Universidad de las Ciencias Informáticas
Facultad 2

Algoritmo de adaptación IBLR-ML para problemas de aprendizaje multi-etiquetas integrado a la herramienta computacional scikit-multilearn

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

Autor:

Armando Milla Izquierdo

Tutor:

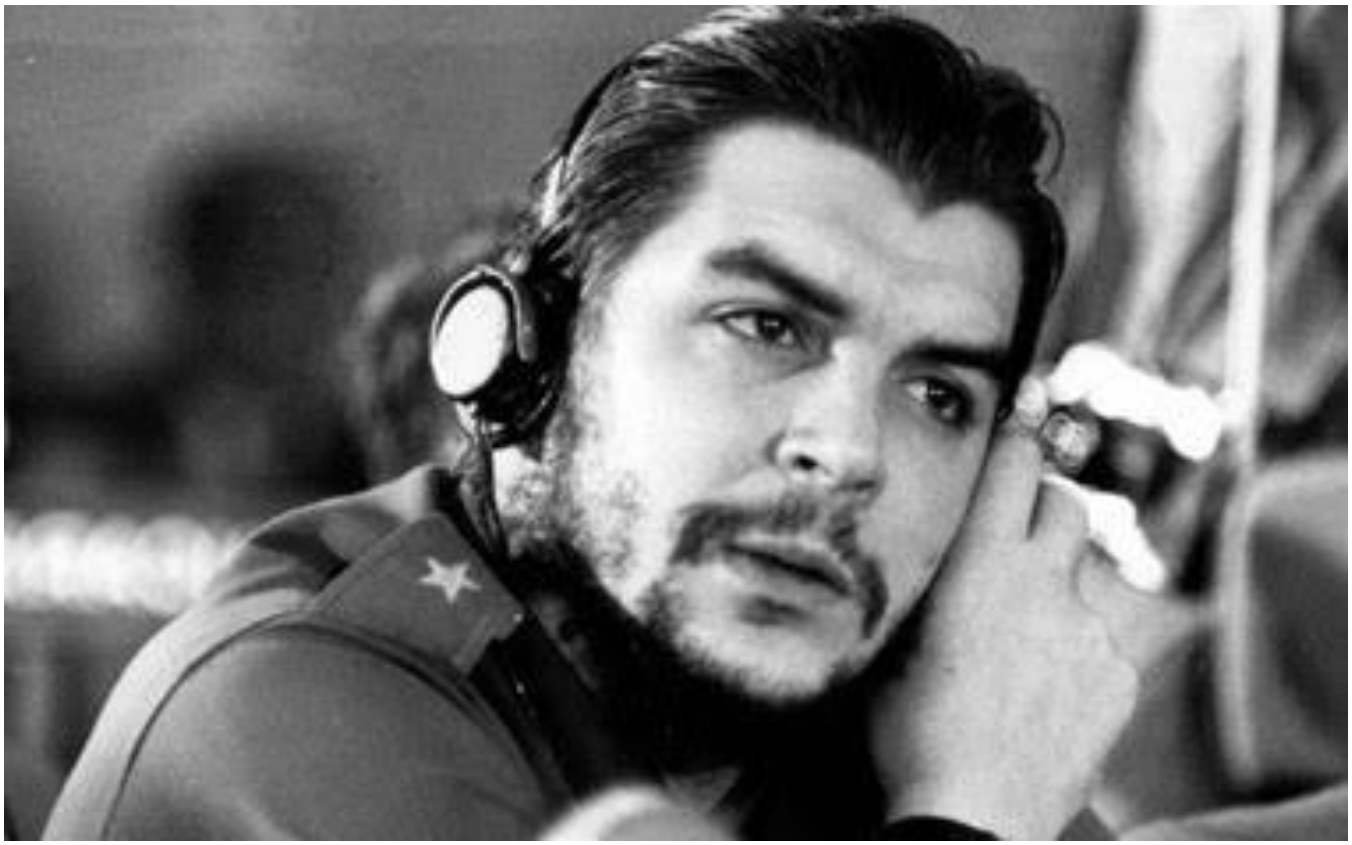
Ing. Vladimir Milián Núñez

La Habana, junio 2018

"Año 60 de la Revolución"

“El futuro de toda la industria, y el futuro de la humanidad, no está en la gente que llena papeles, está en la gente que construye máquinas, que entre otras cosas pueden llenar papeles y perforar tarjetas. Está en la gente que estudia los grandes problemas tecnológicos, los resuelve, los de hoy y los de mañana, descubre nuevas cosas, aprende a sacarle a la naturaleza nuevas cosas”

Ernesto Ché Guevara



DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos 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 _____ del año _____.

_____

Firma del Autor

Armando Milla Izquierdo

_____

Firma del Tutor

Ing. Vladimir Milián Núñez

Dedicatoria

Esta tesis es dedicada especialmente a mi mamá que es quien ha guiado cada paso de mi vida, apoyándome en cada decisión que he tomado entendiendo cada error que he cometido dándome la fuerza que necesito para sobrepasar cada obstáculo que se me ha presentado. En fin, por ser como eres una persona especial no conmigo si no con todos a tu alrededor esa manera tuya de comprender a todos. Esa forma que he tratado de igualar porque si tú has sido mi modelo a seguir. Es por esto que te la dedico porque más que por mí esta tesis fue realizada por ti.

"Mamá"

Agradecimientos

Agradezco a todas las personas que me han ayudado a la realización de esta tesis, no solo de la tesis sino de todos estos años que he pasado en la universidad. A mi familia por estar ahí siempre en todo momento a mis amigos tanto de la escuela como de la casa. A todos los profesores que me enseñaron tanto las materias como los principios de un estudiante universitario por su paciencia conmigo que tiene que ser grande. A la universidad por brindarme todos los recursos para poder terminar esta carrera y convertirme en ingeniero informático.

Resumen

El volumen de información digital que se almacena y crece considerablemente en la era actual, dentro la cual existen terabytes de datos inútiles, por lo que se ha llegado a la necesidad de no solo poder almacenar y obtener los datos con eficiencia, sino también obtener conocimiento útil. En los últimos años el aprendizaje multi-etiqueta se ha utilizado con el fin de poder clasificar esta información y facilitar su uso. Por tal razón, ha surgido la necesidad de utilizar algoritmos de clasificación multi-etiqueta con enfoque de adaptación basado en instancias para realizar tareas de aprendizaje automático y análisis de datos.

El presente trabajo se fundamenta en la implementación del algoritmo IBLR-ML de métodos de adaptación basado en instancias en la herramienta computacional de Python scikit-multilearn para ser incluirlo en la biblioteca especializada que posee esta plataforma en la clasificación multi-etiqueta como los algoritmos de MLknn y BRknn. Para la solución planteada se utilizan como entorno de desarrollo PyCharm en su versión 2017.2.2 que trabaja con el lenguaje de programación Python. Por último, el algoritmo desarrollado fue entrenado mostrando una salida satisfactoria dejando claro su efectividad aplicándoles las pruebas; validación cruzada y la de Friedman.

Palabras claves: algoritmo IBLR-ML, adaptación, clasificación, instancias y multi-etiqueta.

Abstract

The volume of digital information that is stored and is growing considerably in the current era, in which there are terabytes of useless data, which has led to the need not only to store and obtain data efficiently, but also to obtain useful knowledge. In recent years, multi-label learning has been used in order to classify this information and facilitate its use. For this reason, the need has arisen to use multi-label classification algorithms with an instance-based adaptive approach to perform automatic learning and data analysis tasks.

The present work is based on the implementation of the IBLR-ML algorithm of instance-based adaptation methods in the Python scikit-multilearn computational tool for inclusion in the specialized library that this platform has in the multi-label classification as MLknn and BRknn algorithms. For the proposed solution, the PyCharm development environment is used in its 2017.2.2.2 version, which works with the Python programming language. Finally, the algorithm developed was trained showing a satisfactory outcome by making clear its effectiveness by applying the tests; cross validation and Friedman's validation.

Keywords: IBLR-ML algorithm, adaptation, classification, instances and multi-labeling.

Índice

Índice de Tabla

Índice de figuras

Índice de ecuaciones

Introducción

La cantidad de información digital que se crea y almacena se duplica cada año como señala en su último estudio "El Universo Digital" realizado por la consultora IDC y patrocinado por EMC, los cuales están entre los primeros fabricantes de dispositivos de almacenamiento del mundo (Seijas Candelas 2016). Dentro de los volúmenes de información almacenadas existen terabytes de datos inútiles, por lo que se ha llegado a la necesidad de no solo poder almacenar y obtener los datos con eficiencia, sino que también se pueda obtener conocimiento útil y estadísticas de los datos en bruto almacenados. Con el surgimiento de la Minería de Datos (Data Mining, DM) se trata de resolver estos problemas mediante el análisis de toda esta información que ya está almacenada en bases de datos y se puede definir como un proceso de búsqueda y descubrimiento de patrones en datos (Rivero y Perla 2015). Aunque la disciplina de la minería de datos abarca un amplísimo rango de tareas, la mayor parte de las técnicas de la minería de datos caen dentro de lo que se denomina Aprendizaje Automático (*Machine Learning*) (Hall et al. 2009).

