, Roy, El Yoyo, Anna Laura, Reyzer, Yanlee, Román, Yosniel, Oswal, Josue, Cristian, Armando, Eduardo, al Pepe, a Novoa, Aguilera, la banda de los indeseables, y a todos los demás, sé que me faltan gente, pero son muchos, mi gente los quiero. Agradecerles a todos mis profesores por todos los consejos, enseñanzas y regaños, porque me enseñaron a ser una mejor persona. En fin, agradecerle a todos los que estuvieron presentes en esta etapa y quiero que sepan, lo que vivimos es para siempre y que pase lo que pase, nunca los olvidaré. Los quiero mi gente.

Resumen:

La minería de datos como disciplina intenta reducir la brecha existente entre el volumen de información almacenado y el conocimiento que es capaz de extraerse de esta información. Dentro de la minería de datos el aprendizaje automático (AA) emplea información histórica para, a través de modelos, predecir el comportamiento de nuevas variables asociadas al problema. En los últimos años han surgido problemas de la vida práctica relacionados con la necesidad de predecir valores de salidas, modelados a través de diferentes estructuras complejas.

El presente trabajo se fundamenta en la implementación de los algoritmos de clasificación multi-etiqueta con enfoque de adaptación basado en arboles ML-C4.5 y RF-C4.5 en la herramienta computacional de Python scikit-multilearn. Ha surgido la necesidad de poder usar estos algoritmos en otras plataformas de desarrollo como es el caso de Python el cual es uno de los lenguajes más novedosos y competitivos de estos tiempos el cual posee una biblioteca especializada en la clasificación multi-etiqueta que es scikit-multilearn.

Se describe el proceso de implementación e integración a la herramienta scikit-multilearn siguiendo las pautas dictadas por la metodología KDD. Se evalúan los resultados alcanzados aplicando la Prueba de Friedman y haciendo uso de métricas para evaluar el rendimiento de los clasificadores multi-etiqueta. Se obtiene como resultado que los algoritmos propuestos no presentan diferencias significativas con los algoritmos del estado de arte.

Palabras claves: Aprendizaje automático, clasificación multi-label, árboles de decisión, pruebas de Friedman, Python.

Summary:

Data mining as a discipline seeks to reduce the existing gap between the volume of information stored and the knowledge that is able to be extracted from this information. Within data mining, machine learning (AA) uses historical information to predict, through models, the behavior of new variables associated with the problem. In recent years problems of practical life have arisen related to the need to predict output values, modeled through different complex structures.

The present work is based on the implementation of multi-label classification algorithms with adaptation approach based on trees ML-C4.5 and RF-C4.5 in the Python computational tool scikit-multilearn. There has been a need to be able to use these algorithms in other development platforms, such as Python, which is one of the most innovative and competitive languages of these times, which has a specialized library in the multi-label classification that is scikit- multilearn

The implementation and integration process to the scikit-multilearn tool is described following the guidelines dictated by the KDD methodology. The results achieved are evaluated by applying the Friedman Test and using metrics to evaluate the performance of the multi-label classifiers. It is obtained as a result that the proposed algorithms do not present significant differences with the algorithms of the state of art.

Key words: Machine learning, multi-label classification, decision trees, Friedman test, Python.

Tabla de Contenidos:

Índice de Figuras:

Índice de tablas:

Introducción:

La minería de datos como disciplina intenta reducir la brecha existente entre el volumen de información almacenado y el conocimiento que es capaz de extraerse de esta información. El proceso, idealmente, debe ser automático y el objetivo principal es que los patrones descubiertos deben tener algún significado en el sentido de que puedan ser utilizados como conocimiento útil. Dentro de la minería de datos el aprendizaje automático (AA) está considerado una disciplina que emplea información histórica formalmente estructurada sobre un fenómeno, para a través de modelos predecir el comportamiento de nuevas variables asociadas al problema (Witten, et al., 2005) (J, et al., 1998)

El AA estudia los programas que aprenden o evolucionan, que son capaces de generalizar o inferir comportamientos a partir de cierta cantidad de información suministrada en forma de ejemplos, para realizar una tarea determinada cada vez mejor. El objetivo fundamental de todo proceso de AA es utilizar la evidencia conocida (Datos de Entrenamiento) para poder crear una hipótesis (Modelos) y dar una respuesta a nuevas situaciones no conocidas en un contexto dado (Sierra Araujo).

Una clasificación general de los algoritmos de AA los separa en métodos supervisados y no supervisados. La aplicación de uno u otro algoritmo depende de las características del problema a resolver, pues cada uno de ellos puede ser útil en determinadas circunstancias. Los algoritmos supervisados intentan definir un modelo matemático a partir de la información de las clases de un conjunto de datos de entrenamiento. Los algoritmos no supervisados encuentran una partición de los datos sin utilizar la información de la clase. (Jin, y otros)

Entre las tareas de los algoritmos supervisados, los enfoques clásicos son en los que se modela el problema a partir de una serie de variables predictores y una variable que es la que se desea medir (Salida). En los últimos años han surgido un conjunto de problemas prácticos que no es adecuado modelarlos utilizando este tipo de enfoque clásico. En estos casos existe la necesidad de predecir valores de salida, pero modelados a través de diferentes estructuras complejas. (Bakir, y otros, 2007)

La clasificación multi-etiqueta (Multi-Label Classification, MLC) es un paradigma que ha experimentado un gran crecimiento estos últimos años debido a su éxito en problemas de actualidad como la clasificación de textos y multimedia, la predicción de funciones de genes y proteínas, el marketing directo, o la minería de redes sociales. (Gibaja, 2014)

Estas aplicaciones tienen en común que un objeto (patrón o instancia) puede pertenecer a más de una clase (etiqueta) simultáneamente, no satisfaciéndose la restricción only-one-label-per-pattern del aprendizaje clásico (single-label). Para representar este hecho, las etiquetas son variables binarias que indican la pertenencia a cada una de las clases de interés de forma análoga al aprendizaje multi-clase, pero con la diferencia de que un ejemplo puede tener más de un valor binario activo. La MLC implica mayor dificultad debido a aspectos como el coste computacional que generar consultar los modelos, la presencia de etiquetas desbalanceadas y relaciones entre etiquetas o la alta dimensionalidad de los datos (número de patrones, atributos y etiquetas).

Actualmente, existen dos enfoques principales para abordar un problema de clasificación multi-etiqueta: los métodos basados en transformación del problema y los métodos basados en la adaptación de algoritmo.

El método de trasformación se basa en transformar un problema multi-etiqueta en uno o varios problemas de una sola etiqueta que posteriormente son resueltos utilizando un algoritmo clásico de clasificación. Con este enfoque entre los algoritmos más utilizados se encuentran K-NN, Naive Bayes, Decision Trees, Random forest, Logistic Regression, Bagging Trees y otros.

El segundo enfoque (métodos de adaptación) consiste en adaptar o extender algún algoritmo clásico de clasificación de modo trabaje directamente con datos multi-etiqueta. En esta categoría son muy utilizados kNN, Ridge Regression, Decision Trees, Random Forests, entre otros.

Actualmente existen varias herramientas que implementan métodos de MLC desarrolladas sobre diversas tecnologías como Java, Matlab o Python. En Java se encuentran Mulan y Meka; en Matlab los paquetes desarrollados por el grupo LAMDA o en el sitio del profesor Min-Ling Zhang y para el caso de Python la herramienta scikit-multilearn.

La herramienta scikit-multilearn, se encuentra actualmente en desarrollo, y aunque cuenta con varios de los algoritmos más utilizados, en el caso de los algoritmos con enfoque de adaptación, solamente contiene los algoritmos basados en instancia BRkNN y MLkNN, que trabajan con el clasificador KNN. Existen otros algoritmos competitivos del estado del arte, que son utilizados con frecuencia por los investigadores que no están disponibles en la plataforma, como es el caso del algoritmo ML-C4.5 y RF-C4.5. Se debe señalar que actualmente Python es uno de los lenguajes que más se utiliza para realizar tareas de

aprendizaje automático y análisis de datos (González 2017). Además, que se han hecho pruebas de ejecutar el mismo algoritmo en las distintas plataformas y los de scikit-multilearn han mostrado ventajas en aspectos como el tiempo de funcionamiento (Szymański y Kajdanowicz 2017).

ML-C4.5 y RF-C4.5 son muy utilizados para resolver problemas de clasificación multi-label y la ausencia de estos en scikit-multilearn representa un problema para la comunidad de desarrollo debido a la imposibilidad de utilizar estos algoritmos para resolver problemas reales, además de la comparación de resultados con respecto a otros algoritmos para llegar a una mejor comprensión del problema.

Por la problemática anteriormente descrita se plantea como **problema de la investigación:** la inexistencia de los algoritmos ML-C4.5 y RF-C4.5 en la herramienta scikit-multilearn.

Teniendo en cuenta el problema antes propuesto se define como **objeto de estudio:** Algoritmos de Clasificación para problemas de aprendizaje multi-etiqueta.

La presente investigación tiene como **objetivo general:** Desarrollar en scikit-multilearn los algoritmos ML-C4.5 y RF-C4.5 para problemas de aprendizaje multi-etiqueta basado en el enfoque de adaptación.

Enmarcándose en el **campo de acción:** Algoritmos de Clasificación para problemas de aprendizaje multi-etiqueta basados en arboles de decisión.

Para poder alcanzar los objetivos trazados se definen los siguientes **objetivos específicos:**

- Caracterizar el marco teórico-conceptual de los algoritmos de aprendizaje multi-etiquetas basados en árboles de decisión con un enfoque de adaptación, con énfasis en las herramientas de aprendizaje automático disponible para ello.
- Implementar los algoritmos ML-C4.5 y RF-C4.5 en la herramienta scikit-multilearn, así como la base experimental para probar el algoritmo con otros el estado del arte.
- Validar la solución implementada mediante el diseño de experimentos comparando los resultados con algoritmos del estado del arte y con su implementación en otras herramientas.

Para apoyar el desarrollo de la investigación se emplean los siguientes métodos científicos:

Métodos Teóricos:

Analítico-Sintético: Se usa para realizar un estado del arte sobre algoritmos de clasificación para problemas de aprendizaje multi-etiqueta y las herramientas en los que se encuentran implementados.

Inductivo-Deductivo: Se utilizó para describir las bases teóricas del algoritmo ML-c4.5 y RF-4.5.

Métodos Empíricos:

Modelación: Se utiliza en el diseño y la implementación del ML-c4.5 y RF-4.5; en el diseño de los diagramas de clases, el diseño de la implementación y modelación de la arquitectura.

Histórico-Lógico: Este método permite estudiar la trayectoria histórico-real de la evolución y desarrollo de los algoritmos de clasificación MLL basados en adaptación.

Medición: En la experimentación para el cálculo del error en la clasificación y para comparar los resultados de los algoritmos estudiados.

Con el fin de estructurar el proceso de desarrollo del presente trabajo se dividió el mismo en un total de tres capítulos:

Capitulo1: “Fundamentación teórica. Clasificación multi-label.” Aborda el estado del arte del tema, conceptos y definiciones encontrados durante el estudio de las tendencias actuales.

Capitulo2:” Propuesta de solución de los algoritmos ML-C4.5 y RF-C4.5”. Se define la solución propuesta mediante modelos de procesos y las especificaciones de los requisitos de software. Se realiza una planificación para el cumplimiento de los mismos, así como el conjunto de tareas que se realizaran para su terminación.

Capitulo3:” Validación y Pruebas”. Se evalúa la calidad de la solución final mediante una serie de pruebas, con el fin de corregir errores y no conformidades presentes durante la implementación.

Capitulo1: Fundamentación Teórica.

Introducción:

En este capítulo se aborda en qué consisten los problemas de predicción multi-label. Se realiza un estudio del estado del arte de los algoritmos cuyo enfoque está basado en este tipo de problema. Además, se estudian y describen las herramientas existentes que implementan este tipo de algoritmo. Finalmente se describen las herramientas y metodología de desarrollo de software empleadas para elaborar una herramienta acorde a los requisitos planteados.

Clasificación:

La clasificación es uno de los temas de MD más populares. Es una tarea que vale como predicción, usualmente por métodos supervisados de aprendizaje (Aggarwal, C.C., 2014). La clasificación tiene la intención de aprender de patrones etiquetados, crear un modelo capaz de predecir la futura etiqueta, nunca vista anteriormente en los datos de entrenamiento. (Herrera, y otros, 2016)

El conjunto de atributos en un *dataset* de clasificación está dividido en dos subconjuntos. Lo primer contiene las características, las variables que actuarán como predictores. El segundo subconjunto sostiene los atributos de salida, las clases o las etiquetas asignadas a cada instancia. Los algoritmos de clasificación inducen el modelo analizando la correlación entre las características y las clases de salida. Una vez que un modelo entrenado es obtenido, puede usarse para procesar las características de nuevos datos de prueba adquiriendo una predicción de clases. (Herrera, y otros, 2016)

Dependiendo de la naturaleza del segundo subconjunto de atributos, varios tipos de problemas de clasificación pueden ser identificados. La tabla 1 resume las configuraciones más comunes, a merced del número de salidas y sus tipos.

Tabla 1: Problemas de clasificación atendiendo a las variables de salida.

número de salidas	El tipo de salida	Tipo de clasificación
1 por instancia	Binario	Binario
1 por instancia	Multi-valor	Multi class
n por instancia	Binario	Multi-label
n por instancia	Multi-valalor	Multi-dimensional
1 por n instancias	Binario /Multi-valor	Multi-instancia

Clasificación Binaria:

Éste es un de los problema más fáciles de clasificación que podemos afrontar (Aggarwal, C.C., 2014) . Las instancias en un *dataset* binario tienen sólo un atributo de salida, y puede tomar sólo dos valores diferentes. Estos son usualmente conocidos como positivo y negativo, pero también pueden ser interpretados como verdaderos y falsos, 1 y 0, o cualquier otra combinación de dos valores. Un ejemplo clásico de esta tarea es correo electrónico masivo no solicitado filtrado (vea a Fig.1), en el cual el clasificador aprende del contenido de los mensajes cuáles pueden ser considerados como correo electrónico masivo no solicitado. (Herrera, y otros, 2016)

Un clasificador binario toma experiencia para encontrar un límite capaz de separar las instancias en dos grupos, uno formado por las clases positiva y el otro para las negativas. En la práctica, dependiendo del espacio de característica de entrada, la distribución de pruebas puede ser bastante más difícil, necesitando fronteras adicionales entre las instancias. Las aplicaciones actuales de clasificación binaria son, correo electrónico filtrado: para eliminar mensajes del correo electrónico masivo no solicitado, prestar análisis: decide si el cliente es económicamente confiable o no, evaluación médica: determina si un paciente tiene una cierta enfermedad o no, y el reconocimiento de toda clase de patrones binarios. (Herrera, y otros, 2016)

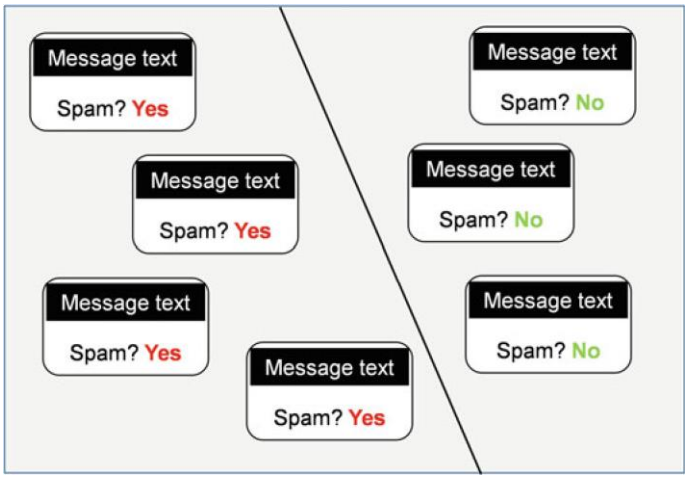


Figura 1. Correo Electrónico Masivo filtrado, es un problema típico de clasificación binaria.

Clasificación Multi-Clase:

Un *dataset* de multi-clase también tiene sólo un atributo de salida, como en los *datasets* binarios, pero puede sujetar cualquier set de valores predefinidos. El sentido de cada valor,

y el valor mismo, serían específicos para cada aplicación. El set de clases será finito y discreto, de lo contrario la tarea no sería clasificación sino regresión. Los valores de clase podrían tener una relación de orden o no. Uno de los mejores los ejemplos conocidos de clasificación de multi-clase es identificación de especies del iris (vea a Fig. 2). De cuatro características de la flor. Las longitudes del pétalo y del sépalo y las anchuras, los clasificadores aprenden cómo clasificar instancias nuevas en la familia correspondiente. (Herrera, y otros, 2016)

Muchos algoritmos de clasificación multi-clases utilizan binarización (Galar, et al., 2011), un método iterativo entrena un clasificador binario cada clase en contra de las demás, después de un acercamiento del OneVs-All (OVA), o para cada par de clases, usando un One-Vs-One (OVO). (Herrera, y otros, 2016)

La clasificación multi-class puede verse como una generalización de clasificación binaria. Hay sólo una salida, pero puede tomar cualquier valor, mientras en el caso binario que es limitado a un subconjunto de dos valores.

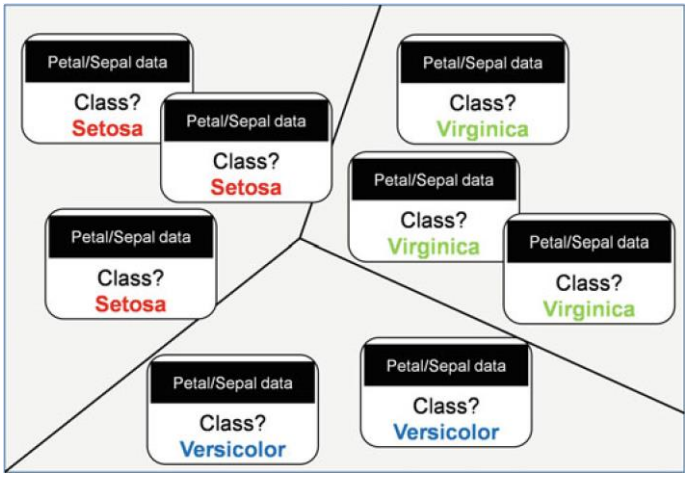


Figura 2: El clasificador aprende de la longitud y la anchura de pétalo y sépalo, prediciendo el cual de las tres especies las flores pertenecen.

Clasificación Multi-dimensional

El conjunto de datos multidimensionales (Breiman, L, 1996) también tiene un vector de salida asociado a cada instancia, en lugar de un único valor. Sin embargo, cada elemento de este vector puede tomar cualquier valor de un conjunto predefinido, sin limitarse a ser binario. Por lo tanto, la relación entre la clasificación multidimensional y multi-label es esencialmente la misma que se menciona anteriormente para la clasificación multi-clase y binaria. (Herrera, y otros, 2016)

Las técnicas multidimensionales están estrechamente relacionadas con las utilizadas en el campo multi-label, y los métodos de transformación desempeñan un papel relevante para afrontar esta tarea. El rango de aplicaciones potenciales también es similar. La clasificación multidimensional se usa para categorizar imágenes, música, textos y recursos análogos, pero predecir el etiquetado de un valor en un conjunto más grande que la clasificación multi-label. (Herrera, y otros, 2016)

Aprendizaje Multi-Instancia:

A diferencia de todos los otros tipos de clasificación descritos anteriormente, el paradigma de aprendizaje de instancias múltiples (Aha, 1997), también conocido como aprendizaje de múltiples instancias, aprende una etiqueta de clase común para un conjunto de vectores de características de entrada. Es un problema muy diferente, ya que cada instancia de datos lógicos se define no solo por un vector de características de entrada, sino por una colección de instancias físicas, cada una con un conjunto de atributos de entrada. Cada grupo de instancias generalmente se conoce como bolsa. La etiqueta de clase asociada pertenece a la bolsa, en lugar de a las instancias de datos individuales que contiene. (Herrera, y otros, 2016)

Entre las aplicaciones de este paradigma de clasificación, una de las más fáciles de comprender es la categorización de imágenes. Las imágenes originales se dividen en regiones o parches, por lo que cada imagen se representa mediante un conjunto de vectores, que contienen cada uno los datos de un parche. La etiqueta de clase asociada podría depender de ciertos objetos que aparecen en algunos de los parches. (Herrera, y otros, 2016)

El aprendizaje de instancias múltiples se puede extender considerando múltiples etiquetas de clase como salida, en lugar de solo una. A partir de esto, la casuística emerge variaciones tales como la clasificación multi-label de múltiples instancias (Zhou, y otros, 2012). Del mismo modo, se podría considerar una tarea de clasificación multidimensional de instancias múltiples. (Herrera, y otros, 2016)

Clasificación Multi-label:

La clasificación de Multi-label es una tarea predictiva de minería de datos con múltiples aplicaciones del mundo real, que incluye el etiquetado automático de muchos recursos como textos, imágenes, música y video. El aprendizaje a partir de datos multi-label se puede

lograr a través de diferentes enfoques, como la transformación de datos, la adaptación de métodos y el uso de conjuntos de clasificadores. (Herrera, y otros, 2016)

Este epígrafe comienza por definir formalmente el problema de clasificación multi-label, introduciendo la notación matemática y la terminología que se usará a lo largo de esta investigación. Luego, se describirán las diferentes áreas en las que se aplica la clasificación multi-label hoy en día, y se introducen los repositorios de los que se puede obtener este tipo de datos.

El aprendizaje de los datos multi-label se está enfrentando actualmente a través de enfoques dispares, que incluyen la transformación de datos y la adaptación de los métodos de clasificación tradicionales. El uso de conjuntos de clasificadores también es bastante popular en este campo. Además, se deben considerar algunos aspectos específicos, como el uso de información de dependencia de etiquetas, los problemas de alta dimensionalidad y el desequilibrio de etiquetas. Todos estos temas se describirán con más detalle, junto con una enumeración de las principales herramientas de software multi-label actualmente disponibles. (Herrera, y otros, 2016)

Definición formal:

La diferencia principal entre la clasificación tradicional y multi-label está en el resultado esperado de los modelos entrenados. Donde un clasificador tradicional solo regresará un valor, uno multi-label tiene que producir un vector de valores de salida. La clasificación de multi-label se puede definir formalmente de la siguiente manera. (Herrera, y otros, 2016)

Definiciones:

Definición 1:

Sea X el espacio de entrada, con muestras de datos $x \in A_1, A_2, \ldots, A_f$, siendo f el número de atributos de entrada y $A_1, A_2, \ldots, A_f$ conjuntos arbitrarios. Por lo tanto, cada instancia X se obtendrá como el producto cartesiano de estos conjuntos. (Herrera, y otros, 2016)

Definición 2:

Si L es el conjunto de todas las etiquetas posibles. $P(L)$ denota el conjunto de poder de L, que contiene todas las combinaciones posibles de etiquetas $l \in L$, incluido el conjunto vacío y L mismo. $k = |L|$ es el número total de etiquetas en L. (Herrera, y otros, 2016)

Definición 3:

Sea γ el espacio de salida, con todos los vectores posibles, $Y \in P(L)$. La longitud de Y siempre será k. (Herrera, y otros, 2016)

Definición 4:

D denota un conjunto de datos multilabel, que contiene un subconjunto finito de $A_1 \times A_2 \times \ldots \times A_f \times P(L)$. Ejemplo $(X,Y) \in D|X \in A_1 \times A_2 \times \ldots \times A_f,\ Y \in P(L)$ será una instancia o muestra de datos. $n = |D|$ será el número de elementos en D. (Herrera, y otros, 2016)

Definición 5:

Si F es un clasificador multilabel, definido como $F: x \rightarrow \gamma$. La entrada a F será cualquier instancia $x \in X$, y la salida será la predicción $Z \in Y$. Por lo tanto, la predicción del vector de etiquetas asociadas con cualquier instancia se puede obtener como $Z = F(x)$. (Herrera, y otros, 2016)

Simbología:

A partir de estas definiciones, se deriva la siguiente lista de símbolos, que se utilizará en este capítulo y en los siguientes (Herrera, y otros, 2016):

$\boldsymbol{D}$ - Cualquier conjunto de datos multi-label (MLD).

n - El número de instancias de datos en D.

L - El conjunto completo de etiquetas que aparecen en D.

l - Cualquiera de las etiquetas en L.

k - Número total de elementos en L.

x - El conjunto de atributos de entrada de cualquier instancia.

f - La cantidad de atributos que comprende x.

X - El espacio de entrada completo en D, que consta de todas las instancias x.

Y - El conjunto de etiquetas de salida (conjunto de etiquetas) de cualquier instancia.

γ - El espacio de salida completo en D, compuesto por todas las instancias Y.

Z - El conjunto de etiquetas predicho por el clasificador.

Al referirse a instancias individuales específicas, se utilizará la notación inusual E_i. Análogamente, Y haré referencia al verdadero conjunto de etiquetas de la instancia, Z_i el conjunto de etiquetas predicho por el clasificador a la i-ésima instancia, y así sucesivamente.

Las medidas destinadas a evaluar el rendimiento de un clasificador (se describirán en próximos epígrafes) calculan las diferencias entre Y_iy Z_i, es decir, el conjunto de etiqueta real asociado con cada instancia y el predicho por el clasificador. El objetivo al entrenar un clasificador multi-label sería reducir la tasa de error provocada por estas métricas.

Terminología:

Además de la notación anterior, los siguientes términos se usarán con frecuencia en todo el texto:

- **MLD / MLDs:** cualquier conjunto de datos multi-label o grupo de conjuntos de datos multi-label.
- **MLC:** cualquier clasificador multi-label o algoritmo de clasificación multi-label.
- Instancia / muestra: una fila en un MLD, incluidos sus atributos de entrada y el conjunto de etiquetas asociado.
- **Atributos / características:** generalmente se usa para referirse al conjunto de atributos de entrada en el MLD, sin incluir el conjunto de etiquetas de salida.
- **Etiqueta:** cualquiera de los atributos de salida asociados con una instancia.
- **Conjunto de etiquetas:** un vector de etiquetas {0,1} k cuya longitud siempre será k.
- **Conjunto de etiquetas verdaderas:** el conjunto de etiquetas con el que se anota una muestra en el MLD.
- **Conjunto de etiquetas pronosticadas:** el conjunto de etiquetas que proporciona un MLC como salida para una nueva muestra.

Aplicaciones de la Clasificación multi-label:

Una vez que se han introducido los conceptos principales vinculados a la clasificación multi-label, la siguiente pregunta que surja posiblemente sea para qué se puede utilizar. Como se explicó en la sección anterior, un clasificador multi-label tiene como objetivo predecir un conjunto de etiquetas relevantes para una nueva instancia de datos.

En esta sección, se representan varias áreas de aplicación que pueden aprovechar esta funcionalidad. Posteriormente, se detallarán casos de uso específicos que pertenecen a cada una de estas áreas.

Categorización de texto:

La clasificación de Multi-label tiene sus raíces como una solución para organizar documentos de texto en varias categorías no exclusivas. Esta es la razón por la cual hay muchas publicaciones sobre este tema.

Los documentos textuales se pueden encontrar en cualquier lugar, desde grandes compañías que almacenan todo tipo de acuerdos e informes a individuos que presentan sus facturas y mensajes de correo electrónico. Todos los libros y revistas publicadas, nuestros registros médicos históricos, así como artículos en medios electrónicos, publicaciones en blogs, mensajes de preguntas y respuestas en el foro, etc., también son documentos de texto. La mayoría de ellos se pueden clasificar en más de una categoría, por lo tanto, la utilidad de la clasificación multi-label para realizar este tipo de trabajo.

El proceso habitual para transformar un conjunto de documentos de texto en un MLD se basa en técnicas de minería de textos. Los documentos fuente se analizan, las palabras no informativas se eliminan y los vectores con cada ocurrencia de palabra entre los documentos se computan. Al final de este proceso, cada documento se describe por una fila en el MLD, y las columnas corresponden a palabras y sus frecuencias o algún otro tipo de representación como TF / IDF (Frecuencia de Término / Frecuencia de Documento Inverso).

Etiquetado de recursos multimedia:

Aunque los documentos que solo contienen texto fueron los más frecuentes en el pasado, hoy en día las imágenes, los videos y la música son recursos comúnmente utilizados debido al enorme crecimiento experimentado por las tecnologías de almacenamiento y comunicación. Atendiendo a las estadísticas de YouTube de 2015, se estima que se suben 432 000 videos nuevos cada día. La cantidad de nuevos clips de música y sonido publicados cada día también es impresionante, y las nuevas imágenes y fotografías publicadas en todas partes, desde blogs hasta sitios web como Tumblr, alcanzan la barrera de millones por día. (Herrera, y otros, 2016)

La clasificación de Multi-label se ha utilizado para etiquetar todos estos tipos de recursos (Boutell, y otros, 2004), (Briggs, y otros, 2012) (Duygulu, y otros, 2002), (Snoek, y otros,

2006), identificando los objetos que aparecen en conjuntos de imágenes, las emociones producidas por los clips de música o los conceptos que pueden derivarse de videos. De esta forma, gran cantidad de recursos nuevos se pueden clasificar correctamente sin necesidad de intervención humana, algo que sería bastante costoso.

Aunque una imagen se puede representar como un vector de valores de color, tomando cada píxel como una columna, generalmente se pre procesan para obtener las características más relevantes. Para hacerlo, se aplican técnicas de segmentación destinadas a extraer límites, la imagen puede ser complicada para transformar el espacio de píxeles en un espacio vacío, etc. Se ponen en práctica procedimientos similares con otros recursos multimedia.

Genética / Biología:

La proteínica y la genómica son campos de investigación que han experimentado un gran crecimiento en los últimos años. Como resultado, se han creado inmensas bases de datos con secuencias de proteínas producidas, pero solo una pequeña fracción de ellos se ha estudiado para determinar su función. El análisis de las propiedades de las proteínas y la expresión génica es una tarea muy costosa, pero las técnicas de DM pueden acelerar el proceso y abaratarlo.

La aplicación de una clasificación multicultural en un área (Protein classification with multiple algoithms. In: Proc. 10th Panhellenic Conference on Informatics, 2005) (A kernel method for multi-labelled classification. In: Advances in Neural Information Processing Systems,, 2001) está destinada a predecir las funciones biológicas de los genes. Cada gen puede expresar más de una función a la vez, de ahí el interés en usar técnicas MLC para enfrentar este problema. Las características utilizadas como predictores suelen ser motivos de la proteína, que son rasgos sobre su estructura interna. Los motivos estructurales indican cómo se conectan los elementos en una capa estructural secundaria. Algunos patrones, como las cadenas de aminoácidos, se pueden identificar y usar como características predictivas.

Otros campos de aplicación:

Además de las mencionadas en las secciones anteriores, la clasificación multi-label se ha utilizado en muchas otras aplicaciones, tanto públicas como privadas. La mayoría de ellos podría incluirse en las dos primeras categorías, ya que con el tiempo la meta es categorizar

textos, sonidos, imágenes y videos. Los siguientes son algunos de los más interesantes (Herrera, y otros, 2016):

- El análisis de las expresiones no verbales en el habla es el enfoque en (Sobol-Shikler, T., Robinson, P, 2010), con el objetivo de detectar cómo se siente la gente. El objetivo es predecir varios estados afectivos no mutuamente excluyentes a la vez. Los atributos predictivos son un conjunto de características vocales como la entonación y la velocidad del habla.
- En (Cong, H., Tong, L.H, 2008), los autores proponen un sistema para clasificar automáticamente los registros de patentes. El problema abordado es facilitar la búsqueda de documentos anteriores de acuerdo con los principios inventivos. Al final, esta propuesta se puede ver como otra solución de categorización de texto.
- El tema presentado en (Sriram, B., Fuhry, D., Demir, E., Ferhatosmanoglu, H., Demirbas, M, 2010) tiene por objeto mejorar el proceso de búsqueda de información pertinente información en Twitter. Cinco etiquetas diferentes se definen para clasificar twits, incluidas noticias, opiniones y eventos. Varios de ellos se pueden asignar simultáneamente al mismo twit.
- Analizar el movimiento complejo en eventos es el objetivo del sistema propuesto en (Chen, X., Zhan, Y., Ke, J., Chen, X, 2015). Combina trayectorias e hipergramas multi-lobulares de objetivos en movimiento en secuencias de video, detectando la relación entre las características de bajo nivel y los conceptos de alto nivel que se preverán.

En general, la clasificación multi-label podría ser adecuada para cualquier escenario en el que algún tipo de información, sin importar su tipo, siempre que pueda ser procesada por un algoritmo MLC, sea asignada a dos o más categorías simultáneamente.

Repositorios de MLD:

Por lo general, cuando se generan nuevos conjuntos de datos, los autores de cada trabajo original proporcionan públicamente los MLD que han producido, para que otros investigadores puedan utilizarlos en sus propios estudios. No obstante, la mayoría de los MLD se pueden obtener directamente de algunos de los siguientes repositorios de datos (Herrera, y otros, 2016):

- ***MULAN*:** El bien conocido paquete de aprendizaje multi-label tiene un repositorio de datos asociado. A partir de 2016, más de 20 MLD están disponibles en él. Estos MLD

están en ARFF (Attribute-Relation File Format) y cada uno tiene un archivo XML asociado que describe las etiquetas y sus relaciones. El archivo XML es necesario, ya que la posición de los atributos que actúan como etiquetas no se supone que esté al final.