Dentro del aprendizaje automático se debe destacar el Aprendizaje Inductivo, en el que se basan la mayoría de las tareas de la minería de datos (Ramírez Quintana y Ramírez 2004). El aprendizaje inductivo se usa para adquirir conocimiento (formulado en forma de descripciones intencionales) a partir de ejemplos. El objetivo de la inducción es formular afirmaciones que explican los hechos dados y se pueden aplicar a hechos no vistos con anterioridad. Estas afirmaciones pueden expresarse como patrones, y estos patrones se representan por vectores, ecuaciones, árboles, reglas o enunciados lógicos. Muchos problemas de inducción se pueden describir como sigue: se parte de un conjunto de entrenamiento de ejemplos preclasificados, donde cada ejemplo (también llamado observación o caso) se describe por un vector de valores para rasgos o atributos, y el objetivo es formar una descripción que pueda ser usada para clasificar ejemplos previamente no vistos con alta precisión (Witten et al. 2011).

Dentro del aprendizaje inductivo se suelen distinguir tres paradigmas clásicos de aprendizaje: el aprendizaje supervisado, el no supervisado y el aprendizaje con refuerzo (Nilsson 1996). Un método clásico de aprendizaje supervisado es el algoritmo de los k vecinos más cercanos (k Nearest Neighbour, kNN) (Cover y Hart 2006). El KNN es un algoritmo que a partir de un juego de datos inicial su objetivo será clasificar correctamente todas las instancias nuevas. El juego de datos típico de este tipo de algoritmos está formado por varios atributos descriptivos y un solo atributo objetivo (también llamado clase). El algoritmo clasifica cada dato nuevo en el grupo que corresponda, según tenga k vecinos más cerca de un grupo o de otro. Es decir, calcula la distancia del elemento nuevo a cada uno de los existentes, y ordena dichas distancias de

menor a mayor para ir seleccionando el grupo al que pertenecen. Este grupo será, por tanto, el de mayor frecuencia con menores distancias (Ruiz 2017).
El algoritmo KNN se basa en la clasificación simple-etiqueta, pero con la profundización de los estudios de los datos se entiende que una instancia no tiene que necesariamente pertenecer a una sola clase. La misma puede presentar varias clases o etiquetas lo que hace inutilizables los métodos tradicionales para este tipo de clasificación, lo que conllevó a desarrollar métodos de aprendizaje multi-etiquetas (MLL Methods). En (Morales 2017) estos se dividen en dos grupos por la forma de operar: los problemas de transformación y los métodos de algoritmos de adaptación.

La diferencia de estos métodos es que el de transformación convierte los datos de multi-etiquetas en datos de etiquetas simple y luego con un clasificador tradicional busca una a una cada clase que pertenece a la instancia por clasificar. Los algoritmos de adaptación trabajan con algoritmos tradicionales y los acoplan con otra serie de operaciones para poder trabajar y clasificar las instancias de forma simultánea (asignando todas las etiquetas que le correspondan a la vez).

El fundamento teórico de varias investigaciones, en centros de desarrollo de software de la línea científica de Inteligencia Artificial y Reconocimiento de Patrones, se basan en la exploración del conocimiento en diversos almacenes de datos de los que se identifican patrones e información útil. Para lograr desarrollar estos procesos es necesario usar algoritmos de minería de datos, como son los de clasificación multi-etiqueta y uno de los que se quiere aplicar es IBLR-ML el cual está basado en el kNN. Actualmente existen varias herramientas que implementan métodos de MLL desarrolladas sobre diversas tecnologías como Java[1], Matlab[2] o Python[3]. En Java se encuentra Mulan[4] ; en Matlab los paquetes desarrollados por el grupo LAMDA o en el sitio del profesor Min-Ling Zhang[5] 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 IBLR-ML (Szymański y Kajdanowicz 2017). 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

[1] *Java: Lenguaje de programación de propósito general, concurrente, orientado a objetos.*
[2] *MATLAB: Lenguaje del cálculo técnico.*
[3] *Python: Lenguaje de programación interpretado.*
[4] *Mulan: Biblioteca de código abierto de Java para aprender de conjuntos de datos multitiquetas.*
[5] *Min-Ling Zhang: Profesor, Escuela de Ciencias de la Computación e Ingeniería, Southeast University, China.*

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).

Para lograr diseñar experimentos y comparar el desempeño del algoritmo IBLR-ML utilizando las mismas métricas de evaluación y el mismo conjunto de datos con otros algoritmos en condiciones similares, se requiere la implementación de este algoritmo en la herramienta computacional scikit-multilearn.

A partir de lo expuesto anteriormente se plantea como **problema de investigación:** la inexistencia del algoritmo IBLR-ML en la herramienta scikit-multilearn.

Enmarcándose en el **objeto de estudio**: en los algoritmos de adaptación de aprendizaje multi-etiqueta basados en instancias.

Para la solución de este problema se tiene como **objetivo general:** desarrollar en la herramienta computacional scikit-multilearn el algoritmo IBLR-ML para problemas de aprendizaje multi-etiqueta basado en el enfoque de adaptación.

Encapsulando el **campo de acción**: en el algoritmo de adaptación basado en instancia IBLR-ML y su implementación en la herramienta computacional scikit-multilearn.

Objetivos específicos:

- Caracterizar el marco teórico-conceptual de los algoritmos de aprendizaje multi-etiquetas basados en instancias con un enfoque de adaptación, haciendo énfasis en las herramientas de aprendizaje automático disponible para ello.
- Realizar el diseño teórico del proyecto para poder implementar de forma correcta el algoritmo IBLR-ML.
- 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 computacionales.

Tareas de investigación:

- Elaboración del marco teórico conceptual de los algoritmos de aprendizaje multi-etiquetas basados en instancias con un enfoque de adaptación.
- Caracterización y comparación de las herramientas de aprendizaje automático.
- Selección de herramientas, tecnologías y metodologías para el desarrollo de la solución propuesta.
- Descripción y conceptualización del algoritmo IBLR-ML para la herramienta computacional scikit-multilearn.

- Implementación del algoritmo IBLR-ML en la herramienta computacional scikit-multilearn.
- Validación del algoritmo mediante pruebas de software y estadísticas.

Métodos Teóricos:

Analítico-Sintético: se utiliza para realizar el estado del arte sobre los algoritmos de clasificación basados en adaptación y las herramientas que los implementan. También se utilizó para un análisis general de las métricas y metodologías para la evaluación de los clasificadores.

Sistémico-Estructural-Funcional: para relacionar hechos aparentemente aislados y se formula una teoría que unifica los diversos elementos.

Inductivo-deductivo: se hace uso de él para describir las bases teóricas del algoritmo IBLR-ML para un mayor entendimiento y dominio del método.

Modelación: se utiliza en el diseño de los diferentes algoritmos estudiados y la implementación del IBLR-ML; en el diseño de los diagramas de clases, 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.

Métodos Empíricos

Observación: se observan las deficiencias que presenta la aplicación del algoritmo IBLR-ML para la clasificación de las multi-etiqueta.

Medición: en la experimentación se utiliza métricas para determinar la precisión del algoritmo como Accuracy y Hamming loss, así como la prueba de Friedman para comparar los resultados de los algoritmos estudiados.

El desarrollo de la investigación ofrece la implementación del algoritmo IBLR-ML para la clasificación multi-etiqueta en la herramienta computacional scikit-multilearn a la comunidad científica en el área de la Inteligencia Artificial.

Estructura del trabajo

Capítulo 1. “Fundamentación teórica”, aborda el estudio del estado del arte del tema relacionado con el trabajo que se desarrolla, los conceptos y definiciones relacionados con las tendencias actuales.

Capítulo 2. “Propuesta de solución del algoritmo lBLR-ML”: se describen las fases del proceso KDD para desarrollar una investigación y crear un modelo conceptual óptimo para implementar el algoritmo.

Capítulo 3. “Implementación y Resultados”: en este capítulo se evalúa la calidad de la solución final mediante la realización de pruebas, con el objetivo de solucionar errores y no conformidades presentes en la implementación.

Capítulo 1. Marco teórico referencial sobre el Algoritmo de adaptación IBLR-ML para problemas de aprendizaje multi-etiquetas

En el presente capítulo se explican los teoremas fundamentales sobre el tema de la clasificación multi-etiqueta. Se realiza el estudio del estado del arte de los algoritmos que están implementados en la herramienta computacional scikit-multilearn con enfoques adaptativo basados en instancias. Además, se estudian y describen las herramientas computacionales existentes que implementan este tipo de algoritmo. Se determina la metodología que va a plantear las etapas por la que pasa el ciclo de vida del software.

1.1 Machine learning

No se puede hablar de reconocimiento de patrones, estadísticas ni de predicciones si mencionar a machine learning: que, de acuerdo a Arthur Samuel[6] en 1959 es un campo de las ciencias de la computación, le da a las computadoras la habilidad de aprender sin ser explícitamente programadas (Pasillas 2017)**.**

El Machine Learning en su uso más básico es la práctica de usar algoritmos para parsear datos, aprender de ellos y luego ser capaces de hacer una predicción o sugerencia sobre algo. Los programadores deben perfeccionar algoritmos que especifiquen un conjunto de variables para ser lo más precisos posibles en una tarea en concreto. La máquina es entrenada utilizando una gran cantidad de datos dando la oportunidad a los algoritmos a ser perfeccionados (Watkins y Dayan 1992).

En (González 2014) se define *Machine Learning* como una disciplina científica del ámbito de la Inteligencia Artificial que crea sistemas que aprenden automáticamente. En este contexto se entiende como "aprender": identificar patrones complejos en millones de datos. La máquina que realmente aprende es un algoritmo que revisa los datos y es capaz de predecir comportamientos futuros. Por otra parte, se entiende por "*automáticamente"* en este contexto, que estos sistemas se mejoran de forma autónoma con el tiempo, sin intervención humana.