- ***KEEL*:** KEEL es un software de fuente abierta que proporciona muchos algoritmos de pre-procesamiento y DM. Tiene un repositorio de datos asociado con casi 20 MLD. Junto con los conjuntos de datos completos, también se suministran particiones 10-fcv y 5-fcv. El formato de archivo KEEL, basado en ARFF, incluye el nombre de los atributos de salida en el encabezado; por lo tanto, no se necesita un archivo XML separado.
- ***MEKA:*** MEKA es un software multi-label desarrollado sobre WEKA. Como los anteriores, también ha asociado un repositorio de datos con más de 20 MLD. El formato de archivo también está basado en ARFF, y la información necesaria para saber qué atributos son las etiquetas, específicamente el número de etiquetas, se incluye en el encabezado. Algunos conjuntos de datos multi-dimensionales están incluidos.
- ***RUMDR*:** R Ultimate Multilabel Dataset Repository proporciona una compilación de todos los MLD disponibles públicamente, así como un paquete R que facilita la creación de particiones y la exportación a varios formatos, incluidos MULAN, KEEL, MEKA, LibSVM y CSV.

Un grupo común de MLD está disponible en cualquiera de los repositorios antes mencionados, pero en los repositorios MULAN y MEKA, se pueden encontrar algunos específicos, no disponibles en los otros sitios.

Problemas multi-label basados en adaptación:

El enfoque de adaptación pretende preparar algoritmos de clasificación existentes para hacer que administren instancias multi-label. Los cambios que deben introducirse en los algoritmos pueden ser bastante simples o realmente difíciles, dependiendo de la naturaleza del método original y también la forma en que se va a considerar la existencia de varias etiquetas. Los métodos de adaptación de algoritmos emplean una técnica de clasificación clásica adaptada para trabajar directamente con datos multi-label (Herrera, y otros, 2016).

Hay varias propuestas de varios clasificadores basados en árboles, redes neurales, vectores de máquinas de soporte, técnicas de aprendizaje basadas en instancias y métodos

probabilísticos, entre otros. Alrededor de treinta de ellos están retratados en las siguientes cinco secciones, cada una dedicada a una de las categorías mencionadas. Una sección adicional enumera otros tipos de propuestas, como las basadas en anti colonias o algoritmos genéticos.

Métodos basados en arboles:

Los árboles de decisión (DT) se encuentran entre los modelos de clasificación más fáciles de entender. A pesar de su aparente simplicidad, algunos algoritmos DT, como C4.5 (C4.5: Programs for Machine Learning, 1993), producen un rendimiento que los hace competitivos frente a otros enfoques de aprendizaje. Los siguientes son algunos de los métodos de clasificación multi-label basados en DT propuestos en la literatura. (A Tutorial on Multi-Label Learning, 2013)

Multi-label C4.5, ML-C4.5:

La investigación realizada en (Knowledge discovery in multi-label phenotype data, 2001) tuvo como objetivo clasificar los genes de acuerdo con su función, teniendo en cuenta que el mismo gen puede intervenir en varias funciones. La solución propuesta se basa en el conocido algoritmo C4.5, modificado adecuadamente para tratar varias etiquetas a la vez. Los dos puntos clave del método adaptado son (A Tutorial on Multi-Label Learning, 2013):

- Las hojas del árbol contienen muestras que están asociadas a un conjunto de etiquetas, en lugar de a una sola clase.
- La medida de entropía original se ajusta para tener en cuenta la probabilidad de que las instancias no pertenezcan a una determinada etiqueta

El proceso iterativo para construir el árbol divide las instancias de acuerdo con el valor de un determinado atributo, calculando la suma ponderada de las entropías de cada subconjunto potencial de etiquetas. Si una muestra está asociada con más de una etiqueta, entonces se acumula varias veces en esta suma ponderada.

Como cada hoja representa un conjunto de etiquetas, esto afecta tanto al proceso de etiquetado de cada nodo en el árbol como a la tarea de poda adicional inherente a C4.5.

Predictive Clustering Trees:

Considera un árbol de decisión como una jerarquía de clúster donde los datos se dividen en una estrategia de arriba hacia abajo minimizando la varianza. Las hojas representan los grupos y están etiquetadas con el prototipo (predicción) del clúster. A diferencia de los

árboles de decisión estándar, la varianza y las funciones de prototipo se tratan como parámetros. Particularmente, en MLL la función de varianza se calcula como la suma de los índices de Gini (Breiman, L, 1996) de las variables de la tupla objetivo y la función prototipo devuelve un vector con probabilidades para cada etiqueta. Se ha utilizado en el aprendizaje jerárquico de etiquetas múltiples y obtiene un desempeño competitivo como clasificador base en conjunto con random forest. (A Tutorial on Multi-Label Learning, 2013)

Ensamblado de métodos MLL:

Madjarov (An extensive experimental comparison of methods for multi-label learning. Pattern Recognition) considera los métodos de conjunto cuyos clasificadores base son técnicas multi-label como un grupo especial, diferente tanto de la transformación de problemas como de la adaptación de algoritmos porque se desarrollan en la parte superior de estos dos enfoques. Por lo tanto, RAKEL, EPS, ECC, EPCC y CDE se pueden considerar conjuntos de transformación de problemas. Con respecto a los conjuntos de adaptación de algoritmo, vale la pena citar Random forest de clústeres predictivos (RF-PCT) en] y Random forest de ML-C4.5 (RF-C4.5), dos métodos de conjunto que usaron PCT y ML-C4.5 árboles como clasificadores de base, respectivamente. La diversidad de los clasificadores base se obtuvo al ensacar y cambiar el conjunto de características durante el aprendizaje (es decir, se consideró que un subconjunto aleatorio de características seleccionaba el mejor atributo de división). (A Tutorial on Multi-Label Learning, 2013).

Podemos encontrar varias implementaciones de la mayoría de estos algoritmos en diferentes lenguajes y herramientas. Las más populares y conocidas son MULAN y MEKA, ambas implementadas en Java utilizando Weka como base. Otras implementaciones populares las encontramos en Matlab, como por ejemplo las propuestas por el grupo LAMBDA. La tabla 2, muestra de forma resumida algunos de los algoritmos para clasificación multi-label disponible en las tres herramientas mencionadas. Aunque existen otras nos centramos en estas por ser las más conocidas.

Tabla 2: Comparación de repositorios MLDs

	MULAN	MEKA	LAMBDA GR.
RSL	Copy, ignore, select	RT	
BR-BASED	BR	BR, BRq	
LC	LP, PS, EPS	LP(LC), PS, EPS	
PW	CLR		

LABEL DEPENDENCES	CC, ECC	CC, ECC, PCC, MCC(Montecarlo)	ML-LOC
LAZY	BR*k*NN, IBLR, MLkNN		MLkNN
ANN	BPMLL, MMP		BPMLL
SVMS			
META-LEARNERS	RA*k*EL, HOMER, LPBR(Subset Learner), AdaBoost.MH, 2BR	BaggingML, 2BR(MBR), EnsembleML, RandomSubspaceML	
OTHER	Hierarchical HMC	Majority labelset, Conditional Dependence Network (CDN), unsupervised (EM)	InsDiff, MIML, WELL, Multimodal MIML
UTILS	DimensionalityReduction, ConverterLibSVM, Converter-CLUS, Iterative Stratification, Statistics of data, metrics, threes holding	Wrapper Mulan	Dimensionality reduction
MULTI-TARGET		CC, CR, Nearest Set Replacement (NSR), EnsembleMT, BaggingMT	
GUI	No	Yes	No
LANGUAJE	Java	Java	Matlab
LICENSE	GNU GPL	GNU GPL	Free for academic purpose

Otros algoritmos multi-label:

Además de los métodos descritos, existen otras formas alternativas de abordar el problema de clasificación multi-label. El más utilizado es la transformación de problema, que fue el primer enfoque utilizado para este tipo de problemas y es quizás la estrategia más simple.

La idea de la transformación es obtener un MLD y generar conjuntos de datos que puedan procesarse por clasificadores binarios o clasificadores multi-clase. Comúnmente, la salida producida por esos clasificadores tiene que ser transformada con carácter retroactivo para obtener la predicción multi-label. En otras palabras, la idea es transformar el problema en múltiples problemas de clasificación binaria o multi-clase.

Algunos de los algoritmos más conocidos que utilizan este enfoque son Binary Relevance, Label PowerSet y sus derivados.

La mayoría de los algoritmos disponibles actualmente para clasificación multi-label, así como su clasificación según el enfoque utilizado, se observan en la figura 3.

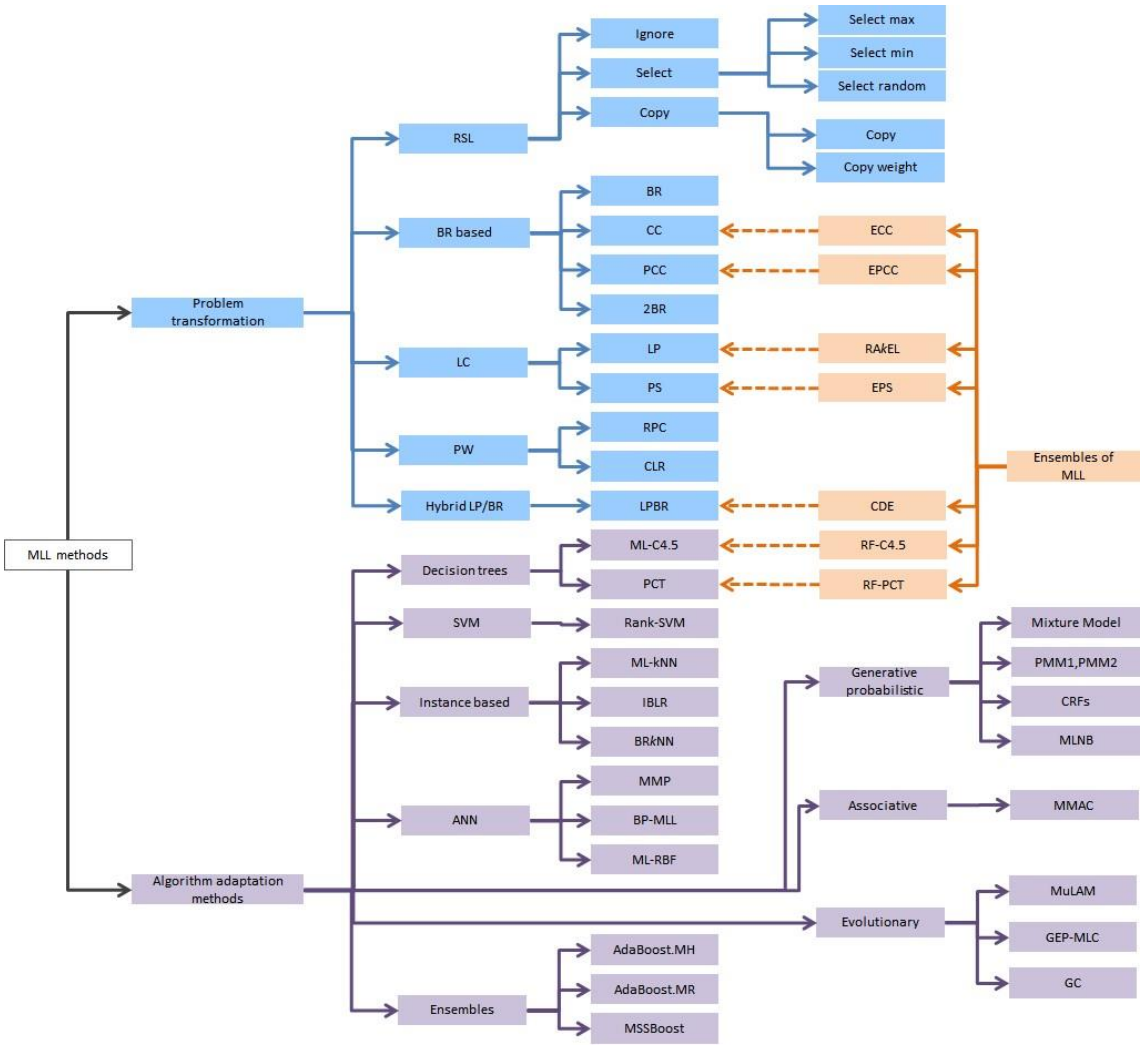


Figura 3: Métodos MLC

COMPARACIONES EXPERIMENTALES DE MÉTODOS MLL:

A pesar del número de propuestas para MLL, el desarrollo de exhaustivos estudios comparativos experimentales para obtener una mejor comprensión de los diferentes algoritmos MLL sigue siendo un tema abierto. Vale la pena citar una comparación extensa con la prueba estadística de significancia que involucra 12 métodos de MLL, 16 medidas de evaluación y 11 conjuntos de datos de referencia con diferentes escalas y de diferentes dominios (es decir, biología, multimedia y texto) (Schilesky y otros, 2015). Los mejores métodos generales fueron RF-PCT y HOMER), seguido por BR y CC. Con respecto a los clasificadores base, se usaron SVM y árboles de decisión (es decir, J48 y PCT). Como los SVM son capaces de explotar la información de todas las características, se desempeñaron mejor en conjuntos de datos con un gran número de funciones, pero pocos ejemplos, mientras que los métodos basados en árboles funcionaron mejor con conjuntos de datos más grandes. ML-kNN tuvo un rendimiento bajo en todas las medidas de evaluación.

Teniendo en cuenta la eficiencia, los métodos basados en árboles fueron más rápidos en tiempo de entrenamiento y predicción. En (Chekina et al. 2011) se propuso un enfoque de meta-aprendizaje basado en las meta-características de los conjuntos de datos para recomendar el mejor algoritmo de MLL que se utilizaría en un determinado dominio. El estudio involucró 11 algoritmos, 12 conjuntos de datos y 18 medidas. HOMER, BR, ECC y EPS obtuvieron los mejores resultados de rendimiento predictivo y se encontró que un conjunto de medidas específicas eran relevantes para recomendar un algoritmo (por ejemplo, número de etiquetas, cardinalidad de etiqueta del conjunto de entrenamiento, ejemplos promedio por clase, el número de pares de etiquetas incondicionalmente dependientes, etc.). Los resultados de esos estudios arrojan algo de luz sobre qué algoritmo seleccionar o qué algoritmos se deben tener en cuenta al desarrollar uno nuevo. La decisión final dependerá del problema a enfrentar y los requisitos a satisfacer: eficiencia, flexibilidad, desempeño predictivo, interpretabilidad del modelo, etc. (A Tutorial on Multi-Label Learning, 2013)

Métricas de modelos multi-label:

La evaluación de modelos en MLL necesita un acercamiento especial porque la actuación sobre todas las etiquetas debería ser tomada en consideración. Además, una predicción podría ser a medias correcta (una cierta cantidad de las etiquetas son correctamente predichas), completamente equivocada (todas las predicciones están mal) o completamente correcta (todas las etiquetas son correctamente predichas). En este apéndice resumiremos las métricas más frecuentes para MLL (G. Tsoumakas, I. Katakis, and I. Vlahavas, 2010) que diferencia entre dos clases de métrica: La métrica para evaluar biparticiones y la métrica para evaluar rankings.

Adoptando la misma definición del problema, $T = \{(\boldsymbol{x_i}, Y_i) | 1 \leq i \leq t\}$ es una prueba MLL con t instancias Y_i y Z_i los sets de etiquetas verdaderas y predichas para una instancia. Para cualquier predicado,π, $[\![\pi]\!]$ retorna 1 si el predicado es verdadero y 0 en caso contrario, y finalmente T_x^*es el verdadero ranking.

Métricas para evaluar Bipartición:

La métrica para evaluar biparticiones puede estar clasificada en dos grupos: basado en etiquetas y basado en ejemplos. En el primero se calcula para cada etiqueta y luego son promediados a través de todas las etiquetas (ignorando relaciones entre etiquetas) mientras

en el segundo se calcula para cada ejemplo experimental y luego promediado a través del set de prueba.

Métricas basadas en etiquetas:

Cualquier métrica de evaluación binaria puede ser usada con este tipo de acercamiento, comúnmente **precisión**, **recall**, **accuracy** y **F1-Score**. La idea computar una métrica single-label para cada etiqueta basada en el número de positivos verdaderos (tp), negativas verdaderas (tn), positivos falsos (fp) y negativas falsas (fn). Debido al hecho de tener varias etiquetas por patrón allí habrá una tabla de contingencia para cada etiqueta, así es que es necesario computar un valor medio. Dos acercamientos diferentes pueden ser usados: macro y micro. Siendo B una medida binaria de evaluación, el acercamiento macro computa una métrica para cada etiqueta y luego los valores son promediados sobre todas las categorías, mientras el acercamiento micro considera predicciones de todas las instancias conjuntamente (agrega los valores de contingencia) y luego calcula la medida a través de todas las etiquetas

$$B_{macro} = \frac{1}{q}\sum_{i=1}^{q} B(tp_i, fp_i, tn_i, fn_i)$$

$$B_{micro} = B(\sum_{i=1}^{q} tp_i, \sum_{i=1}^{q} fp_i \sum_{i=1}^{q} tn_i, \sum_{i=1}^{q} fn_i)$$

Estos dos tipos de promedios son informativos y no hay una regla de cuando usar uno o el otro, por lo que pueden usarse a conveniencia.

De acuerdo con (Yang, 2007), El macro promedió da el peso igual a cada categoría, a pesar de su frecuencia (promedio pre-categoría) y es más utilizado en categorías raras. Por otra parte, los promedios micro proporcionan el peso igual a cada ejemplo y tienden a ser usado en la mayoría de categorías comunes.

Métricas basadas en instancias:

Hamming Loss (Robert E. Schapire and Yoram Singer):

Evalúa cuántas veces, por término medio, un par de la etiqueta de ejemplo está mal-clasificada. Esta métrica tiene en cuenta errores de predicción (una etiqueta incorrecta es predicha) y errores de omisión (una etiqueta correcta no es predicha) normalizado entre

número total de clases y el número total de ejemplos. Mientras inferior sea el valor mejor será la clasificación.

$$Hamming\ loss = \frac{1}{t}\sum_{i=1}^{t}\frac{1}{q}|Z_i \Delta Y_i|$$

Recall se define por la fracción entre las etiquetas predichas correctamente sobre el total mientras, mientras que Precision es la proporción de etiquetas correctamente clasificadas de las predichas etiquetas positivas, promediadas sobre todas las instancias.

$$recall = \frac{1}{t}\sum_{i=1}^{t}\frac{|Z_i \cap Y_i|}{|Y_i|}$$

$$precision = \frac{1}{t}\sum_{i=1}^{t}\frac{|Z_i \cap Y_i|}{|Z_i|}$$

Accuracy es la proporción de etiquetas correctamente clasificadas del total de etiquetas promediadas sobre todas las instancias.

$$accuracy = \frac{1}{t}\sum_{i=1}^{t}\frac{|Z_i \cap Y_i|}{|Z_i \cup Y_i|}$$

F1-Score es la media armónica que combina Presicion y Recall previamente definidas:

$$F1 - score = \frac{1}{t}\sum_{i=1}^{t}\frac{2|Z_i \cap Y_i|}{|Z_i| + |Y_i|}$$

Proceso KDD (Knowledge Discovery in Databases):

La actividad diaria de millones de usuarios, trabajando e interactuando en negocios o instituciones, se graba digitalmente en bases de datos, conocido como sistemas de Proceso de Transacciones En Línea. Esto ha conducido a la disponibilidad de cantidades enormes de datos en toda clase de corporaciones, sin importar si estas compañías son pequeñas o grandes empresas. Extraer conocimiento útil de estos datos de manera manual es sumamente difícil, en muchos casos no imposible. Es por esto que las técnicas de Minería de Datos (MD) aumentan su popularidad como una forma automática de obtener conocimiento oculto en los datos. Este conocimiento es muy valioso para sistemas de apoyo para la toma de decisiones, para predecir valores futuros de los datos, para describir la estructura de la información, etcétera.

La MD, es una disciplina muy conocida en la actualidad, usualmente vista como una de las etapas del proceso de KDD, (Fayyad, U., Piatetsky-Shapiro, G., Smyth, P, 1996) define KDD como el proceso no trivial de identificar valores novedosos, potencialmente útiles y patrones comprensibles en los datos. Estos patrones serían el resultado obtenido al final del proceso y podrían ser de naturaleza dispar como estará posteriormente.

Extraer comprensiones nuevas y útiles de una base de datos es un proceso que puede estar dividido en múltiples etapas. Estos han sido representados en la Fig. 4. El punto de partida es entender el dominio el problema, especificando las metas a alcanzar. Después, los pasos serían:

Obtención e integración de los datos:

Los datos necesitaron cumplir los objetivos establecidos puede residir en fuentes heterogéneas, tal como bases de datos relacionales, hojas de cálculos electrónica. Una vez todos los datos han sido recogidos, tiene que ser correctamente integrado en una representación común, apropiada para siguientes pasos.

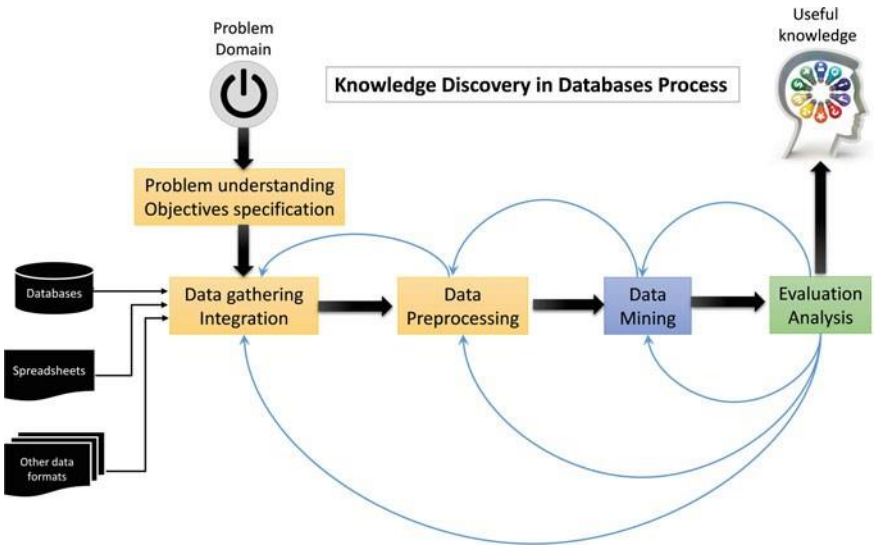


Figura 4: Etapas del proceso KDD

Pre procesamiento de los datos:

Una parte de la información inferida de fuentes de datos puede ser inconsistente y / o irrelevante. Las diferencias en las escalas, la presencia de ruido, y otras anomalías tienen que ser tratados directamente por métodos de limpieza. Quitando datos poco importantes, sólo las informaciones útiles y verdaderamente pertinentes son seleccionadas para las siguientes etapas. Además, dependiendo de su naturaleza, distintos métodos de pre

procesamiento para la reducción de datos pueden aplicados. Todos ellos tienen la intención de preparar los datos para el aprendizaje aplicado en la siguiente fase.

Minería de datos:

Este es el paso más conocido en el proceso KDD, y algunos autores (Han,J.,Kamber,M.,Pei,J., 2011) lo ven como la etapa principal en el proceso KDD. Para trabajar con los datos anteriormente integrados, limpiados, seleccionados y transformados, un algoritmo de MD tiene que estar escogido para aprender de esta información. Según los objetivos establecidos al comienzo, el algoritmo puede tener como meta agrupar las pruebas de datos según algunos atributos, adquirir un modelo capaz de automáticamente clasificar pruebas nuevas, etc.

Es un proceso donde los conjuntos de métodos, técnicas y herramientas son aplicados con el fin de realizar una búsqueda y extracción de patrones de interés. Esta etapa se puede abordar, en función del problema a resolver, desde dos puntos de vistas distintos:

- **Predictivo:** en el que se intenta obtener conocimiento para clasificación o predicción.
- **Descriptivo:** Cuyo objetivo fundamental es descubrir conocimiento de interés dentro de los datos, intentando obtener información que describa el modelo que existe detrás de los datos.

Evaluación y Análisis:

Los resultados obtenidos a partir del paso anterior tienen que ser evaluado y analizado. Interpretarlos ayudará al usuario a lograr las metas deseadas, también ayudará con la comprensión del problema global. Éste sería el conocimiento útil y poco trivial extraído de la información.

Como el diagrama en Fig.1.1 indica, todos los pasos en el proceso KDD pueden saltar atrás, dependiendo de diferentes condiciones. Como consecuencia, cada etapa puede ser aplicada varias veces hasta que un cierto estado sea encontrado, teniendo la intención de mejorar calidad de datos en cada iteración.

Aunque la MD es considerada la etapa esencial en KDD, la mayor parte del esfuerzo y el tiempo son usualmente gastados en las tareas de pre procesamiento. Estas son responsables de encargarse de problemas como pérdida de valores, presencia de ruido, selección de instancias y características, y compresión de datos.

Herramientas y tecnologías:

En el presente epígrafe se menciona las herramientas y tecnologías utilizada para el desarrollo de este proyecto.

Herramienta computacional scikit-multilearn:

La herramienta scikit-multilearn es una biblioteca de Python para realizar la clasificación de etiquetas múltiples. La biblioteca es compatible con el ecosistema scikit / scipy / numpy y utiliza matrices dispersas para todas las operaciones internas. Proporciona implementaciones nativas de Python de métodos populares de clasificación multi-label junto con el nuevo framework para división de espacios de etiquetas. Incluye métodos de detección de comunidades basadas en gráficos que utilizan la poderosa biblioteca igraph para extraer información de dependencia de etiquetas. Además, scikit-multilearn sigue la idea de construir una biblioteca de clasificación multitarea sobre una solución de clasificación existente. El objetivo principal de scikit-multilearn es proporcionar un Implementación eficiente de Python de tantos algoritmos de clasificación multi-label como sea posible para la comunidad (SZYMAŃSKI, P. y KAJDANOWICZ, T, 2017)

La biblioteca proporciona envoltorios compatibles con scikit a bibliotecas de referencia como MEKA, MULAN y WEKA a través de MEKA cuando surge la necesidad de utilizar métodos que aún no se han implementado en Python de forma nativa.

Lenguaje de programación:

Un lenguaje de programación es un conjunto de reglas, notaciones, símbolos y/o caracteres que permiten a un programador poder expresar el procesamiento de datos y sus estructuras en la computadora.

En esta investigación se utilizó Python, que en (ALVAREZ, M.A., 2003) se define como un lenguaje de scripting independiente de plataforma y orientado a objetos, preparado para realizar cualquier tipo de programa, desde aplicaciones Windows a servidores de red o incluso, páginas web. Es un lenguaje interpretado, lo que significa que no se necesita compilar el código fuente para poder ejecutarlo, lo que ofrece ventajas como la rapidez de desarrollo e inconvenientes como una menor velocidad. Se estará utilizando específicamente la plataforma scikit-multilearn.

Conclusiones:

Se dejaron fundamentados los conceptos relacionados con los algoritmos de predicción multi-label basados en adaptación del estado del arte. Se evidenció la utilidad del uso de dichos algoritmos en la actualidad para la resolución de problemas de la vida práctica estudiando diversas aplicaciones que forman parte del conjunto de problemas en este campo. Además, se describieron los métodos para evaluar los resultados luego de implementar los algoritmos propuestos, así como las herramientas y tecnologías a utilizar en la investigación.

Capitulo2:" Propuesta de solución":

Introducción:

En este capítulo se realiza una descripción teórica del algoritmo ML-C4.5 y RF-C4.5, además del modelo conceptual de la propuesta de solución y los requerimientos que necesita la misma para poderse llevar a cabo. También las especificidades de su implementación.

Algoritmos:

En este epígrafe se presenta una descripción detallada de la propuesta de algoritmos ML-C4.5 y RF-4.4.

ML-C4.5:

Es un algoritmo de inducción que genera una estructura de reglas o árbol a partir de subconjuntos (ventanas) de casos extraídos del conjunto total de datos de "entrenamiento". En este sentido, su forma de procesar los datos es parecido al de Id3. El algoritmo genera una estructura de reglas y evalúa su "bondad" usando criterios que miden la precisión en la clasificación de los casos.

Algunas premisas guían este algoritmo, son las siguiente (C4.5: Programs for Machine Learning, 1993):

- Si todos los casos son de la misma clase, el árbol es una hoja y la hoja se devuelve etiquetada con esta clase
- Para cada atributo, calcule la información potencial proporcionada por una prueba en el atributo (basado en las probabilidades de que cada caso tenga un valor particular para el atributo). Calcule también la ganancia en información que resultaría de una prueba en el atributo (basada en las probabilidades de cada caso con un valor particular para el atributo de una clase particular).
- dependiendo del criterio de selección, encuentre el mejor atributo para ramificar.

Ganancia de información:

Es la diferencia entre la entropía de todo el conjunto de ejemplos de entrenamiento restantes y la suma ponderada de la entropía de los subconjuntos causados por la partición en los valores de ese atributo

$$informatio\_gain(S, A) = -entropy(S) - \sum_{v \in A} \frac{|S_v|}{|S|} * \log(S_v)$$

donde A es el atributo que se considera, S es el conjunto de ejemplos de entrenamiento que se consideran, y S_v es el subconjunto de S con el valor v para el atributo A.

Las etiquetas múltiples son un problema para C4.5, y todos los métodos clásicos de aprendizaje, ya que esperan que cada ejemplo se etiquete como perteneciente a una sola clase. Y en la actualidad existen problemas en el que una instancia puede pertenecer a varias clases diferentes. En el caso de una etiqueta de clase única para cada ejemplo, la entropía para un conjunto de ejemplos es:

$$entropy(S) = -\sum_{i=1}^{N} (p(c_i)\log p(c_i)) + (q(c_i)\log q(c_i))$$

Donde:

$p(c_i)$ = probabilidad (frecuencia relativa) de la clase c_i.

$q(c_i)$ = 1- $p(c_i)$ probabilidad de no ser miembro de la clase c_i

La información después de una partición según algún atributo, se puede calcular como una suma ponderada de la entropía para cada subconjunto (calculada como arriba), donde esta vez, suma ponderada significa si un elemento aparece dos veces en un subconjunto porque pertenece a dos clases, luego lo contamos dos veces.

Para generar reglas a partir del árbol de decisiones, esto se puede hacer de la manera habitual, excepto cuando se trata de que una hoja sea un conjunto de clases, se generará una regla por separado para cada clase, antes de la regla- poda parte del algoritmo C4.5rules. Podríamos haber generado reglas que simplemente generen un conjunto de clases: fue una elección arbitraria generar reglas separadas, elegidas para la comprensión de los resultados.

Re-muestreo:

Todas las mediciones de precisión se realizaron utilizando la m-estimación [9], que es una generalización de la estimación de Laplace, teniendo en cuenta la probabilidad a priori de la clase. La m-estimación para la regla $r(M(r))$es:

$$M(r) = \frac{p + m\dfrac{P}{p+N}}{p+n+m}$$

Donde:

P = número total de ejemplos positivos,

N = número total de ejemplos negativos.

p = número de ejemplos positivos cubiertos por la regla r,

n = número de ejemplos negativos cubiertos por la regla r

m = parámetro a alterar.

Usando esta fórmula, la precisión de las reglas con cobertura cero será la probabilidad a priori de la clase. M, es un parámetro que puede alterarse para ponderar la probabilidad a priori. Usamos m = 1.

RF-C4.5:

Random Forest clasificador que consiste en una colección de clasificadores de árboles de decisión $\{h(x, \theta_k), k = 1 \ldots\}$ donde $\{\theta_k\}$ son vectores independientes distribuidos de forma aleatoria y cada árbol emite un voto unitario para la clase más popular de entrada x. El bosque elige la clasificación que tenga más votos sobre todos los árboles en el bosque.

Cada árbol crece de la siguiente manera:

1. Si el número de casos en el conjunto de entrenamiento es N, muestrea N casos al azar, pero con reemplazo, de los datos originales. Esta muestra será el conjunto de entrenamiento para hacer crecer el árbol.
2. Si hay M variables de entrada, se especifica un número m < M tal que, en cada nodo, m variables se seleccionan al azar fuera de la M y la mejor división en estas se usa para dividir el nodo. El valor de m se mantiene constante durante el crecimiento del bosque.
3. Cada árbol crece en la mayor medida posible. No hay poda.

La tasa global de error forestal depende de dos cosas:

- La correlación entre dos árboles en el bosque. Aumentando la correlación aumenta la tasa de error del bosque.
- La fuerza de cada árbol individual en el bosque. Un árbol con una baja tasa de error es un clasificador fuerte. Aumentar la fuerza de los árboles individuales disminuye la tasa de error del bosque.

Después de construir cada árbol, todos los datos se ejecutan en el árbol y las proximidades se calculan para cada par de casos. Si dos casos ocupan el mismo nodo terminal, su proximidad se incrementa en uno. Al final de la carrera, las proximidades se normalizan

dividiendo por el número de árboles. Las proximidades se utilizan para reemplazar datos faltantes, localizar valores atípicos y producir vistas iluminadoras de baja dimensión de los datos.