El campo de aplicación práctica depende de la imaginación y de los datos que estén disponibles en la empresa (González 2014):

- Detectar fraude en transacciones.
- Predecir de fallos en equipos tecnológicos.
- Prever qué empleados serán más rentables el año que viene (el sector de los Recursos Humanos está apostando seriamente por el Machine Learning).

[6] **Arthur Samuel:** Pionero en el campo de los juegos informáticos y la inteligencia artificial y el creador de uno de los primeros juegos didácticos.

- Seleccionar clientes potenciales basándose en comportamientos en las redes sociales, interacciones en la web.
- Predecir el tráfico urbano.
- Saber cuál es el mejor momento para publicar tuits, actualizaciones de Facebook o enviar las hojas informativas.
- Hacer pre diagnósticos médicos basados en síntomas del paciente.
- Cambiar el comportamiento de una app móvil para adaptarse a las costumbres y necesidades de cada usuario.
- Detectar intrusiones en una red de comunicaciones de datos.
- Decidir cuál es la mejor hora para llamar a un cliente.

1.2 Clasificación multi-etiqueta

Con la profundización del estudio de la minería de datos se han descubierto nuevos fenómenos como es el caso del aprendizaje multi-etiqueta, una de las características de este aprendizaje es que es capaz de trabajar con instancias que poseen más de una etiqueta.

La clasificación multi-etiqueta es un campo del aprendizaje automatizado que se desarrolla rápidamente. A pesar de su corta vida, se han propuestos varios métodos para resolver tareas para este tipo de clasificación. La mayoría de los problemas de clasificación asocian a cada instancia una única etiqueta, l_i de un conjunto disjunto de etiquetas posibles, L. Si dicho conjunto de etiquetas tiene dos posibles valores, que representan la pertenencia o no pertenencia a una clase, hablamos de clasificación binaria (|L|=2), mientras que en el resto de los casos se habla de clasificación multi-clases (|L|>2). No obstante, este no es el único escenario posible, ya que hay numerosos problemas de creciente interés en los que una instancia puede tener asociado un conjunto de etiquetas de clase. En estos casos se habla de multi-etiqueta (Rivero y Perla 2015).

Los métodos de clasificación multi-etiqueta basados en aprendizaje supervisado existentes se agrupan en dos categorías: métodos de transformación de problemas y métodos de adaptación de algoritmos. Los métodos de transformación, aquellos métodos que transforman el problema de clasificación multi-etiqueta en uno o más problemas de clasificación única o problemas de regresión (M.Browna et al. 2004). Se clasifican como métodos de adaptación de algoritmo, aquellos métodos que contienen algoritmos de aprendizaje capaces de manejar datos de etiqueta múltiple directamente (Tsoumakas y Katakis 2007).El presente trabajo se enfoca en los métodos de adaptación ya que es a la familia que pertenece el algoritmo que se implementa en esta investigación.

1.3 Algoritmos de adaptación multi-labels

La clasificación de multi-etiqueta ha recibido una atención cada vez mayor en el aprendizaje automático en los últimos años, no solo por su relevancia práctica, sino también porque es interesante desde un punto de vista teórico. De hecho, aunque es posible reducir el problema de la clasificación multi-etiqueta a la clasificación convencional de una manera o el otro y, por lo tanto, aplicar los métodos existentes para que este último resuelva el primero. Las soluciones directas de este tipo generalmente no son óptimas. En particular, desde la presencia o ausencia de las diferentes etiquetas de clase debe predecirse simultáneamente, obviamente es importante explotar las correlaciones e interdependencias entre estas instancias. Esto generalmente no se logra mediante simples transformaciones a clasificación estándar (Cheng y Hüllermeier 2009).

Los métodos de adaptación emplean una técnica de clasificación clásica adaptada para trabajar directamente con datos multi-etiqueta. Casi todas las técnicas de clasificación se han tratado de adaptar para solventar problemas multi-etiqueta entre las que podemos mencionar las Máquinas de Vectorial Soporte (Xu 2011), Árboles de Decisión (Gold y Petrosino 2010), Redes Neuronales (Zhang 2009), Clasificación Asociativa (Rak, Kurgan y Reformat 2008), Métodos Probabilísticos (Larrañaga, Li y Bielza 2011), Algoritmos Bioinspirados (Vallim et al. 2008), y los basados en instancias.

En este enfoque, se han desarrollado extensiones de clasificadores de una sola etiqueta, adaptando sus mecanismos internos para permitir su uso en problemas de etiquetas múltiples. Además, se pueden desarrollar nuevos algoritmos específicos para problemas de multilabel (Read et al. 2009). Se espera que un algoritmo desarrollado específicamente para resolver problemas de múltiples etiquetas tenga un mejor rendimiento que los métodos basados en la transformación del problema (Cerri, Silva y Carvalho 2009).

Los algoritmos de adaptación tienen ventajas sobre los de transformación ya que son capaces de asignar etiquetas de forma simultánea, sin cambiar los datos almacenados evitando la perdida de información y poden trabajar con la correlación entre etiquetas.

1.4 Algoritmos de adaptación basados en instancias

Los métodos de adaptación basados instancias son también llamados métodos basado en casos, basado en ejemplares o perezoso (lazy) denominados así porque retrasan el procesamiento hasta que hay que clasificar una nueva instancia (calonso 2014). La clasificación de multi-etiqueta es una extensión de la clasificación clásica en el cual una sola instancia puede asociarse con múltiples etiquetas. Al igual que para la clasificación convencional, los algoritmos de aprendizaje basados en instancias que utilizan el principio de estimación del vecino más cercano se pueden usar con bastante éxito en este contexto (Cheng y Hüllermeier 2009).

Los métodos basados en instancias utilizan enfoques conceptualmente sencillos para las aproximaciones de valores reales o discretos de las funciones de salida. Aprender en estos modelos consiste en almacenar los datos de entrenamiento presentados y cuando una nueva instancia es encontrada, un grupo de ejemplos similares relacionados son recuperados de memoria y usados para clasificar la nueva instancia consultada. Una diferencia clave en estos enfoques, con respecto a otros métodos, es que pueden construir una aproximación diferente de la función de salida para cada ejemplo que debe ser clasificado (Rivero y Perla 2015). En la herramienta scikit-multilearn se encuentran desarrollados dos algoritmos de este tipo MLkNN y BRkNN.

Estos métodos dependen de los datos almacenados ya que a mayor cantidad de casos halla para estudiar y comparar más certera será la respuesta. Es que estos se basan en la información existente para clasificar la nueva instancia. Son capaces de asignar de forma simultanea las etiquetas.

1.5 Herramientas computacionales especializada en aprendizaje multi-etiqueta

En la actualidad existen varias herramientas computacionales en diversas plataformas para el trabajo con el aprendizaje multi-etiqueta dentro de las más destacadas se encuentran: la plataforma Java con Mulan, Matlab con los paquetes desarrollados por el grupo LAMDA (Núñez et al. 2006) o el paquete Matlabarsenal (M. Martínez 2010) y en Python la herramienta scikit-multilearn. En el presente epígrafe se realizara la comparación con Mulan ya que es la herramienta más usada (Szymański y Kajdanowicz 2017), contiene el algoritmo que se desea implementar, además Matlab es un entorno académico bajo diferentes tipos de licencias privativas.

1.5.1 Herramienta computacional Mulan

La herramienta es un software de código abierto para máquinas de aprendizaje cuyo objetivo principal es el trabajo con datos multi-etiquetas. Ofrece los algoritmos que existen sobre clasificación multi-etiqueta y ranking de etiquetas, así como un marco de trabajo que contiene un compendio de medidas para evaluar la clasificación multi-etiqueta mediante la validación *hold-out* y la validación cruzada. Mulan está escrita en Java y se construye encima de Weka para emplear todos los recursos que contiene sobre algoritmos de aprendizaje supervisado, aunque esta es independiente. Una característica exclusiva de la librería Mulan es la introducción reciente de un paquete de experimentos que tiene como objetivo reproducir los resultados de los experimentos en el ámbito de la predicción con salidas compuestas (Tsoumakas et al. 2011).

El objetivo principal de MULAN es llevar los beneficios del software de código abierto para el aprendizaje automático (MLOSS) (Sonnenburg et al. 2007) a las personas que trabajan con datos de múltiples etiquetas. La disponibilidad de MLOSS es especialmente importante en áreas emergentes como el aprendizaje de múltiples etiquetas, porque elimina la carga de implementar el trabajo relacionado y acelera el progreso científico. En el aprendizaje con múltiples etiquetas, una carga adicional es la implementación de medidas de evaluación apropiadas, ya que éstas son diferentes en comparación con las tareas tradicionales de aprendizaje supervisado. La evaluación de algoritmos multi-etiqueta con una variedad de medidas, es considerada importante por la comunidad, debido a los diferentes tipos de salida (bipartición, ranking) y diversas aplicaciones (Tsoumakas et al. 2011).

Con este objetivo, MULAN ofrece una gran cantidad de algoritmos de punta para la clasificación de etiquetas múltiples, así como un marco de evaluación que calcula una gran variedad de medidas de evaluación de etiquetas múltiples a través de la evaluación diferida y la validación cruzada. Además, la biblioteca ofrece una serie de estrategias de umbralización que producen biparticiones a partir de vectores de puntuación, métodos simples de línea de base para la reducción de la dimensionalidad de múltiples etiquetas y soporte para la clasificación jerárquica de múltiples etiquetas, incluyendo un algoritmo implementado (Tsoumakas et al. 2011).