Para formalizar el funcionamiento del Random Forest deje que el bosque contenga K árboles clasificadores $h_1(x), h_2(x) \dots h_k(x)$ y el clasificador conjunto sea $h(x)$. Cada instancia de aprendizaje está representada por un par ordenado (x, y) donde cada vector de atributos x consiste en atributos individuales $A_i i = 1 \dots a$ (a es el número de atributos) está etiquetado con el valor objetivo $Y_j, j = 1 \dots c$ (c es el número de clases), la clase correcta se denota como y. Cada atributo discreto A_i tiene valores v_i a través de v_{m_i}(m_i es el número de valores del atributo A_i). Denotamos $p(v, k)$ a la probabilidad que el atributo A_i tome valor v_k, $p(Y_j)$ es la probabilidad de la clase Y_j y $p(Y_j|v_{i,k})$ es la probabilidad de que la clase Y_j este condicionada por el atributo A_i teniendo el valor v_k.

Cada conjunto de entrenamiento de n instancias se dibuja al azar reemplazando n instancias del conjunto de entrenamiento de. Con este muestreo llamado replicación de arranque, en promedio 36.8% de las instancias de entrenamiento no se usan para construir cada árbol. Estas instancias fuera de la bolsa son útiles para calcular una estimación interna de la fuerza y a correlación del bosque. Permitir un conjunto de instancias out-of-bag para el clasificador. $h_k(x)$ como $O_k(x)$. $Q(x, Y_j)$ es la proporción de votos out-of-bag para la clase Y_j en entrada x y una estimación de $P(h(x) = Y_j)$:

$$Q(x, Y_j) = \frac{\sum_{k=1}^{K} I(h_k(x) = Y_j; (x, y) \in O_k)}{\sum_{k=1}^{K} I(h_k(x); (x, y) \in O_k)}$$

donde $I()$ es la función indicadora.

Calculamos la función de margen que mide el grado en que el voto promedio para la clase correcta Y excede el voto promedio para cualquier otra clase de la siguiente manera:

$$mr(x, y) = P(h(x) = y) - \max_{\substack{j=1 \\ j \neq y}} P(h(x) = Y_j)$$

se estima con $Q(x, y)$ y $Q(x, Y_j)$. La fuerza se define como el margen esperado, y se calcula como el promedio sobre el conjunto de entrenamiento:

$$s = \frac{1}{n} \sum_{i=1}^{n} \left[Q(x_i, y) - \max_{\substack{j=1 \\ j \neq y}} Q(x_i, Y_j) \right]$$

La correlación promedio se calcula como la varianza del margen sobre el cuadrado de la desviación estándar del bosque:

$$\bar{p} = \frac{var(mr)}{(sd(h()))^2} = \frac{\frac{1}{n}\sum_{i=1}^{n}\left[Q(x_i, y) - \max_{\substack{j=1 \\ j\neq y}} Q(x_i, Y_j)\right]}{\left[\frac{1}{k}\sum_{t=1}^{k}\sqrt{p_k + \hat{p}} + [p_k - \hat{p}]^2\right]^2}$$

Donde:

$$p_k = \frac{\sum_{(x,y)\in O} I(h_k(x) = y)}{\sum_{(x_i,y)\in O} I(h_k(x))}$$

es un *out-of-bag* estimado de $p(h_k(x) = y)$ y

$$\hat{p}_k = \frac{\sum_{(x_i,y)\in O} I\left[h_k(X) = \hat{Y}_j\right]}{\sum_{k=1}^{k} I(h_k(x),(x,y)\in O)}$$

es un *out-of-bag* estimado de $P\left[h_k(x) = \hat{Y}_j\right]$ donde

$$\hat{Y}_j = \arg\max_{\substack{j=1 \\ j\neq y}} Q(x, Y_j)$$

se estima para cada instancia x en el conjunto de entrenamiento $\mathrm{Q(x, Y_j)}$.

Breiman usó árboles de decisión sin podar como clasificadores básicos e introdujo aleatoriedad adicional en los árboles (Improving Random Forests, 2004, pp. 359-370.). En cada nodo interior de cada árbol, un subconjunto de r atributos se selecciona y evalúa al azar con la heurística del índice de Gini. El atributo con el índice de Gini más alto se elige como dividido en ese nodo.

En los problemas de clasificación, los métodos de evaluación de atributos son el índice de Gini (Classification and regression trees, 1984.), la relación de ganancia (C4.5: Programs for Machine Learning, 1993), ReliefF (Estimating attributes: analysis and extensions of Relief, 1994), MDL (On biases in estimating multi-valued attributes, 1995) y KDM (.Applying the weak learning framework to understand and improve C4.5., 1996). Random Forests usa el índice de Gini tomado del sistema de aprendizaje CART (Classification and regression trees, 1984.). El índice de gini viene dado por la fórmula

$$Gini(A_t) = \sum_{i=1}^{c} p(Y_i)^2 + \sum_{j=1}^{m_i} p(v_{i,j}) \sum_{i=1}^{c} P(^{Y_i}/_{v_{i,j}})^2$$

Algoritmo

Entrada: Set de entrenamiento T con secuencia e m etiquetas $< (x_i, y_i) \ldots (x_m, y_m) >$ donde las etiquetas $y_i \in Y = \{1 \ldots k\}$. Random Forest como aprendizaje base RF especificando el número de iteraciones T

Inicializar $D_l(i) = {}^1/_m$ para toda i, inicializar los pesos

$rf_{final} = 0$; // Hipótesis final

$D_l total = 0$

$for\ i = 1; i \leq m; i++$

$D_l total = D_l total + D_l(i)$

$for\ i = 1; i \leq T; i++$

// Calcular los pesos normalizados

$for\ i = 1; i \leq m; i++$

$D_l total = {}^{D_l(i)}/_{D_l total}$

// llamar a Random Forest para obtener la hipótesis ht

K= número de árboles generados en el Random Forest

$for\ i = 1; i \leq m; i++\{$

Generar vector O_k

Construir árbol $h(x_i, O_k)$ con el algoritmo ML-C4.5

Cada árbol arroja 1 voto para la selección de las clases.

}

La selección final de las clases para una fila de prueba se selecciona por mayoría de votos.
Retornar hipótesis h_t

//calcular error en h_t
$\mathcal{E}_t = 0$
$for\ i = 1; i \leq m; i++$

$if(h_t(x_i) \neq y_i)$

$\mathcal{E}_t = \mathcal{E}_t + D_t(i)$

$if\ \mathcal{E}_t < {}^1/_2$

$T = t + 1$

else

$\beta_t = {}^{\mathcal{E}_t}/_{1 - \mathcal{E}_t}$ //Romper el bucle/

// actualizar la distribución D_t
$if(h_t(x_i) = y_i)$

$D_{t+1}(i) = \frac{D_t}{D_l(i)} * \beta_t$

else

$D_{t+1}(i) = 1$

//actualizar la hipótesis final
$rf_{final} = rf_{final} + \log({}^1/_{\beta_t})$

Salida: Hipótesis final.
$for\ (y = 1; y \leq k; y++)$

$$rf_{final} = \arg\max_{y \in Y} rf_{final}$$

Arquitectura de Sistema:

La herramienta scikit-multilearn no presenta interfaz gráfica (GUI), solo presenta interfaz de programación de aplicaciones (API). A la misma se le agregaron los algoritmos ML-C4.5 y RF-C4.5 en el paquete de algoritmos basados en adaptación (adapt). Como se muestra en la siguiente figura:

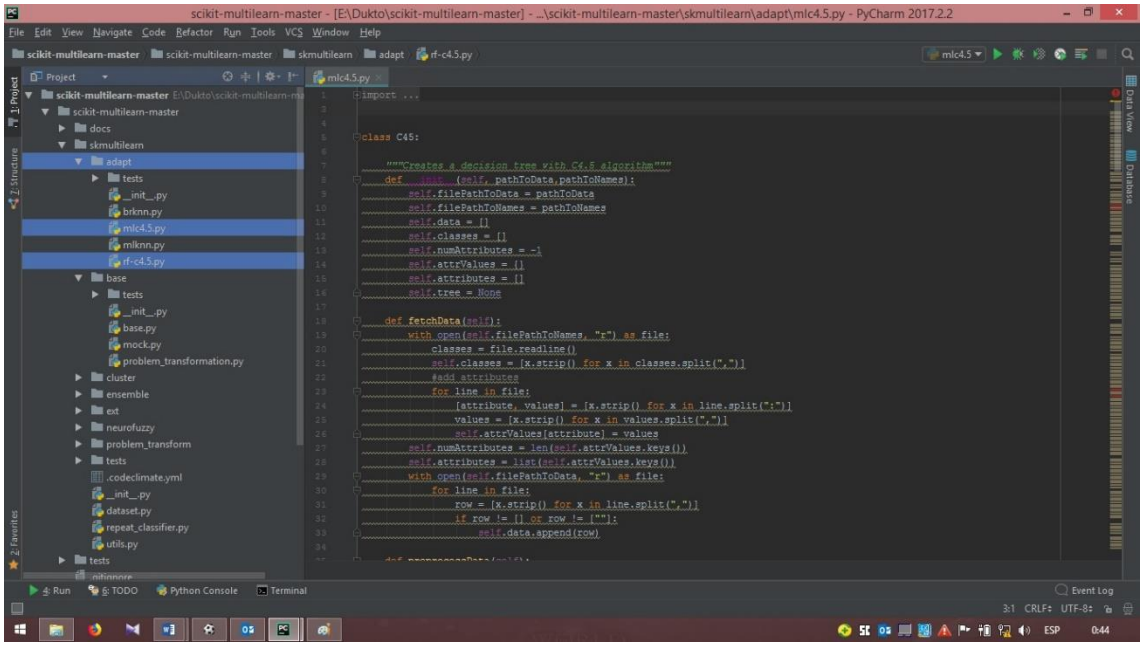


Figura 5: Integración de los algoritmos en la plataforma scikit-multilearn

Siguiendo la siguiente estructura de paquetes:

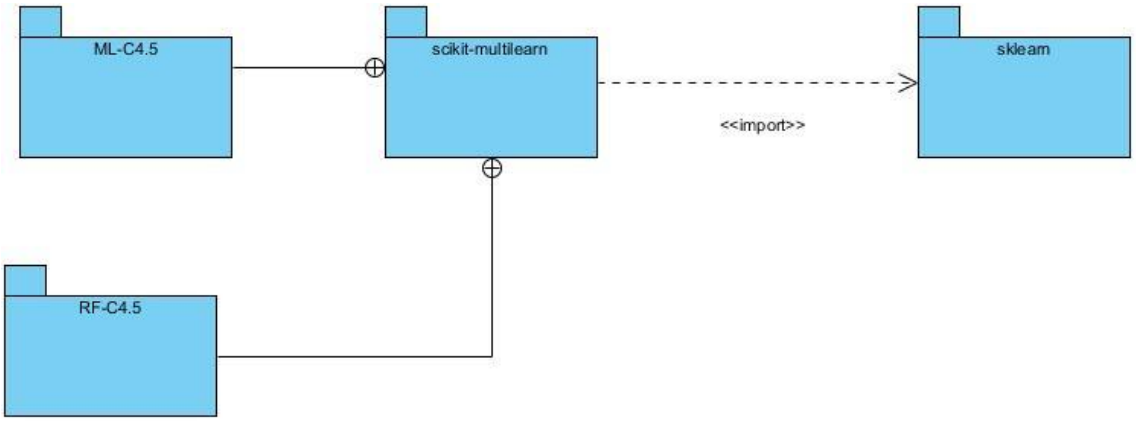


Figura 6: Diagrama de paquetes

Modelo Conceptual:

La línea base que se propone como solución para los algoritmos ML-C4.5 y RF-Cf.5 se rigen por la arquitectura de los sistemas clásicos de aprendizaje automático el cual cuenta de dos procesos fundamentales: entrenamiento y clasificación.

En el proceso de entrenamiento se aprende un modelo matemático a partir de un conjunto de datos representativos de un problema real. Estos datos han sido colectados como parte de la información histórica del problema, el cual queda definido en forma de variables conocidas como predictores. Por otra parte, esta colección de datos puede presentar, en algunos de los casos del conjunto de datos, valores anómalos, valores ausentes (Valores que no han sido medidos para una variable) o variables en diferentes escalas de medición lo cual es muy común en estos problemas.

La siguiente ilustración ejemplifica el proceso de evaluación de los algoritmos propuesto en el presente trabajo.

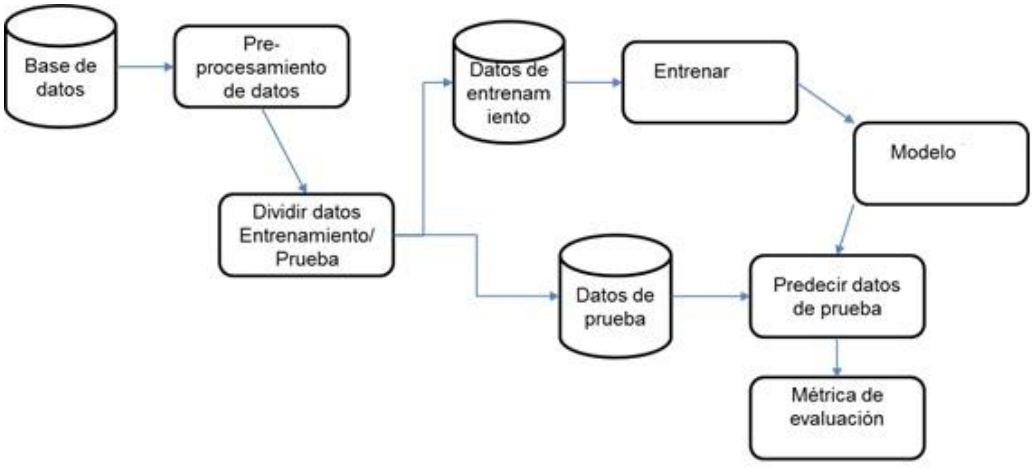


Figura 7: Proceso de entrenamiento y evaluación.

Estándares de codificación:

Este epígrafe se brindan las convenciones de escritura de código Python abarcando la librería estándar en la principal distribución de Python.

Diseño del código:

Usa 4 (cuatro) espacios por indentación.

Las líneas de continuación deben alinearse verticalmente con el carácter que se ha utilizado (paréntesis, llaves, corchetes) o haciendo uso de la “hanging indent” (aplicar tabulaciones en todas las líneas con excepción de la primera). Al utilizar este último método, no debe haber argumentos en la primera línea, y más tabulación debe utilizarse para que la actual se entienda como una (línea) de continuación.

```
# Alineado con el paréntesis que abre la función
foo = long_function_name(var_one, var_two,
                         var_three, var_four)

# Más indentación para distinguirla del resto de las líneas
def long_function_name(
        var_one, var_two, var_three,
        var_four):
    print(var_one)
```

Figura 8: Estándares de codificación, Diseño de código: ejemplo 1

El paréntesis / corchete / llave que cierre una asignación debe estar alineado con el primer carácter que no sea un espacio en blanco:

```
my_list = [
    1, 2, 3,
    4, 5, 6,
    ]
result = some_function_that_takes_arguments(
    'a', 'b', 'c',
    'd', 'e', 'f',
    )
```

Figura 9: Estándares de codificación, Diseño de código: ejemplo 2

O puede ser alineado con el carácter inicial de la primera línea:

```
my_list = [
    1, 2, 3,
    4, 5, 6,
]
result = some_function_that_takes_arguments(
    'a', 'b', 'c',
    'd', 'e', 'f',
)
```

Figura 10: Estándares de codificación, Diseño de código: ejemplo 3

¿Tabulaciones o espacios?

Nunca mezcles tabulaciones y espacios.

El método de indentación más popular en Python es con espacios. El segundo más popular es con tabulaciones, sin mezclar unos con otros. Cualquier código indentado con una mezcla de espacios y tabulaciones debe ser convertido a espacios exclusivamente. Al iniciar la línea de comandos del intérprete con la opción "-t", informa en modo de advertencias si se está utilizando un código que mezcla tabulaciones y espacios. Al utilizar la opción "-tt", estas advertencias se vuelven errores. ¡Estas opciones son altamente recomendadas!

Líneas en blanco:

Separa funciones de alto nivel y definiciones de clase con dos líneas en blanco.

Definiciones de métodos dentro de una clase son separadas por una línea en blanco.

Líneas en blanco adicionales pueden ser utilizadas (escasamente) para separar grupos de funciones relacionadas. Se pueden omitir entre un grupo (de funciones) de una línea (por ejemplo, un conjunto de implementaciones ficticias).

Usa líneas en blanco en funciones, escasamente, para indicar secciones lógicas.