MULAN es una biblioteca. Por lo tanto, sólo ofrece (Interfaz de programación de aplicaciones) API, no hay interfaz gráfica de usuario (GUI) disponible. La posibilidad de utilizar la biblioteca a través de la línea de comandos, es también actualmente no soportado. Otra desventaja de MULAN es que funciona con todo lo que hay en la memoria principal por lo que existen limitaciones con conjuntos de datos muy grandes. MULAN está escrito en Java y está construido sobre Weka (Witten and Frank, 2005). Esta elección se hizo con el fin de aprovechar los vastos recursos de Weka en algoritmos de aprendizaje supervisado, ya que muchos de los algoritmos de aprendizaje multi-etiqueta de última generación se basan en la transformación de problemas (Tsoumakas et al. 2011).

1.5.2 Comparación entre las Herramientas

Con estudios realizados como los que se muestran en (Szymański y Kajdanowicz 2017) se evidencia la competitividad de la herramienta scikit-multilearn sobre la que se quiere desarrollar el algoritmo IBLR-ML. La cual muestra ventaja sobre Mulan probando el mismo algoritmo en ambas herramientas y mejorando aspectos como tiempo de ejecución. scikit-multilearn al estar implementada en Python permite más flexibilidad en el código. Desde esta herramienta se pueden hacer llamadas a los métodos de scikit-learn y tiene una interfaz a Meka/Weka/Mulan lo que le permite sus métodos por si se quieren llamar a métodos que aún no se han implementado en scikit. En conclusión, se tiene una herramienta con un mejor

rendimiento en tiempo de ejecución y con la capacidad de llamar a métodos de la biblioteca de clasificación multietiqueta más destacada hasta la fecha (en términos de popularidad) que es Mulan (Szymański y Kajdanowicz 2017). Por estas razones se determinó la necesidad de implementar el algoritmo IBLR-ML en la herramienta computacional scikit-multilearn

1.6 Algoritmos de adaptación multi-etiquetas basados en instancias de scikit-multilearn

La herramienta scikit-multilearn está actualmente en desarrollo es implementada sobre la plataforma Python. Cuenta con dos métodos de adaptación de algoritmos basados en instancias o en clasificador kNN. En el presente epígrafe se explica el funcionamiento de dichos algoritmos y porque sería factible agregar el algoritmo IBLR-ML en esta herramienta.

1.6.1 Multi-Label k-Nearest Neighbor (MLkNN)

El modelo MLkNN es un algoritmo perezoso derivado del tradicional algoritmo kNN, este es uno de los más ampliamente utilizados en la clasificación clásica. Fue desarrollado por Min-Ling Zhang[7] y Zhi-Hua Zhou[8]. Este algoritmo determina si una instancia le pertenece una clase, determinada por las k instancias más cercanas del conjunto de entrenamiento y examinando la clase a la que pertenecen estas.

Dada una instancia x y sus etiquetas asociadas $Y \subseteq Y$, se obtiene que $\overrightarrow{yx}$ es el vector categoría para x donde la l-ésima componente de $\overrightarrow{yx}$ (l) (l $\in$Y) toma el valor 1 si l $\in Y$ y 0 en otro caso. Se denota N (x) el conjunto de los k vecinos de x identificados en el conjunto de entrenamiento. Así, basados en el conjunto de etiquetas de estos vecinos el vector de conteo de membresía puede definirse como:

Ecuación 1. Conteo de membresía

$$\overrightarrow{\mathbf{C_x}}(\mathbf{l}) = \sum \overrightarrow{\mathbf{y_a}}_{\mathbf{a}\in\mathbf{N(x)}}(\mathbf{l}), \mathbf{l} \in \mathbf{r}$$

Donde $\overrightarrow{Cx}$ cuenta el número de vecinos de x que pertenecen a la l-ésima clase. Para cada instancia t, MLkNN primero identifica sus k vecinos más cercanos N (t) en el conjunto de entrenamiento. Así H_1^l es el evento que t tiene la etiqueta l, mientras H_0^l es el evento que t que no tiene la etiqueta l. Además, el E_j^l (j$\in$Y) denota el evento que entre los k vecinos hay exactamente j instancias con la etiqueta l. Sin embargo, basado en el vector de conteo de membresía $\overrightarrow{Cx}$ (l) el Vector categoría $\overrightarrow{yt}$ es determinado usando el siguiente principio “maximum a posteriori”:

[7] *Min-Ling Zhang: Profesor, Escuela de Ciencias de la Computación e Ingeniería, Southeast University, China.*
[8] *Zhi-Hua Zhou: Profesor, Departamento de Ciencias de la Computación y Tecnología, Universidad de Nanjing, China.*

Ecuación 2. Maximum a posteriori

$$\overrightarrow{y_t}(l) = argmax_{be\{0,1\}} P\left(H_b^l \middle| PE^l_{\overrightarrow{C_t}(l)}\right), l\varepsilon r$$

Usando las reglas Bayesianas, la ecuación anterior puede rescribirse como:

Ecuación 3. Regla de bayes para $\overrightarrow{y_t}(l)$

$$\overrightarrow{y_t}(l) = argmax_{be\{0,1\}} \frac{P(H_b^l)P(E^l_{\overrightarrow{C_t}(l)}|H_b^l)}{P(E^l_{\overrightarrow{C_t}(l)})}$$

En general, para cada instancia no vista, se determina del conjunto de instancias de entrenamientos sus k vecinos más cercanos y después basado en la información estadística obtenida del conjunto de etiquetas de dichos vecinos, es decir, el número de instancia que pertenecen a cada clase, se utiliza el principio de "*maximum a posteriori*" para determinar el conjunto de etiquetas para la instancia no vista (LingZhangZhi-HuaZhou y HuaZhou 2007).

1.6.2 Binary Relevance in conjunction with the kNN algorithm (BRkNN)

El modelo BRkNN es una adaptación del algoritmo kNN para clasificación multi-etiqueta que es conceptualmente equivalente a usar el popular método de transformación BR en conjunción con el algoritmo kNN. Se identifican dos extensiones de este modelo que mejoran su rendimiento de predicción general. Esta extensión para la clasificación multi-etiqueta fue creada por Eleftherios Spyromitros[9], Grigorios Tsoumakas, Ioannis Vlahavas[10].

Uno de los métodos de transformación de problemas para datos multi-etiqueta más conocidos es el BR. Este utiliza un clasificador binario $h_\lambda: X \to \{\neg\lambda, \lambda\}$ para cada etiqueta diferente λ_ε L. El método BR transforma el conjunto de datos originales en |L| conjunto de datos D_λ que contienen todos ejemplos del conjunto de datos original, etiquetando como λ si las etiquetas del ejemplo original contienen a λ_y como $\neg\lambda$ en otro caso. Es la misma solución utilizada con el fin de hacer frente a un problema multi-clase utilizando un clasificador binario, conocido comúnmente como uno-contra-todos o uno-contra-resto.

En lugar de implementar BRkNN se podría haber utilizado implementaciones existentes de BR y kNN. Sin embargo, el problema de emparejar BR con kNN es que este va a realizar |L| veces el proceso de calcular los k vecinos más cercanos. Para evitar estos cálculos que consumen mucho tiempo redundante, BRkNN

[9] *Eleftherios Spyromitros: Doctor en Machine Learning de la Facultad de Informática de la Universidad Aristóteles de Thessaloniki.*
[10] *Ioannis Vlahavas: Profesor, Asistente en la Facultad de Informática de la Universidad Aristóteles de Tesalónica.*

extiende el algoritmo kNN para que las predicciones independientes se hagan para cada etiqueta, después de una sola búsqueda de los k vecinos más cercanos. De esta manera BRkNN es |L| veces más rápido que BR más kNN durante prueba, un hecho que podría ser crucial en dominios con grandes conjuntos de etiquetas y los requisitos para los tiempos de respuesta bajos.

Este algoritmo tiene dos extensiones, las cuales son basadas en el cálculo de la confidencia para cada etiqueta que puede fácilmente obtenerse considerando el porcentaje de los k vecinos más cercanos que la incluyen. Formalmente, se sabe que Y_j ,$j = 1 \ldots k$ es el conjunto de etiquetas de los k vecinos más cercanos de una nueva instancia x. La confidencia c_λ de una etiqueta λ_ε es igual a:

Ecuación 4. Confidencia

$$c_{\lambda=\frac{1}{k}-\sum_{j=1}^{k} I_{Y_j}(\lambda)}$$

Donde l_{Y_j}: L→ {0,1} es una función que da como salida 1 si la etiqueta de salida λ pertenece al conjunto Y_j y 0 en otro caso, llamada función indicadora en la teoría de conjuntos (Tsoumakas y Katakis 2008).

Este método devuelve una matriz con las instancias como filas y etiquetas como columnas y en cada espacio se guardan las confidencias correspondientes. El método BRkNN contiene dos extensiones para dar la solución a de sus clasificaciones el método BRkNNa y BRkNNb. El algoritmo BRkNNa hereda los métodos de este último, toma la matriz de confidencia creada por BRkNN, redondea cada posición convirtiendo en uno los valores mayores igual que 0.5 y cero los menores, devuelve una matriz de indicadores binarios con asignación de etiquetas en la fila se muestran las instancias y en las columnas las etiquetas donde exista un uno es que esa instancia posee esa etiqueta y cero en caso contrario. El método BRkNNb igual devuelve una matriz binaria de correspondencia de las etiquetas solo que antes de redondear este ordena la matriz de probabilidades (confidencia) de menor a mayor y luego redondea pudiendo detectar en caso de pertenecer cual es la etiqueta de mayor etiqueta.

1.6.3 Valoración de los algoritmos

Con el estudio de estos algoritmos se entendió su funcionamiento a la hora de clasificar, cuales son los métodos que adaptan y la ventaja de usar los métodos combinado y no por separado. Una de las desventajas que se descubrió con este análisis es que ninguno de estos algoritmos tiene presente la dependencia entre las etiquetas. Estudios previos demuestran que, para obtener modelos más precisos, durante el proceso inductivo se deben tener en cuenta las posibles dependencias que existen entre las etiquetas (Gil Martínez 2011).

Por esta razón se decide implementar en la herramienta scikit-multilearn el algoritmo IBLR-ML. La idea de IBLR es obtener un balance óptimo entre inferencia global y local, se propone una versión extendida IBLR+

también entre aprendizaje basado en instancia y basado en modelo que puede lograrse mediante la estimación de los coeficientes de regresión óptimos. Estos coeficientes reflejan directamente con su signo y magnitud las dependencias entre las etiquetas. Esta capacidad distingue a IBLR de métodos basados en instancias hasta ahora existentes para clasificación multi-etiqueta y es probable que sea uno de los principales factores para su excelente rendimiento (Rivero y Perla 2015).

1.7 Metodología de evaluación de algoritmos de aprendizaje automático Validación cruzada

La validación cruzada es un método de evaluación de modelos que se emplea para mejorar el error predictivo. El problema de la evaluación del error es que no se conoce que tan bien el modelo ha aprendido para predecir datos desconocidos. Una forma de solucionar este problema es no emplear todo el conjunto de datos para entrenar el modelo. Una parte del conjunto de datos es removido del conjunto de entrenamiento y empleado para la prueba, una vez que el modelo haya sido entrenado (Sierra Martínez y Pérez González 2017).

El método hold-out es el más simple de los tipos de validación cruzada. Este consiste en dividir en dos conjuntos complementarios los datos de muestra, realizar el análisis de un subconjunto (denominado datos de entrenamiento o training set), y validar el análisis en el otro subconjunto (denominado datos de prueba o test set), de forma que la función de aproximación sólo se ajusta con el conjunto de datos de entrenamiento y a partir de aquí calcula los valores de salida para el conjunto de datos de prueba (valores que no ha analizado antes). La ventaja de este método es que es muy rápido a la hora de computar. Sin embargo, este método no es demasiado preciso debido a la variación del resultado obtenido para diferentes datos de entrenamiento. La evaluación puede depender en gran medida de cómo es la división entre datos de entrenamiento y de prueba, y por lo tanto puede ser significativamente diferente en función de cómo se realice esta división. Debido a estas carencias aparece el concepto de validación cruzada (Schneider y Moore 1997).

En la validación cruzada de K iteraciones o K-fold cross-validation los datos de muestra se dividen en K subconjuntos. Uno de los subconjuntos se utiliza como datos de prueba y el resto (K-1) como datos de entrenamiento. El proceso de validación cruzada es repetido durante k iteraciones, con cada uno de los posibles subconjuntos de datos de prueba. Finalmente se realiza la media aritmética de los resultados de cada iteración para obtener un único resultado. Este método es muy preciso puesto que evaluamos a partir de K combinaciones de datos de entrenamiento y de prueba, pero aun así tiene una desventaja, y es que,

a diferencia del método de retención, es lento desde el punto de vista computacional. En la práctica, la elección del número de iteraciones depende de la medida del conjunto de datos. Lo más común es utilizar la validación cruzada de 10 iteraciones (10-fold cross-validation) (joanneum 2005).

Este método consiste al dividir aleatoriamente el conjunto de datos de entrenamiento y el conjunto de datos de prueba. Para cada división la función de aproximación se ajusta a partir de los datos de entrenamiento y calcula los valores de salida para el conjunto de datos de prueba. El resultado final se corresponde a la media aritmética de los valores obtenidos para las diferentes divisiones. La ventaja de este método es que la división de datos entrenamiento-prueba no depende del número de iteraciones. Pero, en cambio, con este método hay algunas muestras que quedan sin evaluar y otras que se evalúan más de una vez, es decir, los subconjuntos de prueba y entrenamiento se pueden solapar (Moore 2001).

La validación cruzada dejando uno fuera o Leave-one-out cross-validation (LOOCV) implica separar los datos de forma que para cada iteración se tiene una sola muestra para los datos de prueba y todo el resto conformando los datos de entrenamiento. La evaluación viene dada por el error, y en este tipo de validación cruzada el error es muy bajo, pero en cambio, a nivel computacional es muy costoso, puesto que se tienen que realizar un elevado número de iteraciones, tantas como N muestras tengan y para cada una analizar los datos tanto de entrenamiento como de prueba (Schneider, 1997).

De los métodos de validación cruzada antes expuesto se emplea para evaluar los modelos predictivos la validación cruzada de k iteraciones con k = 10, teniendo en cuenta que es el más empleado en la evaluación de los algoritmos del estado del arte. Además, este método es más preciso que el hold-out, puesto que evalúa a partir de k combinaciones de datos de entrenamiento y de prueba, todo el conjunto de datos y computacionalmente es menor el costo que el Leave-one-out.

1.7.1 Métricas para comprobar el rendimiento

También se usarán algunas métricas para medir el rendimiento del algoritmo se sabe que existe una gran variedad de estas, pero serán las basadas en ejemplo ya que estos métodos detectan la dependencia entre etiquetas. La mayoría de ellas tienen su origen en el campo de la recuperación de información.

Acierto (accuracy):

Calcula el porcentaje de etiquetas relevantes predichas en el subconjunto formado por la unión de las etiquetas asignadas por el clasificador y las relevantes.

Ecuación 5. Accuracy

$$Acierto\big(y,h(x)\big) = \frac{\sum_{i=1}^{m}[y_i = 1 \ \cup \ h_i(x) = 1]}{\sum_{i=1}^{m}[h_i(x) = 1]}$$

Hamming loss:

Definida como la proporción de etiquetas cuya relevancia es incorrectamente predicha.

Ecuación 6. Haming loss

$$\mathrm{Hamming}\big(\mathrm{y},\mathrm{h}(\mathrm{x})\big) = \frac{1}{m}\sum_{i=1}^{m}[y_i \neq h_i(x)]$$

1.8 Herramienta, metodología y tecnología utilizada

En el presente epígrafe se menciona las herramientas y tecnologías utilizada para el desarrollo de este proyecto. El igual se explica la metodología de desarrollo de proceso por la que se implementa el algoritmo planteado en la investigación.

1.8.1 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 y utiliza matrices dispersas para todas las operaciones internas. Dicha herramienta ofrece solo API (Interfaz de programación de aplicaciones). No hay interfaz gráfica de usuario (GUI) disponibles. Proporciona implementaciones nativas de Python de métodos populares de clasificación multi-etiquetas 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-etiqueta como sea posible tanto para la comunidad de fuente abierta como para los usuarios comerciales de la pila de datos de Python de scikit / scipy (Szymański y Kajdanowicz 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.

Los pasos para usar scikit-multilearn son sencillos: cargar los datos, seleccionar el método y realizar la clasificación. La librería soporta la carga de matrices de todos los formatos bien establecidos gracias a scipy y opera internamente sobre representaciones scipy[11]/numpy[12], convirtiéndose en matrices densas o listas de listas sólo si es requerido por una base scikit classifier, si se emplea en el escenario de transformación de problemas. scikit-multilearn proporciona tres sub-paquetes siguiendo la categorización establecida de métodos con múltiples etiquetas: enfoque de adaptación de métodos (en skmultilearn.adapt), enfoque de transformación de problemas (en skmultilearn.pt) y conjuntos de ambos (en skmultilearn.ensemble). Para utilizar la versión multi-etiqueta adaptada de un método de una sola etiqueta sólo hay que importarlo e instanciar un objeto de la clase. Un enfoque de transformación de problemas toma una base compatible con scikit-learn classifier e información opcional sobre si la base classifier soporta entradas escasas (Szymański y Kajdanowicz 2017).

1.8.2 Metodología de desarrollo

Una metodología es un conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar un software. Está formada por fases (sub-fases) que guiarán el ciclo de vida del proyecto. Debido a la responsabilidad que tiene sobre la vida del software, es necesario que sea elegido el enfoque adecuado para el desarrollo eficiente de este (Pressman 2007).

La selección de la metodología está estrechamente relacionada con las características específicas del tipo de proyecto que se desea realizar, de ahí la importancia de una selección adecuada. En este trabajo se está en presencia de una herramienta de minería de datos lo que conllevó a seleccionar como metodología el proceso Knowledge Discovery in Databases (KDD).

Con el proceso KDD se identifican patrones válidos, novedosos, potencialmente útiles y principalmente entendibles, implica el análisis de grandes cantidades de datos observacionales, para encontrar relaciones insospechadas (Data 2004). Muchas personas que empiezan a explorar el área de reconocimientos de patrones y estadísticas en volúmenes de información confunden que este concepto es Minería de Datos, sin embargo, no es así. La Minería de Datos en realidad es el núcleo de todo un proceso llamado Descubrimiento de Conocimiento en Base de Datos (Knowledge Discovery in Databases – KDD), el cual es un proceso metodológico para encontrar un “modelo” válido, útil y entendible que describa patrones de acuerdo a la información, y como modelo se entiende que es la representación que intenta explicar ese patrón en los datos (Landa 2016). El proceso de KDD persigue la extracción automatiza da de conocimiento no trivial, implícito, previamente desconocido y potencialmente útil a partir de grandes volúmenes de datos

[11] *scipy : es un entorno de desarrollo de software de código abierto basado en Python para matemáticas, ciencias e ingeniería.*
[12] *NumPy: es el paquete fundamental para la computación científica con Python.*

(Fayyad, Shapiro y Smyth 1996). El proceso de KDD está constituido por cinco etapas tal y como se ilustra en la Figura 1.