Importaciones:

Las importaciones deben estar en líneas separadas, por ejemplo:

```
Sí:  import os
     import sys

No:  import sys, os
```

Figura 11: Estándares de codificación: Importaciones

Las importaciones siempre se colocan al comienzo del archivo, simplemente luego de cualquier comentario o documentación del módulo, y antes de globales y constantes. Las importaciones deben estar agrupadas en el siguiente orden:

1. importaciones de la librería estándar
2. importaciones terceras relacionadas
3. importaciones locales de la aplicación / librería

Espacios en blanco en Expresiones y Sentencias

Evita usar espacios en blanco extraños en las siguientes situaciones:

- Inmediatamente dentro de paréntesis, corchetes o llaves:

```
Sí: spam(ham[1], {eggs: 2})
No: spam( ham[ 1 ], { eggs: 2 } )
```

Figura 12: Estándares de codificación: Espacios en blanco en Expresiones y Sentencias, ejemplo1

- Inmediatamente antes de una coma, un punto y coma o dos puntos:

```
Sí: if x == 4: print x, y; x, y = y, x
No: if x == 4 : print x , y ; x , y = y , x
```

Figura 13: Estándares de codificación: Espacios en blanco en Expresiones y Sentencias, ejemplo2

- Inmediatamente antes del paréntesis que comienza la lista de argumentos en la llamada a una función:

```
Sí: spam(1)
No: spam (1)
```

Figura 14: Estándares de codificación: Espacios en blanco en Expresiones y Sentencias, ejemplo3

- Inmediatamente antes de un corchete que empieza una indexación o “slicing” (término utilizado tanto en el ámbito de habla inglesa como española):

```
Sí: dict['key'] = list[index]
No: dict ['key'] = list [index]
```

Figura 15: Estándares de codificación: Espacios en blanco en Expresiones y Sentencias, ejemplo4

- Más de un espacio alrededor de un operador de asignación (u otro) para alinearlo con otro:

```
x = 1
y = 2
long_variable = 3
```

Figura 16: Estándares de codificación: Espacios en blanco en Expresiones y Sentencias, ejemplo5

- No uses espacios alrededor del = (igual) cuando es utilizado para indicar un argumento en una función ("*keyword argument*") o un parámetro con un valor por defecto.

```
def complex(real, imag=0.0):
    return magic(r=real, i=imag)
```

Figura 17: Estándares de codificación: Espacios en blanco en Expresiones y Sentencias, ejemplo6

- Mientras que en algunas ocasiones es aceptado el colocar un if/for/while con un cuerpo pequeño en la misma línea, nunca lo implementes para sentencias de múltiples cláusulas.

Comentarios:

Comentarios que contradigan el código es peor que no colocar comentarios. ¡Siempre realiza una prioridad de mantener los comentarios al día cuando el código cambie!

Los comentarios deben ser oraciones completas. Si un comentario es una frase u oración, su primera palabra debe comenzar con mayúscula, a menos que sea un identificador que comienza con minúscula. ¡Nunca cambies las mayúsculas/minúsculas de los identificadores! (Nombres de clases, objetos, funciones, etc.).

Si un comentario es corto, el punto al final puede omitirse.

Comentarios en bloque generalmente consisten en uno o más párrafos compuestos por oraciones completas, por lo que cada una de ellas debe finalizar en un punto.

Comentarios en bloque:

Los comentarios en bloque generalmente se aplican a algunos (o todos) códigos que los siguen, y están indentados al mismo nivel que ese código. Cada línea de un comentario en bloque comienza con un # (numeral) y un espacio (a menos que esté indentado dentro del mismo comentario).

Los párrafos dentro de un comentario en bloque están separados por una línea que contiene únicamente un # (numeral).

Comentarios en línea:

Usa comentarios en línea escasamente.

Un comentario en línea es aquel que se encuentra en la misma línea que una sentencia. Los comentarios en línea deberían estar separados por al menos dos espacios de la sentencia. Deberían empezar con un # (numeral) seguido de un espacio.

Los comentarios en línea son innecesarios y de hecho molestos si su acción es obvia. No hagas esto:

```
x = x + 1                 # Incrementar x
```

Figura 18: Estándares de codificación: Comentarios en línea

Convenciones de nombramiento:

Las convenciones de nombramiento (o nomenclatura) de la librería Python son un poco desastrosas, por lo que nunca vamos a obtener esto completamente consistente – sin embargo, aquí hay algunos de los actualmente recomendados estándares de nombramiento. Los nuevos módulos o paquetes (incluyendo frameworks de terceros) deberían estar escritos en base a estos estándares, pero donde una librería existente tenga un estilo diferente, la consistencia interna es preferible.

Descriptivo: Estilos de nombramiento:

No hay gran cantidad de diversos estilos de nombramiento. Te ayuda a ser capaz de reconocer qué estilo de nombramiento está siendo utilizado, independientemente de para qué están usados. Las tildes no deben incluirse, tómese como una regla para mantener la traducción con la correcta ortografía. En cualquiera de los casos, no deben utilizarse caracteres con tildes para el nombramiento.

Se distinguen los siguientes estilos:

- b (una letra en minúscula)
- B (una letra en mayúscula)
- minúscula (lowercase)
- minúscula_con_guiones_bajos
- (lower_case_with_underscores)
- MAYÚSCULA (UPPERCASE)
- MAYÚSCULA_CON_GUIONES_BAJOS

- (UPPER_CASE_WITH_UNDERSCORES)
- PalabrasConMayúscula (CapitalizedWords) (“CapWords” o “CamelCase”).
- minúsculaMayúscula (mixedCase) (difiere con “CapWords” por la primera letra en minúscula).
- Palabras_Con_Mayúsculas_Con_Guiones_Bajos
- (Capitalizad_Words_With_Underscores) (¡horrible!).

Existe también el estilo de usar un prefijo único para identificar a un grupo de funciones relacionadas. Esto no es muy utilizado en Python, pero se menciona para cubrir los estilos en su totalidad. Por ejemplo, la función os.stat() retorna una tupla cuyos ítems tradicionalmente tienen nombres como st_mode, st_size, st_mtime y así. (Se realiza esto para enfatizar la correspondencia con los campos del “POSIX system call struct”, que ayuda a los programadores a estar familiarizados.).

Además, la siguiente forma precedida por un guion bajo es reconocida (esta puede ser combinada con cualquiera de las convenciones nombradas anteriormente):

- _simple_guion_bajo_como_prefijo (_single_leading_underscore): débil indicador de “uso interno”. Por ejemplo, from M import * no importa los objetos cuyos nombres comienzan con un guion bajo.
- simple_guion_bajo_como_sufijo_ (single_trailing_underscore_): utilizado como convención para evitar conflictos con un nombre ya existente, ejemplo:

```
Tkinter.Toplevel(master, class_='ClassName')
```

Figura 19: Estándares de codificación: Estilos de nombramientos

- __doble_guion_bajo_como_prefijo (__double_leading_underscore): al nombrar un atributo en una clase, se invoca el “name mangling” (dentro de la clase FooBar, __boo se convierte en _FooBar__boo; véase abajo).
- __doble_guion_bajo_como_prefijo_y_sufijo__ (__double_leading_and_trailing_underscore__): los objetos y atributos que viven en “namespaces” controlados por el usuario. Ejemplo, __init__, __import__ o __file__. Nunca inventes nombres como esos; únicamente utiliza los documentados.

Prescriptivo: Convenciones de nombramiento:

Nombres para evitar

Nunca uses los caracteres ‘l’ (letra ele en minúscula), ‘O’ (letra o mayúscula), o ‘I’ (letra i mayúscula) como simples caracteres para nombres de variables.

En algunas fuentes, estos caracteres son indistinguibles de los números uno y cero. Cuando se quiera usar ‘l’, en lugar usa ‘L’.

Nombres de paquetes y módulos:

Los módulos deben tener un nombre corto y en minúscula. Guiones bajos pueden utilizarse si mejora la legibilidad. Los paquetes en Python también deberían tener un nombre corto y en minúscula, aunque el uso de guiones bajos es desalentado (poco recomendado).

Ya que los nombres de módulos están ligados a los de los archivos, y algunos sistemas operativos distinguen caracteres entre minúsculas y mayúsculas y truncan nombres largos, es importante que sean bastante cortos – esto no será un problema en Unix, pero podrá ser un problema cuando el código es portado a una antigua versión de Mac, Windows o DOS.

Nombres de clases:

Casi sin excepción, los nombres de clases deben utilizar la convención “CapWords” (palabras que comienzan con mayúsculas). Clases para uso interno tienen un guion bajo como prefijo.

Nombres de excepciones:

Debido a que las excepciones deben ser clases, se aplica la convención anterior. De todas maneras, deberías usar un sufijo “Error” en los nombres de excepciones (en caso que corresponda a un error).

Nombres de variables globales:

(Esperemos que esas variables sean únicamente para uso dentro del módulo). Las convenciones son las mismas que se aplican para las funciones.

Los módulos que estén diseñados para usarse vía from M import * deben usar el mecanismo __all__ para prevenir la exportación de las variables globales, o usar la antigua convención especificando un guion bajo como prefijo (lo que tratarás de hacer es indicar esas globales como “no públicas”).

Nombres de funciones:

Las funciones deben ser en minúscula, con las palabras separadas por un guión bajo, aplicándose éstos tanto como sea necesario para mejorar la legibilidad.

“mixedCase” (primera palabra en minúscula) es aceptado únicamente en contextos en donde éste es el estilo predominante (por ejemplo, threading.py) con el objetivo de mantener la compatibilidad con versiones anteriores.

Argumentos de funciones y métodos:

Siempre usa self para el primer argumento de los métodos de instancia.

Siempre usa cls para el primer argumento de los métodos de clase.

Si los argumentos de una función coinciden con una palabra reservada del lenguaje, generalmente es mejor agregar un guión bajo como sufijo antes de usar una abreviación u ortografía incorrecta. class_ es mejor que clss. (Tal vez es mejor evitar las coincidencias usando un sinónimo).

Nombres de métodos y variables de instancia:

Usa las mismas reglas que para el nombramiento de funciones: en minúscula con palabras separadas con guiones bajos, tantos como sea necesario para mejorar la legibilidad.

Usa un solo guión bajo como prefijo para métodos no públicos y variables de instancia.

Para evitar coincidencias con subclases, usa dos guiones bajos como prefijo para invocar las reglas del “name mangling” de Python.

Python cambia (“mangles”, en el texto original) estos nombres con el nombre de la clase: si la clase Foo tiene un atributo llamado __a, no puede ser accedido por Foo.__a. (Un usuario insistente aún puede acceder llamando a Foo._Foo_a.) Generalmente, el doble guion bajo como prefijo debería ser únicamente utilizado para evitar conflicto con los nombres en atributos y clases diseñados para ser parte de una subclase.

Constantes

Las constantes son generalmente definidas a nivel módulo, escritas con todas las letras en mayúscula y con guiones bajos separando palabras. Por ejemplo, MAX_OVERFLOW y TOTAL.

Diseñando para herencia:

Siempre decide si los métodos de clase o variables de instancia (atributos) deben ser públicos o no públicos. En caso de estar en una duda, elige no público; es más fácil de hacerlo público posteriormente que hacer un atributo público no público.

Los atributos públicos son aquellos los cuales esperas que los clientes no relacionados con tu clase usen, con tu compromiso de evitar cambios incompatibles con versiones anteriores. Los atributos no públicos son aquellos que no tienen la intención de ser utilizados por terceras personas; por lo que no das la garantía que éstos no sufrirán cambios o serán removidos.

No hablamos del término “privado”, ya que ningún atributo es realmente privado es Python (sin un innecesario monto de trabajo).

Otras categorías de atributos son aquellos que son parte del “API de subclase” (a menudo llamado “protegido” en otros lenguajes). Algunas clases están diseñadas para que sean heredadas, o para extender o modificar aspectos del comportamiento de la clase. Al momento de diseñar una clase de dicha índole, ten cuidado al hacer decisiones explícitas acerca de cuáles atributos son públicos, cuales son parte del API de subclase, y cuales son de uso únicamente dentro de la clase.

Con esta mentalidad, a continuación, hay algunas guías Pythonicas:

- Los atributos públicos no deben tener un guion bajo como prefijo.
- Si tu atributo público provoca problemas con una palabra reservada, agrega un guion bajo como sufijo al nombre. Esto es preferible ante una abreviación o mala ortografía. (De todas maneras, a pesar de esta regla, ‘cls’ es preferido ante cualquier variable o argumento que se conoce como una clase, especialmente el primer argumento de un método de clase.)
- Para atributos públicos de datos simples, es mejor exponer únicamente el nombre del atributo, sin complicados métodos de acceso. Ten en cuenta que Python provee una ruta fácil para futuras mejoras, encontrarás que un atributo de simples datos necesita aumentar su comportamiento funcional. En ese caso, utiliza propiedades para ocultar la implementación funcional debajo de un acceso por un atributo de simples datos.
- Si tu clase está intencionada a ser utilizada como una subclase, y tienes atributos que no quieres que dicha subclase use, considera nombrarlos con dos guiones bajos como prefijo y sin sufijos. Esto invoca el algoritmo de “name mangling” de Python, donde el nombre de la clase es insertado dentro del nombre del atributo. Esto ayuda a evitar colisiones de nombres de atributos entre clase y subclase.

Interfaces públicas e internas:

Toda compatibilidad con antiguas versiones garantiza la aplicación únicamente en interfaces públicas. Por lo tanto, es importante que los usuarios sean capaces de distinguir claramente entre interfaces públicas e interfaces internas.

Las interfaces documentadas se consideran públicas, a menos que dicha documentación especifique que se trata de interfaces internas. Todas las interfaces indocumentadas deben ser asumidas como internas.

Para un mejor soporte de introspección, los módulos deberían declarar explícitamente los nombres en su API pública usando el atributo __all__. Aplicando una lista vacía a __all__ indica que el módulo no tiene un API pública.

Incluso una vez especificado correctamente el atributo __all__, las interfaces internas (paquetes, módulos, clases, funciones, atributos u otros nombres) deben estar prefijadas con un guion bajo.

Una interfaz es también considerada interna si cualquiera de los “namespace” que contiene (paquete, módulo o clase) es considerado interno.

Los nombres importados siempre deberían considerar un detalle de implementación. Otros módulos no deben confiar en el acceso indirecto a dichos nombres importados a menos que sean parte del API del módulo y estén explícitamente documentados, como os.path o el módulo __init__ de un paquete que expone funcionalidad para submódulos.

Conclusiones:

Se realizó el diseño teórico del proyecto el cual se desarrolló guiándose por el proceso KDD. Se realizaron cada una de las etapas de este para poder realizar una extracción correcta del conocimiento y aplicar como es debida la minería de datos. Se planteó el pseudocódigo de los algoritmos para poder implementarlos.

Capítulo 3: Resultados

En este capítulo se presentarán los resultados finales de la investigación, así como la efectividad de los algoritmos propuestos en comparación con los del estado del arte.

Pruebas de software:

Se realizaron pruebas para medir la eficiencia de los algoritmos implementados, dichas arrojando resultados positivos como se describen a continuación.

Pruebas unitarias:

Las pruebas realizadas arrojaron resultados satisfactorios como se muestra a continuación.

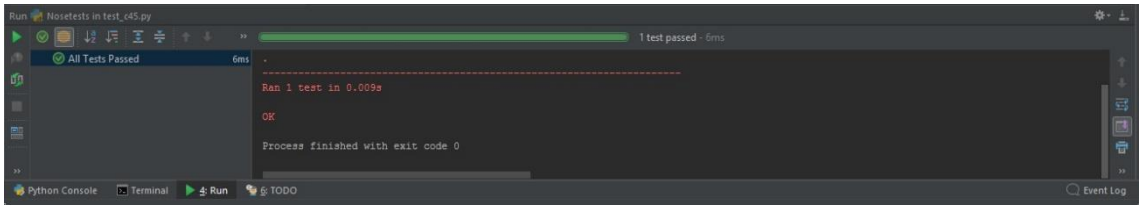


Figura 20: Resultados de prueba unitaria1

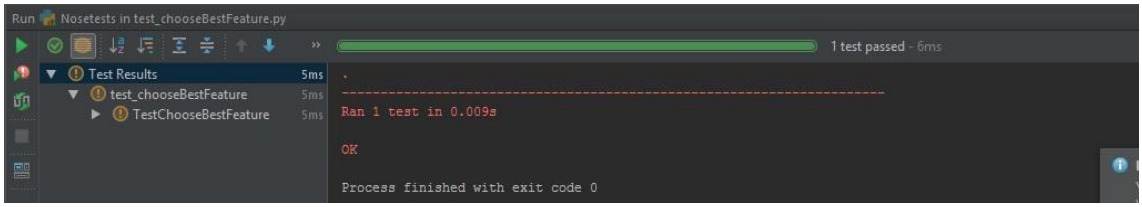


Figura 21: Resultados de prueba unitaria2

Datasets:

Utilizaremos 5 Datasets de uso común de los repositorios de Meka y Mulan.

Enron: (Read, 2008)**:** es un subconjunto de cuerpos de textos del correo electrónico Enron. Se basa en una colección de correos electrónicos permutados entre los empleados de la Corporación Enron, lo cual fuera hecho disponible durante una investigación legal. Contiene 1702 correos electrónicos que se forman 53 categorías de temas, como la estrategia de compañía, humor y asistencia jurídica.