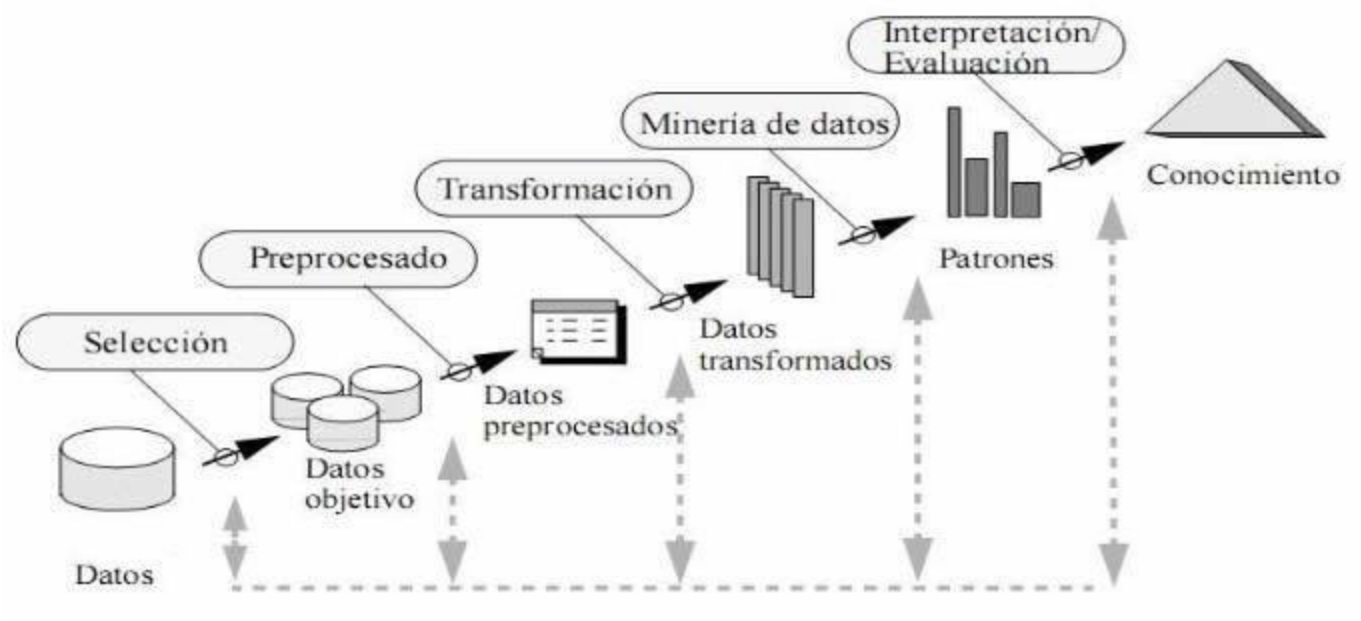


Figura 1. Etapas de KDD, extraída de (Batista y Ramírez Pérez 2015)

Como paso previo al proceso de KDD es necesario entender el dominio de aplicación y los objetivos del usuario final. Todo ello requiere cierta interacción entre el usuario y el ingeniero de Data Mining para que éste último conozca las partes que son susceptibles de un procesado automático y las que no, los objetivos, los criterios de rendimiento exigibles, para qué se usarán los resultados que se obtengan, etc.

A continuación, se explican las etapas que constituyen el proceso de KDD que se plantea en la investigación (Lara Torralbo 2010):

I. **Selección**: en esta etapa se elige el conjunto de datos objetivo sobre los que se realizará el análisis. Una consideración importante en esta etapa es que los datos pueden proceder de diferentes fuentes y, por tanto, se necesita unificarlas.

II. **Preproceso**: el objetivo de esta etapa es asegurar la calidad de los datos a analizar ya que de ello depende, en gran medida, la calidad del conocimiento descubierto. En esta fase se incluyen tareas como el filtrado de individuos atípicos (datos que no se ajustan al comportamiento general de los datos), la eliminación de ruido, el manejo de valores ausentes o la normalización de los datos.

III. **Transformación** y reducción de los datos: esta etapa consiste en modificar la estructura de los datos con el objetivo de facilitar el análisis de los mismos. Eso incluye, entre otras, consideraciones como la transformación del esquema original de los datos a otros esquemas

(tabla única, esquema desnormalizado, etc.), la proyección de los datos sobre espacios de búsqueda de menor dimensionalidad en los que es más sencillo trabajar o la discretización de datos continuos. Este paso resulta fundamental dentro del proceso global, ya que requiere un buen conocimiento del problema y una buena intuición que, con frecuencia, marcan la diferencia entre el éxito o fracaso en el descubrimiento de conocimiento

IV. **Minería de Datos (Data Mining)**: esta es la etapa en la que se analizan los datos mediante un conjunto de técnicas y herramientas, y se extrae información oculta en ellos. La etapa de Data Mining se divide, a su vez, en otros tres pasos: determinación del tipo de problema que se necesita resolver. Elección del algoritmo de minería de datos más adecuado para el problema en cuestión. Estado del arte, búsqueda de conocimiento con una determinada representación del mismo. El éxito de esta etapa depende, en gran medida, de la correcta realización de los pasos previos.

V. **Interpretación/evaluación del conocimiento extraído**: en esta última etapa se evalúa y se interpreta el conocimiento extraído en la fase anterior, atendiendo a tres criterios fundamentales: Precisión, claridad e interés. Tras la etapa de interpretación y evaluación del conocimiento extraído es posible que se necesite retroceder a etapas anteriores y repetir el proceso o parte de él. Además, la interacción entre los ingenieros de Data Mining y los expertos en el dominio de aplicación suele ser necesaria. Una vez se descubre y consolida el conocimiento, éste es incorporado al sistema en cuestión o, simplemente, es documentado y enviado a la parte interesada. Además, los modelos generados han de ser reevaluados, reentrenados y reconstruidos cada cierto tiempo para actualizarse según las nuevas tendencias que puedan aparecer en los datos.

1.8.3 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 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.

1.8.4 Entorno de desarrollo Integrado (IDE)

Un Entorno de Desarrollo Integrado (IDE) es una aplicación de software que proporciona servicios integrales para el desarrollo de software. Un IDE normalmente consiste en un editor de código fuente que permite construir herramientas de automatización. En este trabajo se trabajará con PyCharm en su versión 2017.2.2 es un IDE *(Integrated Development Environment)* utilizado para la programación en Python. Proporciona análisis de código, depuración gráfica, integración con VCS/DVCS y muchas otras características que facilitan el desarrollo de código. Se escogió debido al número de prestaciones que brinda y su especialización para ambientes de escritorio. Es un producto de código abierto, con todos los beneficios del software disponible en forma gratuita, el cual ha sido examinado por una comunidad de desarrolladores. Proporciona varias ventajas como es el caso de la facilidad de uso durante todo el ciclo de desarrollo y el soporte para lenguajes de modelado, lo que aumenta la productividad, y las razones que motivaron su uso.

1.9 Conclusiones del Capítulo

A partir de la literatura científica consultada se:

a. Profundizó en el estudio de los conceptos que están familiarizados con el entorno del algoritmo IBLR-ML.
b. Concluye la factibilidad de la implementación del algoritmo IBLR-ML en la herramienta scikit-multilearn como consecuencia de la comparación con la herramienta Mulan.
c. Estudiaron los algoritmos implementados en la herramienta scikit-multilearn y se identificó la necesidad de implementar IBLR-ML.
d. Definió el proceso (KDD) por el cual se va a regir el ciclo de vida del desarrollo del algoritmo IBLR-ML.
e. Identificaron las herramientas y tecnologías que se van a utilizar para la implementación.
f. Determinaron las métricas y las medidas de evaluación que se le realizarán al algoritmo IBLR-ML después de su implementación.

Capítulo 2: Propuesta de solución para el desarrollo del algoritmo IBLR-ML

En el presente capítulo se describe el algoritmo IBLR-ML fundamento teórico de la investigación. Primeramente, se explican los algoritmos base simple-etiqueta (kNN y IBLR) posteriormente se describe la extensión que se ha realizado para poder clasificar a más de una clase toda esta en descripción es obtenida de (Cheng y Hüllermeier 2009), la cual se implementa, para dar solución al problema de investigación planteado, en la herramienta computacional scikit-multilearn.

2.1 k-Nearest Neighbour (kNN)

El algoritmo del vecino más cercano (kNN) hace las predicciones de clasificación etiquetando a las instancias con la etiqueta de la instancia más cercana en los datos de entrenamiento (bajo diversas normas de distancias). El algoritmo puede hacerse más robusto tomando en cuenta no solo la instancia más cercana, sino las k instancias más cercanas. Las predicciones se hacen mediante el voto mayoritario de las etiquetas de clase de los k vecinos. El método de los k vecinos más cercanos es conocido por sus siglas en inglés kNN (k-nearest neighbour) (Sánchez Tarragó 2014).

2.1.1 Descripción del algoritmo

Una instancia x se describe en términos de características $\emptyset_i$, i=1,2…n, donde $\emptyset_i$(x) denota el valor de la característica i-ésima para la instancia x. La instancia en el espacio X está dotado de una medida de distancia: Δ(x, x') es la distancia entre x y x'. Primero se centra en el caso de la clasificación binaria y, por lo tanto, defina el conjunto de etiquetas de clase por Y = {-1, +1}. Una tupla (x, y) ∈ X × Y es nombrada una instancia etiquetada. D denota una muestra que consiste en N instancias etiquetadas (x_i, y_i), 1 ≤ i ≤ N. Finalmente, una nueva instancia $x_0 \in$ X (una consulta) cuya etiqueta $y_0 \in$ {-1, +1} debe ser estimada.