Medical: (J.P. Pestian, C. Brew, P.M. Matykiewicz, D.J. Hovermale, N. Johnson, K.B. Cohen, and W. Duch, 2007): Se basa en los datos hechos disponible durante el Centro de Medicina Computacional 2007 Medical Natural Language Processing Challenge. Consta de 978 informes clínicos de radiología etiquetados ICD-9-CM. El dataset tiene 45 códigos y 1149 atributos.

Yeast: (Andre Elisseeff and Jason Weston, 2001): Contiene 103 características numéricas acerca de expresiones del microarray y los perfiles filogenéticos para 2417 genes de levadura.

Emotions: (K. Trohidis, 2008.): es un dataset pequeño para clasificar las emociones que provoca la música según el modelo de estado de ánimo de TellegenWatson-Clark: amazed-surprised , happy- pleased , relaxing- clam , quiet-still , sad- lonely , and angry- aggressive. Consta de 593 canciones, 6 etiquetas y de 72 características en dos categorías: rítmico y timbre.

Scene: (M. Boutell, J. Luo, X. Shen, and C. Brown., 2004.): Tiene 2407 imágenes anotadas con hasta 6 conceptos (por ejemplo, la playa, la montaña, etc). Cada uno está descrito con 294 características numéricas correspondiente a momentos espaciales de color en el espacio LUV.

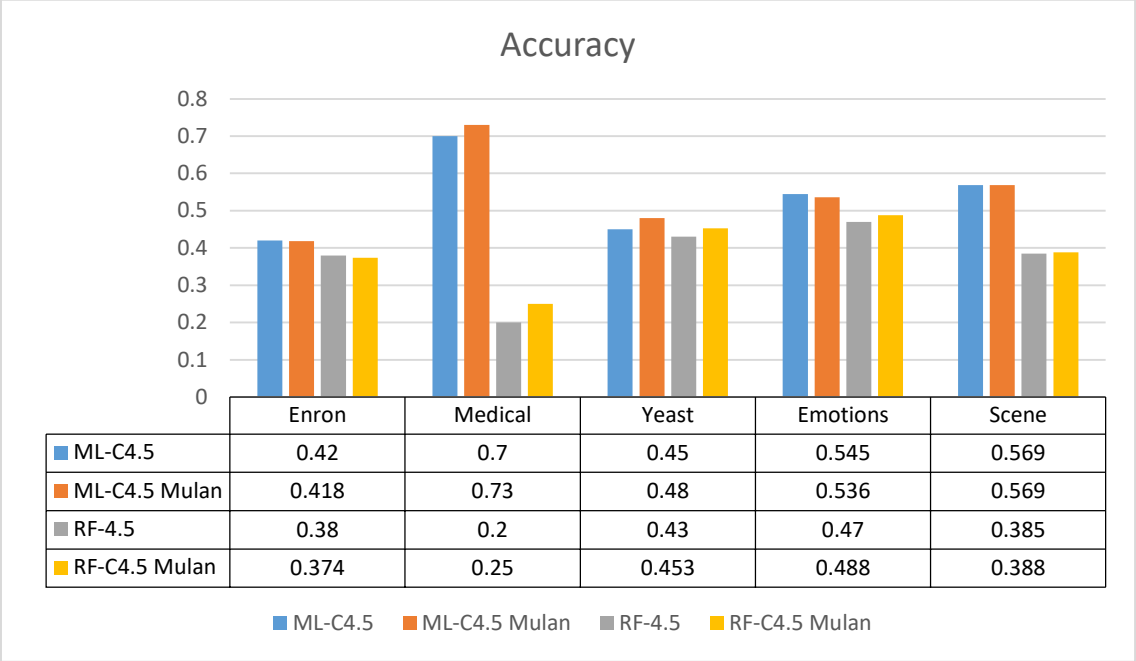
La siguiente tabla resume las estadísticas clave de estos conjuntos de datos: número de etiquetas L, número de instancias N, número de características F, número promedio de etiquetas LC, y densidad de etiqueta LC / L.

Tabla 3: Descripción de los Datasets

	L	N	F	LC	LC/L
Enron	53	1702	1001	3.378	6.37%
Medical	45	978	1149	1.245	2.76%
Yeast	14	2417	103	3.392	30.26%
Emotions	6	593	72	1.869	31.14%
Scene	6	2407	294	1.074	17.89%

Metodologías para pruebas de software:

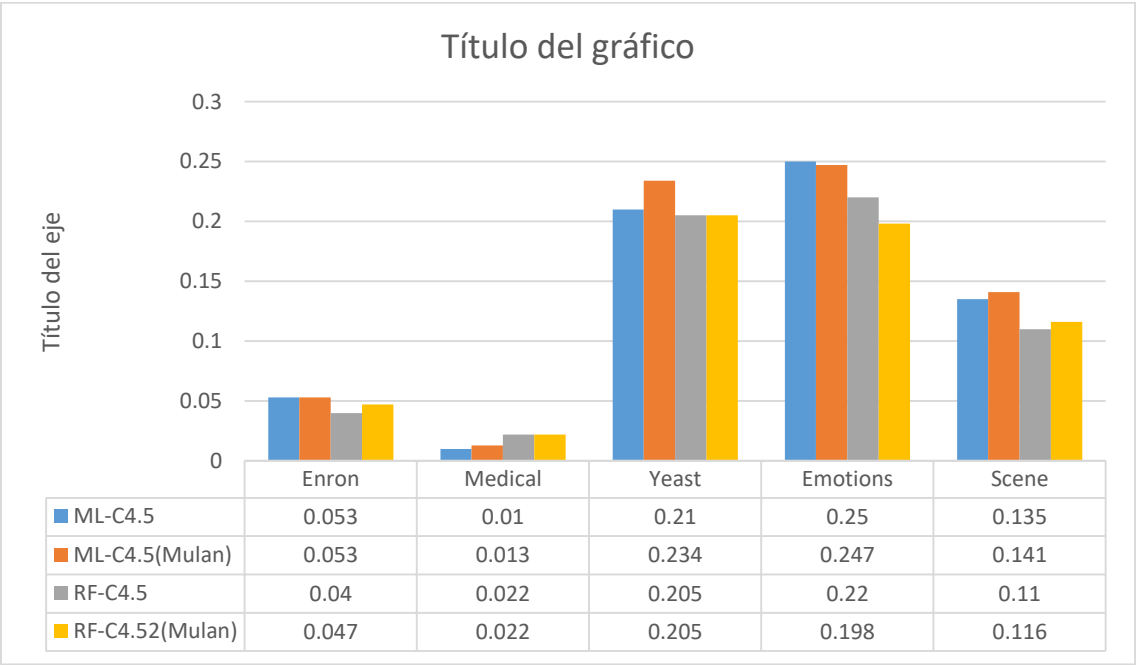
Accuracy:



	Enron	Medical	Yeast	Emotions	Scene
ML-C4.5	0.42	0.7	0.45	0.545	0.569
ML-C4.5 Mulan	0.418	0.73	0.48	0.536	0.569
RF-4.5	0.38	0.2	0.43	0.47	0.385
RF-C4.5 Mulan	0.374	0.25	0.453	0.488	0.388

Figura 22: Resultados de Accuracy

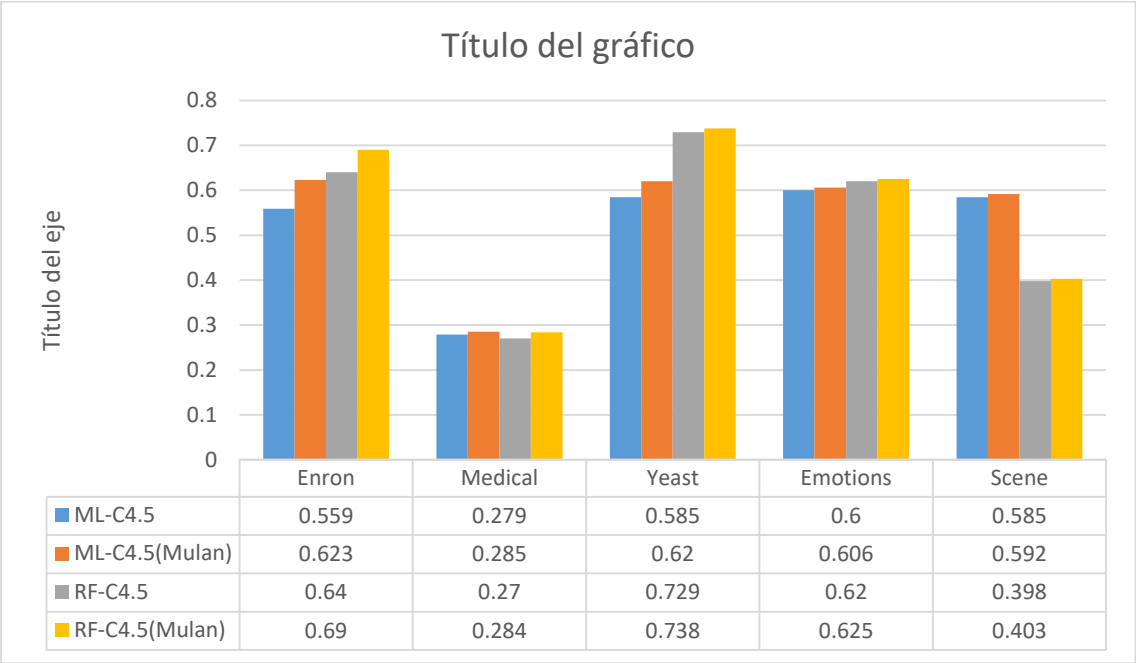
Hamming Loss:



	Enron	Medical	Yeast	Emotions	Scene
ML-C4.5	0.053	0.01	0.21	0.25	0.135
ML-C4.5(Mulan)	0.053	0.013	0.234	0.247	0.141
RF-C4.5	0.04	0.022	0.205	0.22	0.11
RF-C4.52(Mulan)	0.047	0.022	0.205	0.198	0.116

Figura 23: Resultados de Hamming Loss

Precisión:



	Enron	Medical	Yeast	Emotions	Scene
ML-C4.5	0.559	0.279	0.585	0.6	0.585
ML-C4.5(Mulan)	0.623	0.285	0.62	0.606	0.592
RF-C4.5	0.64	0.27	0.729	0.62	0.398
RF-C4.5(Mulan)	0.69	0.284	0.738	0.625	0.403

Figura 24: Resultados de Precisión

Pruebas de Friedman:

Para la realización de esta prueba se tomó las métricas *Accuracy*, *Hamming Loss* y *Precisión*, para determinar si existen diferencias significativas entre los algoritmos. Se obtuvo un χ 2 = 58.5, χ 2 = Inf, χ 2 = 2.25, con un p_value = 1.959e-07, p_value = 2.2e-16, p_value = 0.1349, respectivamente menor que β = 0.05 en las dos primeras métricas. Esto significa que se rechaza la hipótesis nula, es decir, existen diferencias significativas entre los algoritmos, en las primeras métricas y lo contrario en la última. Finalmente se obtiene un ranking de los algoritmos como se puede observar en las siguientes imágenes.

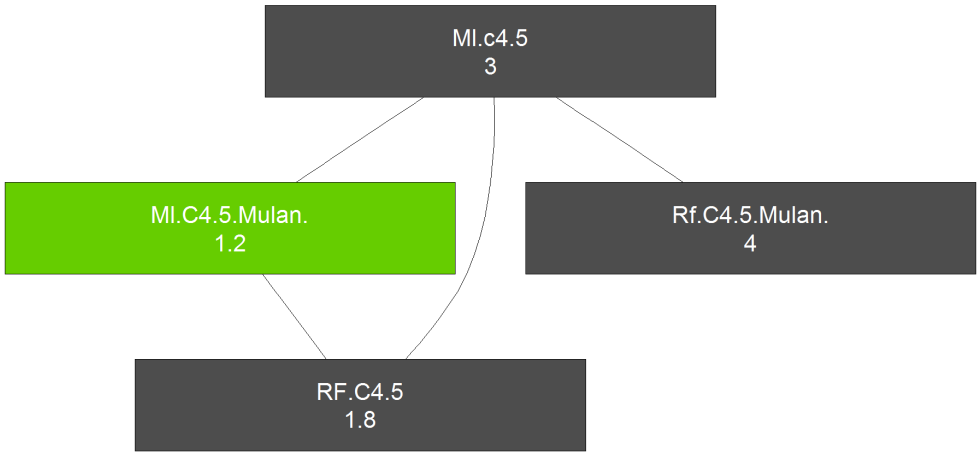


Figura 25: Ranking de Friedman para Accuracy

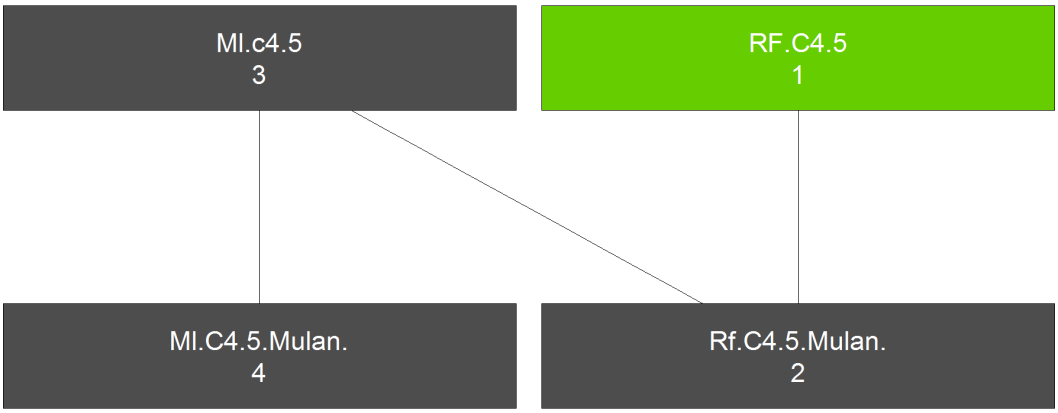


Figura 26: Ranking de Friedman para Hamming Loss

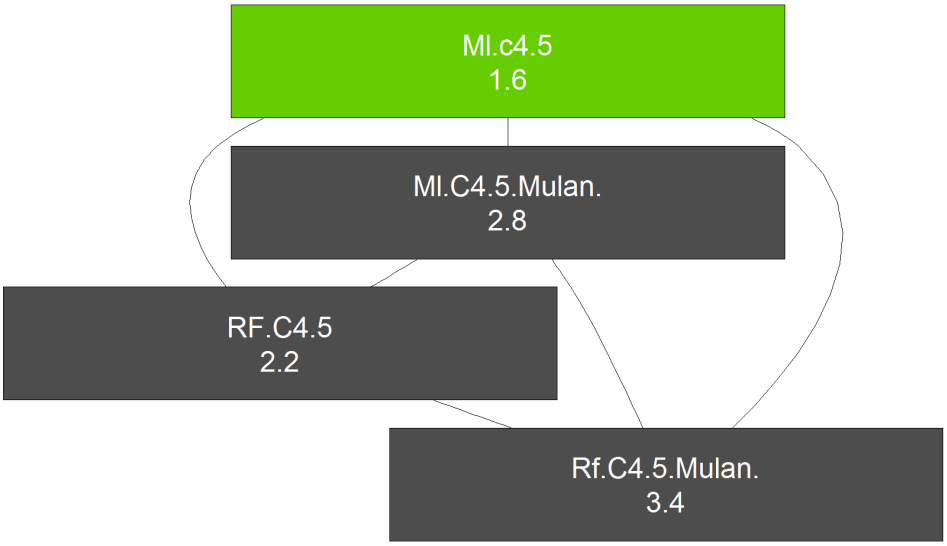


Figura 27: Ranking de Friedman para Precisión

Tras obtener los rankings para las distintas métricas 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 β = 0.05

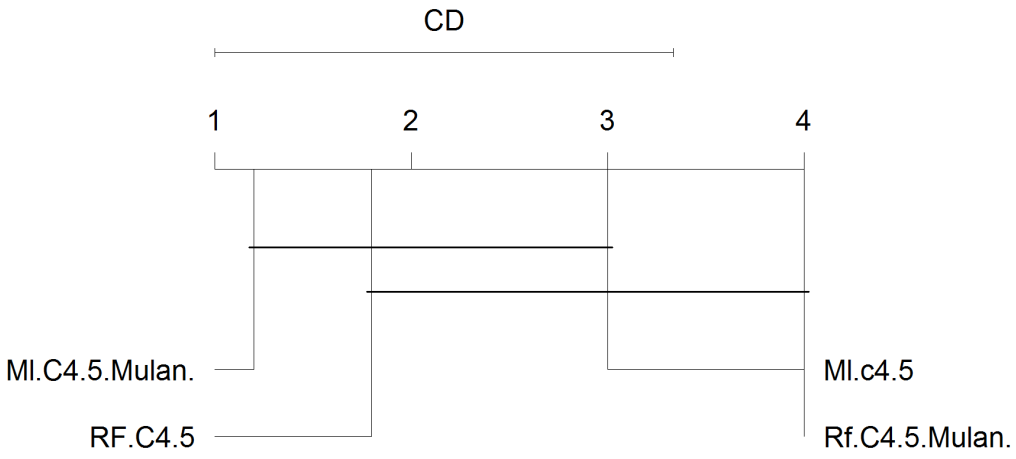


Figura 28: Prueba de Bonferroni-Dunn para Accuracy

Como se aprecia en la figura 28 los algoritmos ML-C4.5(Mulan) y RF-C4.5(Mulan) se encuentran fuera de la distancia critica por lo que presentan diferencias significativas entre

ellos, pero si comparamos los algoritmos propuestos con sus homólogos de Mulan no presentan dichas diferencias, por lo que sus funcionamientos son similares.

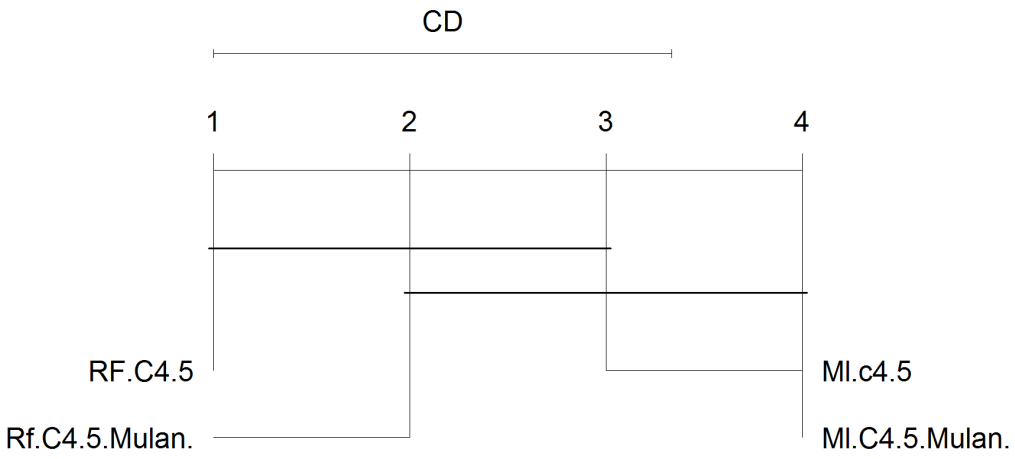


Figura 29: Prueba de Bonferroni-Dunn para Hamming Loss

Para esta métrica como muestra la figura 29 las diferencias significativas se encuentran entre los algoritmos RF-C4.5 y el ML-C4.5(Mulan), pero al igual que en el caso anterior si comparamos los algoritmos propuestos con sus homólogos de Mulan no presentan diferencias significativas.

Conclusiones:

Se probaron los algoritmos y las pruebas de software aplicadas arrojaron resultados positivos. Los test de estadísticas realizadas demuestran que existen diferencias entre los algoritmos propuestos con respecto a otros del estado del arte, pero, con respecto a sus implementaciones en otras plataformas se comportan de matera equivalente.

Conclusiones:

A modo de conclusión se puede afirmar que se dejaron fundamentados los conceptos relacionados con los algoritmos de clasificación multi-label con enfoque de adaptación, principalmente los basados en arboles de decisión. Esto evidenció la utilidad del uso de dichos algoritmos en la actualidad para la resolución de problemas de la vida práctica estudiando diversas aplicaciones que forman parte del conjunto de problemas en este campo. Además, se describieron los métodos para evaluar los resultados luego de implementar los algoritmos propuestos, así como las herramientas y tecnologías a utilizar en la investigación.

Se describió una propuesta de integración de los algoritmos ML-C4.5 y RF-C4.5 para la plataforma scikit-multilearn, siguiendo la metodología KDD, utilizando como lenguaje de programación Python, ide de desarrollo PyCharm en su versión 2017.2.2, logrando su implementación satisfactoriamente.

Las pruebas de software demostraron que la solución es robusta y los test de estadísticas aplicadas arrojaron que los algoritmos propuestos presentan diferencias significativas con algunos algoritmos del estado del arte, pero en comparación con sus homólogos de otras plataformas funcionan de manera semejante.

Referencias Bibliográficas

Aggarwal, C.C. 2014. *Data Classification: Algorithms and Applications.* s.l. : CRC Press , 2014.

Breiman, L. 1996. *Bagging predictors.* s.l. : Mach. Learn, 1996.

Chen, X., Zhan, Y., Ke, J., Chen, X. 2015. *Complex video event detection via pairwise fusion of trajectory and multi-label hypergraphs.* s.l. : Multimedia Tools Appl, 2015.

Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. 1984.*Classification and regression trees.* **.** .Belmont, California, : Wadsworth Inc, 1984. .

Cong, H., Tong, L.H. 2008. *Grouping of triz inventive principles to facilitate automatic patent classification.* s.l. : Expert Syst. Appl, 2008.

Elisseeff, A., Weston, J. 2001. *A kernel method for multi-labelled classification.* s.l. : In: Advances in Neural Information Processing Systems, 2001. MIT Press .

Fayyad, U., Piatetsky-Shapiro, G., Smyth, P. 1996. *From data mining to knowledge discovery in databases. .* s.l. : AI Mag, 1996.

Fürnkranz, J., Hüllermeier, E., Loza Mencía, E., Brinker, K. 2008. *Multilabel classification via calibrated label ranking. Mach. Learn.* 2008.

Han,J.,Kamber,M.,Pei,J. 2011. *DataMining: Concepts and Techniques.* s.l. : Morgan Jaufmann, 2011.

Sriram, B., Fuhry, D., Demir, E., Ferhatosmanoglu, H., Demirbas, M. 2010. *Short text classification in twitter to improve information filtering. In: Proceedings of 33rd international ACM SIGIR conference on Research and development in information retrieval.* s.l. : ACM , 2010.

M, Zhang and Z, Zhou. 2013.*"A review on multi-label learning algorithms",Knowledge and Data Engineering,.* s.l. : IEEE Transactions on, 2013, Vol. In Press.

Thomas G. Dietterich, Michael Kerns, and Yishay Mansour. 1996*.Applying the weak learning framework to understand and improve C4.5***.** San Francisco : Morgan Kaufmann, 1996.

Elisseeff, A., Weston, J. 2001. *A kernel method for multi-labelled classification. In: Advances in Neural Information Processing Systems,.* s.l. : MIT Press, 2001, Vol. 14.

Eva Gibaja and Sebastian Ventura. 2013.*A Tutorial on Multi-Label Learning.* 2013.

Aha, D.W. 1997. *Lazy Learning.* s.l. : Springer, 1997.

Alaonso, Jimenez, A, Jose and Guitierres Naranjo, Miguel. 2000. *Introducción al Aprendizaje Automatico.* [En linea] : s.n., 2000.

ALVAREZ, M.A. 2003. *Qué es Python.* [en línea], [Consulta: 22 enero 2018]. Disponible en: https://desarrolloweb.com/articulos/1325.php. : s.n., 2003.

Gjorgji Madjarov, Dragi Kocev, Dejan Gjorgjevikj, and Saso Dzeroski. 2012 *An extensive experimental comparison of methods for multi-label learning. Pattern Recognition.*: s.n.

André de Carvalho and Alex Freitas. 2009. . A Tutorial on Multi-label Classification Techniques. *In Foundations of Computational Intelligence Volume 5. Studies in Computational Intelligence, Vol. 205.* Springer , 2009, Berlin / Heidelberg.

Andre Elisseeff and Jason Weston. 2001 . *Kernel methods for Multi-labelled classification and Categorical regression problems. In In Advances in Neural Information Processing Systems 14. .* s.l. : MIT Press, 2001 .

Bakir, G. H., et al. 2007. Predicting Structured Data (Neural Information Processing). 2007.

Boutell, M, et al. 2004. *Learning multi-label scene classification. .* s.l. : Pattern Recogn., 2004.

Briggs, F, et al. 2012. *Acoustic classification of multiple simultaneous bird species: a multi-instance multi-label approach.* s.l. : J. Acoust. Soc. Am., 2012.

Brinker, Klaus. 2006. *On Active Learning in Multi-label Classification. In From Data and Information Analysis to Knowledge Engineering.* s.l. : Myra Spiliopoulou, Rudolf Kruse, Christian Borgelt, Andreas Nürnberger, and Wolfgang Gaul (Eds.).Springer Berlin Heidelberg, 2006.

Quinlan, J. Ross. 1993. *C4.5: Programs for Machine Learning.* San Francisco, : Morgan Kaufmann, 1993.

California, University of. 12. Random Forests. [Online] marzo 2018, 12. http://www.stat.berkeley.edu/users/breiman/RandomForests/cc_home.htm.

Cherkassky, V., Mulier, F. 2007. *Learning from Data: Concepts.* s.l. : Theory and Methods, 2007. Wiley-IEEE Press.

Rodriguez, Juan Manuel, et al. 2014. *Clasificación Multi-etiqueta Utilizando Computación Distribuida.* 978-1-4799-4270-1, Argentina : IEEE Computer Society., 2014. SBN.

Durrant, B., et al. 2013. Weka 3: Data Mining Software in Java. 2013.

Duygulu, P, et al. 2002. *Object recognition as machine translation: learning a lexicon for a fixed image vocabulary.* s.l. : Proceedings of 7th European Conference on Computer Vision. Springer, 2002. Vol. 2353.

Kononenko, Igor. 1994. *Estimating attributes: analysis and extensions of Relief.* Springer Verlag, Berlin, : Luc De Raedt and Francesco Bergadano editors, 1994, Vol. .

FAYYAD, Usama and PIATETSKY-SHAPIRO, Gregory and SMYTH, Padhraic. 1996. From data mining to knowledge discovery in databases. s.l. : AI magazine, 1996. Vol. 17, 3.

G. Tsoumakas and Katakis. 2007. *Multi Label Classification: An Overview. International Journal of Data.* 2007.

G. Tsoumakas, I. Katakis, and I. Vlahavas. 2010 . Data Mining and Knowledge Discovery Handbook, Part6. *Chapter Mining Multi-label Data, 667–685.* Springer, 2010 .

G. Tsoumakas, I. Katakis, and I. Vlahavas. 2010a. Data Mining and Knowledge Discovery Handbook, Part.

Galar, M, et al. 2011. *An overview of ensemblemethods for binary classifiers in multi-class problems: experimental study on one-vs-one and one-vs-all schemes.* s.l. : Pattern Recognitions. Elservier, 2011.

García, S., Luengo, J., Herrera, F. Data Preprocessing in Data Mining.. 2015 : Springer, Data Preprocessing in Data Mining.

Gibaja, E., Ventura, S. 2014. Multi-label learning: a review of the state of the art and ongoing research. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery.* 2014. 411–444.

Herrera, Francisco, et al. 2016. *Multilabel Classification. Problem Analysis, Metrics and Techniques.* s.l. : Springer International Publishing AG Switzerland, 2016. ISBN 978-3-319-41110-1.

M, Sikonja. 2004. *Improving Random Forests.* **pp. 359-370. .** LNAI 3210, Springer, Berlin, : In J.-F. Boulicaut et al.(Eds), 2004, pp. 359-370. , Vol. ECML 2004.

Ioannis Katakis, Grigorios Tsoumakas, and Ioannis Vlahava. 2008. *Multilabel Text Classification for Automated Tag Suggestion.* In Proceedings of the ECML/PKDD 2008 Discovery Challenge. : s.n., 2008.

J, Matas, et al. 1998. *Pattern Analysis and Machine Intelligence.* 1998.

J.P. Pestian, C. Brew, P.M. Matykiewicz, D.J. Hovermale, N. Johnson, K.B. Cohen, and W. Duch. 2007. *A shared task involving multi-label classification of clinical free text. In Proceedings of ACL BioNLP .* Prague. : s.n., 2007.

Jin, Rong, Wang, Shijung and Yang, Zhou. Regularized Distance Metric Learning: Theory and Algorithm.

K. Trohidis, G. Tsoumakas, G. Kalliris, and I. Vlahavas. 2008. *Multi-label Classification of Music into Emotions. In Proc. 9th International Conference on Music Information Retrieval (ISMIR 2008),,.* Philadelphia : s.n., 2008. .

Clare, A., King, R.D. 2001. *Knowledge discovery in multi-label phenotype data.* Springer, 2001.

Amanda Clare and Ross D. King. 2001.*Knowledge Discovery in Multi-label Phenotype Data In: Proceedings of the5thEuropeanConferencePrinciplesonDataMiningandKnowledgeDiscovery.* Springer, 2001, Vol. 2168.

M. Boutell, J. Luo, X. Shen, and C. Brown. 2004. . *Learning multi-label scene classification. Pattern Recognition .* 2004. .

Praveen Boinee, Alessandro De Angelis, and Gian Luca Foresti. 2008. *Meta Random Forests.*: World Academy of Science, Engineering and Technology International Journal of Computer and Information Engineering , 2008, Vol. 2.

Min-Ling Zhang and Zhi-Hua Zhou. . 2005. *A k-Nearest Neighbor Based Algorithm for Multi-label Classification, In Proceedings of the IEEE International Conference on Granular Computing (GrC). .* s.l. : IEEEInternational Conference on Granular Computing 2 , 2005. 718–721..

Kononenko, Igor. 1995.*On biases in estimating multi-valued attributes.* s.l. : Morgan Kaufmann,, 1995.

Diplaris, S., Tsoumakas, G., Mitkas, P., Vlahavas, I. 2005. *Protein classification with multiple algoithms. In: Proc. 10th Panhellenic Conference on Informatics.* PCI'05,, s.l. : Springer, 2005, Vols. 3746,.

12. Random Forests. *18 "" Disponible: [citado en 12 de marzo de 2008].* [Online] University of California., marzo 2008, 12. [Cited: abril 2018, 15.] http://www.stat.berkeley.edu/users/breiman/RandomForests/cc_home.htm.

Read, Jesse. 2008. *A Pruned Problem Transformation Method for Multi-label Classification.* s.l. : In Proceedings of the NZ Computer Science Research Student Conference., 2008.

Read., Jesse. 2010. *Scalable Multi-label Classification. Ph.D.Dissertation. University of Waikato.* 2010.

Robert E. Schapire and Yoram Singer. *. BoosTexter: A Boosting-based System for Text Categorization.* 2000 : s.n.

Sierra Araujo, Basilio. *Aprendizaje Automático: Conceptos básicos y avanzados.* s.l. : Pearson Education.

Snoek, C.G.M, et al. 2006. *The challenge problem for automated detection of 101 semantic concepts in multimedia.* s.l. : Proceedings of 14th ACM International Conference on Multimedia., 2006.

Sobol-Shikler, T., Robinson, P. 2010. *Classification of complex information: Inference of cooccurring affective states from their expressions in speech.* s.l. : IEEE Trans. Pattern Anal. Mach, 2010.

Sorower, Mohammad S. 2010. . A Literature Survey on Algorithms for Multi-label Learning. *Technical Report. Oregon State University. .* [Online] 2010. . http://people.oregonstate.edu/ ~ sorowerm/pdf/Qual-Multilabel-Shahed-CompleteVersion.pdf.

Sotiris Diplaris, Grigorios Tsoumakas, Pericles Mitkas, and Ioannis Vlahavas. 2005. *Protein Classification with Multiple Algorithms.* s.l. : In Proceedings of the 10th Panhellenic Conference on Informatics (PCI 2005)., 2005.

SZYMAŃSKI, P. y KAJDANOWICZ, T. 2017 . *A scikit-based Python environment for performing multi-label classification. ., .* ResearchGate [en línea]. [Consulta: 23 abril 2018]. Disponible en: https://www.researchgate.net/publication/313394357_A_scikit-based_Python : s.n., 2017 .

Tsoumakas, G., et al. 2011. Mulan: A Java library for multi-label learning. *The Journal of Machine Learning Research.* 2011. 12, pp. 2411-2414.

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

—. 2000. Weka. Machine Learning Algorithms in Java. 2000. pp. 265-320.

Wu, Q., Ye, Y., Zhang, H., Chow, T.W., Ho. 2015. *Ml-tree: a tree structure based approach to multilabel learning.* s.l. : IEEE Trans. Neural Networks Learn., 2015. Syst. 26(3), 430–443.

Yang, Liu. 2007. An Overview of Distance Metric Learning. 2007.

Z., . Sanden and J. 2011. *Enhancing multi-label music genre classification through ensemble techniques.* New York, : s.n., 2011.

Zhou, Z.H, et al. 2012. *Multi-instancemulti-labellearning.* s.l. : Artif.Intell, 2012.