El principio del vecino más cercano (NN) prescribe estimar la etiqueta de la consulta x_0 por la etiqueta de la instancia más cercana (menos distante). El enfoque KNN es una ligera generalización, que tiene en cuenta el k ≥ 1 vecinos más cercanos de x_0. Es decir, una estimación $\widehat{y_0}$ de y_0 se deriva del conjunto $N_k(x_0)$ de los k vecinos más cercanos de x_0, generalmente por medio de un voto mayoritario:

Ecuación 7. Función del kNN

$$\widehat{y_0} = arg\ max_{y \in Y} \#\{x_i \in N_k | y_i = y\}$$

2.2 Instance Based Learning by Logistic Re-gression (IBLR)

El algoritmo IBLR junto con el kNN son la base del IBLR-ML. Una idea clave de este enfoque es considerar las etiquetas de las instancias vecinas como "características" de la consulta x_0 cuya etiqueta será estimada. Vale la pena mencionar que ideas similares se han explotado recientemente en el aprendizaje relacional y la clasificación colectiva.

Denotada por p_0 la probabilidad previa de y_0 = +1 y por π_0 la correspondiente probabilidad posterior. Por otra parte, deja $\delta_i \stackrel{\text{def}}{=} \Delta(x_0, x_i)$ sea la distancia entre x_0 y x_i. Tomando la etiqueta conocida y_i como la información sobre la etiqueta desconocida y_0, se considera la probabilidad posterior.

Ecuación 8. Probabilidad posterior

$$\pi_0 \stackrel{\text{def}}{=} P(y_0 = +1 | y_i)$$

Más específicamente, la regla de Bayes

Ecuación 9. Regla de bayes para π_0

$$\frac{\pi_0}{1-\pi_0} = \frac{P(y_i|y_0=+1)}{P(y_i|y_0=-1)} \cdot \frac{p_0}{1-p_0}$$

$$= \rho \cdot \frac{p_0}{1-p_0},$$

donde ρ es el índice de probabilidad tomando logaritmos en ambos lados, se obtiene

Ecuación 10. Logarítmica

$$\log\left(\frac{\pi_0}{1-\pi_0}\right) = \log(\rho) + \omega_0$$

$$\omega_0 = \log(p_0) - \log(1-p_0)$$

El modelo (10) aún requiere la especificación del índice de probabilidad ρ. A fin de que obedecer el principio básico que subyace a IBL, este último debería ser una función de distancia δ_i. De hecho, ρ debería ser grande para $\delta_i \to 0\ si\ y_i = +1$ y pequeño si $y_i = -1$: observando una instancia muy cercana x_i con la etiqueta $y_i = +1 (y_i = -1)$ hace $y_0 = +1$ que sea menos probable en comparación con $y_i = -1$. Además, p debería tender a 1 como $\delta_i \to \infty$: si x_i está demasiado lejos, su etiqueta no proporciona ninguna evidencia, ni en $y_0 = +1$, ni en $y_0 = -1$. Una función parametrizada que satisface estas propiedades es:

Ecuación 11. Índice de probabilidad en la función de distancia

$$\rho = \rho(\delta) \stackrel{\text{def}}{=} \exp(y_i \cdot \frac{\alpha}{\delta})$$

Donde $\alpha > 0$ es una constante. Tenga en cuenta que la elección de una forma funcional especial para ρ es bastante comparable a la especificación del kernel utilizada en la estimación de densidad basada en kernel (no paramétrica), así como a la elección del peso de la función en la estimación NN ponderada. $\rho(\delta)$ en realidad determina la probabilidad de que dos instancias cuya distancia viene dada por $\delta = \Delta(x_0, x_i)$ tienen

la misma etiqueta. Ahora, tomando la muestra completa del vecindario N(x_0) de x_0 teniendo en cuenta y como en el enfoque ingenuo de Bayes haciendo la suposición simplificadora de la independencia condicional, se tiene:

Ecuación 12. Dependencia condicional

$$\log\left(\frac{\pi_0}{1-\pi_0}\right) = \omega_0 + \alpha \sum_{x_i N(x_0)} \frac{y_i}{\delta_i}$$

$$= \omega_0 + \alpha \cdot \omega_+(x_0)$$

Donde $\omega_+(x_0)$ se puede ver como un resumen de la evidencia a favor de la etiqueta +1. Como se puede ver, este último simplemente está dado por la suma de vecinos con etiqueta +1, ponderado por su distancia, menos la suma ponderada de vecinos con etiqueta -1. En lo que respecta a la clasificación de la consulta x_0, la decisión está determinada por la regla de la mano derecha en la ecuación (12). Desde este punto de vista, la ecuación (12) básicamente realiza una estimación NN ponderada o, dicho de otra forma, está "basada en un modelo" versión del aprendizaje basado en instancia. Aun así, difiere del simple esquema de NN en que incluye un término de ω_0, que juega el mismo papel que la probabilidad anterior en inferencia bayesiana.

Desde un punto de vista estadístico, ecuación (12) no es más que una ecuación de regresión logística. En otras palabras, tomando una vista "basada en características" del aprendizaje basado en instancia y la aplicación de un enfoque Bayesiano a la inferencia se reduce a la realización de IBL como Regresión logística.

Al introducir una medida de similitud, inversamente relacionada con la función de distancia, la ecuación (7) puede escribirse en la forma

Ecuación 13. Medida de similitud

$$\log\left(\frac{\pi_0}{1-\pi_0}\right) = \omega_0 + \alpha \sum_{x_i \in N(x_0)} k(x_0, x_i) \cdot y_i$$

Tenga en cuenta que, como un caso especial, este enfoque puede simular el clasificador estándar KNN, en la ecuación (7) concretamente estableciendo $\omega_0 = 0$ y de definiendo k en términos de la (data-dependencia) "KNN kernel" (Cheng y Hüllermeier 2009).

Ecuación 14. Pertenencia del vecindario

$$k(x_0,x_i) = \begin{cases} 1 \; si \; x_i \in \; N_k(x_0) \\ 0 \quad e.o.c \end{cases}$$

2.2.1 Estimación y clasificación

El parámetro α en (12) determina el peso de las pruebas

Ecuación 15. Vecinos con etiqueta

$$\omega_+(x_0) = \sum_{x_0 \in N(x_0)} k(x_0, x_i) \cdot y_i$$

y, por lo tanto, es en influencia en la estimación de probabilidad posterior π_0. De hecho, α desempeña el papel de un parámetro de suavizado (regularización). El más pequeño es elegido, la función de probabilidad estimada más uniforme (obtenida al aplicar (13) a todos puntos $x_0 \in X$) será. En el caso extremo donde $\alpha = 0$, uno obtiene una constante función (igual a ω_0).

Se puede lograr una especificación óptima de α adaptando este parámetro a los datos D, utilizando el método de máxima probabilidad (ML). Para cada muestra punto x_j denota por la validez de la muestra a favor de y = +1:

Ecuación 16. Máxima probabilidad (ML)

$$\omega_+(x_j) \stackrel{\text{def}}{=} \sum_{x_j \neq \; x_i \in N(x_j)} k(x_i, x_i) \cdot y_i$$

La función logarítmica de la probabilidad es entonces dada por el mapeo, y el parámetro óptimo α^* es el maximizador (17):

Ecuación 17. Mapeo

$$\alpha \longrightarrow \sum_{j:\, y_i = +1} w_0 + \alpha\omega_+(x_j) - \sum_{j=1}^{N} \log(1 + \exp\left(w_0 + \alpha\omega_+(x_j)\right))$$

Estos últimos pueden ser calculados mediante métodos estándar de regresión logística. La probabilidad posterior π_0 para la consulta es entonces dada por (18):

Ecuación 18. Probabilidad posterior con α^*

$$\pi_0 = \frac{\exp(\omega_+ + \alpha^*\omega_+(x_0))}{1 + \exp(\omega_+ + \alpha^*\omega_+(x_0))}$$

Para clasificar x_0, se aplica la regla de decisión (19):

Ecuación 19. Regla de decisión

$$\widehat{y_0} \stackrel{\text{def}}{=} \begin{cases} +1 \; si \; \pi_0 \geq 1/2 \\ -1 \; si \; \pi_0 < 1/2 \end{cases}$$

Se nombra al método descrito anteriormente como IBLR (Instance-Based Learning by Logistic Regression).

2.2.2 Incluyendo características adicionales

En la sección anterior, el aprendizaje basado en la instancia se ha integrado en la regresión logística, utilizando la información procedente de los vecinos de una consulta x_0 como una "característica" de esa consulta. En esta sección, consideramos una posible generalización de este enfoque, a saber, la idea de ampliar el modelo (13) teniendo en cuenta otras características de x_0:

Ecuación 20. ampliar modelo

$$\log\left(\frac{\pi_0}{1-\pi_0}\right) = \alpha\omega_+(x_0) + \sum_{\varphi_s \in \mathbb{F}} \beta_s \varphi_s(x_0)$$

Donde $\mathbb{F} = \{\varphi_0, \varphi_1 \ldots \varphi_r\}$ es un subconjunto de las características disponibles $\{\emptyset_0, \emptyset_1 \ldots \emptyset_n\}$ y $\varphi_0 = \emptyset_0 = 1$, lo que significa que β_0 juega el papel de ω_0. La ecuación (20) tiene mucho en común con la de regresión logística, excepto que $\omega_+(x_0)$ es una característica "no estándar". El enfoque (20), que llamaremos IBLR+, integra el enfoque basado en la instancia y el aprendizaje basado en modelos (basado en atributos) y, al estimar los coeficientes de regresión en (20), logra un equilibrio óptimo entre ambos enfoques. El modelo ex-tendido (20) puede interpretarse como un modelo de regresión logística de IBL, como se describe en el epígrafe 2.2, donde la variable ω_+ ya no es constante:

Ecuación 21. Sustitución de variable $\omega_0(x_0)$

$$\log\left(\frac{\pi_0}{1-\pi_0}\right) = \omega_0(x_0) + \alpha\omega_+(x_0)$$

Con $\omega_0(x_0) \stackrel{\text{def}}{=} \sum \beta_s \varphi_s(x_0)$ siendo una instancia específica determinada por la función (20) basada en modelos.

2.3 Algoritmo IBLR-ML

El algoritmo IBLR-ML fue desarrollado tomando en cuenta los modelos basados en adaptación específicamente los basados en instancias. Sus autores fueron Weiwei Cheng y Eyke Hullermeier. Estos algoritmos utilizan los métodos de clasificación tradicional y los mesclan con una serie de funciones para lograr la clasificación multi-etiqueta a continuación se le explicara cómo funciona este.

El modelo IBLR-ML combina el aprendizaje basado en instancias con la regresión logística. Este método de aprendizaje automatizado cuya idea básica es considerar la información que se deriva de las instancias vecinas a una de consulta como una característica de ella, borrando con ello la distinción en cierta medida entre el aprendizaje basado en instancia y basado en modelo. Esta idea se pone en práctica por medio de un algoritmo de aprendizaje que se da cuenta de la clasificación basada en instancia como regresión logística. Crea un nuevo conjunto de entrenamiento con la información de las etiquetas como nuevas características, finalmente para cada etiqueta creada entrena un clasificador basado en regresión logística. Este enfoque captura la interdependencia entre las etiquetas, además, de combinar la inferencia basada en modelo y basada en similitud para la clasificación multi-etiqueta (Rivero y Perla 2015).

Hasta ahora, solo se ha considerado el caso de la clasificación binaria. Para ampliar el enfoque a la clasificación multi-etiqueta con un conjunto de etiquetas L={$\lambda_1, \lambda_2 \ldots \lambda_m$}, la idea es entrenar una clase h_i para cada etiqueta. Para la i-esima etiqueta λ_i este clasificador es derivado del modelo

Ecuación 22. Clasificador IBLR-ML

$$\log\left(\frac{\pi_0{}^{(i)}}{1-\pi_0{}^{(i)}}\right) = \omega_0{}^{(i)} + \sum_{j=1}^{m} \alpha_j^{(i)} \cdot \omega_{+j}^{(i)}(x_0)$$

Donde $\pi_0{}^{(i)}$ denota la probabilidad (posterior) de que λ_1 es relevante para x_0 y es un resumen de la

Ecuación 23. Vecinos con la etiqueta

$$\omega_{+j}^{(i)}(x_0) = \sum_{x \in \mathrm{N}(x_0)} k(x_0, x) \cdot y_j(x)$$

presencia de la j-ésima etiqueta λ_j en el barrio de x_0; aquí, $y_j(x)$=+1 si λ_j está presente (relevante) para el vecino x, y , $y_j(x)$= -1 en caso de que esté ausente (no relevante). Obviamente, el enfoque de la ecuación (22) es capaz de tomar interdependencias entre clase etiquetas en consideración. Más específicamente, el coeficiente estimado $\alpha_j^{(i)}$ indica en qué el nivel de relevancia de la etiqueta λ_i es influenciada por la relevancia de λ_j .Un valor $\alpha_j^{(i)}$>0 significa que la presencia de λ_j hace la relevancia de λ_i con mayor probabilidad, es decir, hay una correlación positiva. En consecuencia, un coeficiente negativo indicaría una

correlación negativa. Tenga en cuenta que las probabilidades estimadas ${\pi_0}^{(i)}$ naturalmente se puede considerar como puntajes para las etiquetas λ_i . Por lo tanto, una clasificación de las etiquetas se obtiene simplemente ubicándolas en orden decreciente según sus probabilidades. Por otra parte, una predicción pura multi-etiqueta para x_0 se deriva de este ranking tomando un umbral de t=0.5.

Por supuesto, también es posible combinar el modelo (22) con la extensión propuesto en la Sección 2.4 Esto conduce a un modelo

Ecuación 24. Modelo extendido IBLR-ML+

$$\log\left(\frac{{\pi_0}^{(i)}}{1-{\pi_0}^{(i)}}\right)=\sum_{j=1}^{m}\alpha_j^{(i)}\cdot\omega_{+j}^{(i)}(x_0)+\sum_{\varphi_s\in\mathbb{F}}\beta_s^{(i)}\varphi_s(x_0)$$

Representan las ecuaciones (22) y (24) de IBLR a la clasificación multi-etiqueta como IBLR-ML e IBLR-ML+, respectivamente.

2.4 Pseudo-código de IBLR-ML y IBLR-ML+

Leyenda: las palabras señaladas con negrita son variables creadas:

- Se le aplica el algoritmo kNN sobre la instancia a clasificar para saber cuáles son sus k vecinos más cercanos (k=10) de la instancia a etiquetar (x_0).
- Se crea un vector con todas las etiquetas que tiene el DataSet.
- Se recorre el vector guardando las posiciones en (i)
- Se calcula la **confidencia:** sumando la cantidad de los k vecinos de x_0 que contengan la etiqueta en (i) y se divide ese total en la cantidad de vecinos (k). se termina de recorrer.
- **Si**
- Se está usando el método IBLR-ML+ que se determina pasando por parámetros una variable bool = True.
- Se guarda en el arreglo **attvalue** los atributos que tienen los k vecinos y seguido se almacenan los valores de **confidencia** (lo que cambia es la variable **attvalue** que a la hora de ver el grado de dependencia se tiene en cuenta los atributos de los vecinos, lo demás sigue igual).
- **Si no**
- Se guarda en el arreglo **attvalue** los valores de **confidencia**
- Se vuelve a recorrer la cantidad de etiquetas en (j) y se va almacenando en la última posición de **attvalue** el ultimo atributo de x_0 menos (j)
- Se crea una instancia **newinst** que va a contener (x_0,1, **attvalue**)
- Se crea un arreglo **confcorrected** que va a ir almacenando en la posición (j) la dependencia entre etiquetas llamando al método de regresión logística. Se termina el recorrido

- Del arreglo **confcorrected** se ordena de forma decreciente y se toman las etiquetas con un grado de pertenencia mayor de 0.5.
- Esas serán las etiquetas que contendrá la instancia insertada x_0.

2.5 Implementación en la herramienta scikit-multiearn

La herramienta scikit-multilearn solo ofrece solo API (Interfaz de programación de aplicaciones). No hay interfaz gráfica de usuario (GUI) disponibles. En esta herramienta fue incorporado el modelo IBLR-ML descrita en la sección 2.5, para ello se creó una nueva clase en el paquete *scikit-multilearn.adapt* que extienda de la clase *MLClassifierBase* como se ilustra en la figura 2.

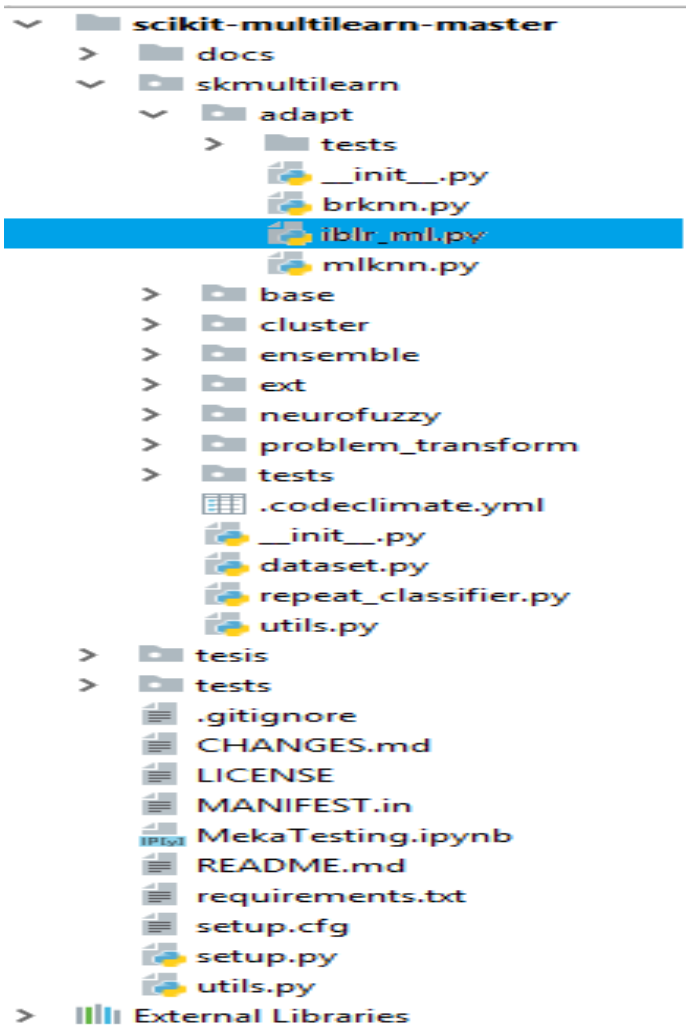


Figura 2. Incorporación de IBLR-ML a la herramienta scikit-multilearn

En la figura 3 se ilustra el diagrama de paquetes que se utilizan para la implementación del algoritmo en la herramienta scikit-multilearn.

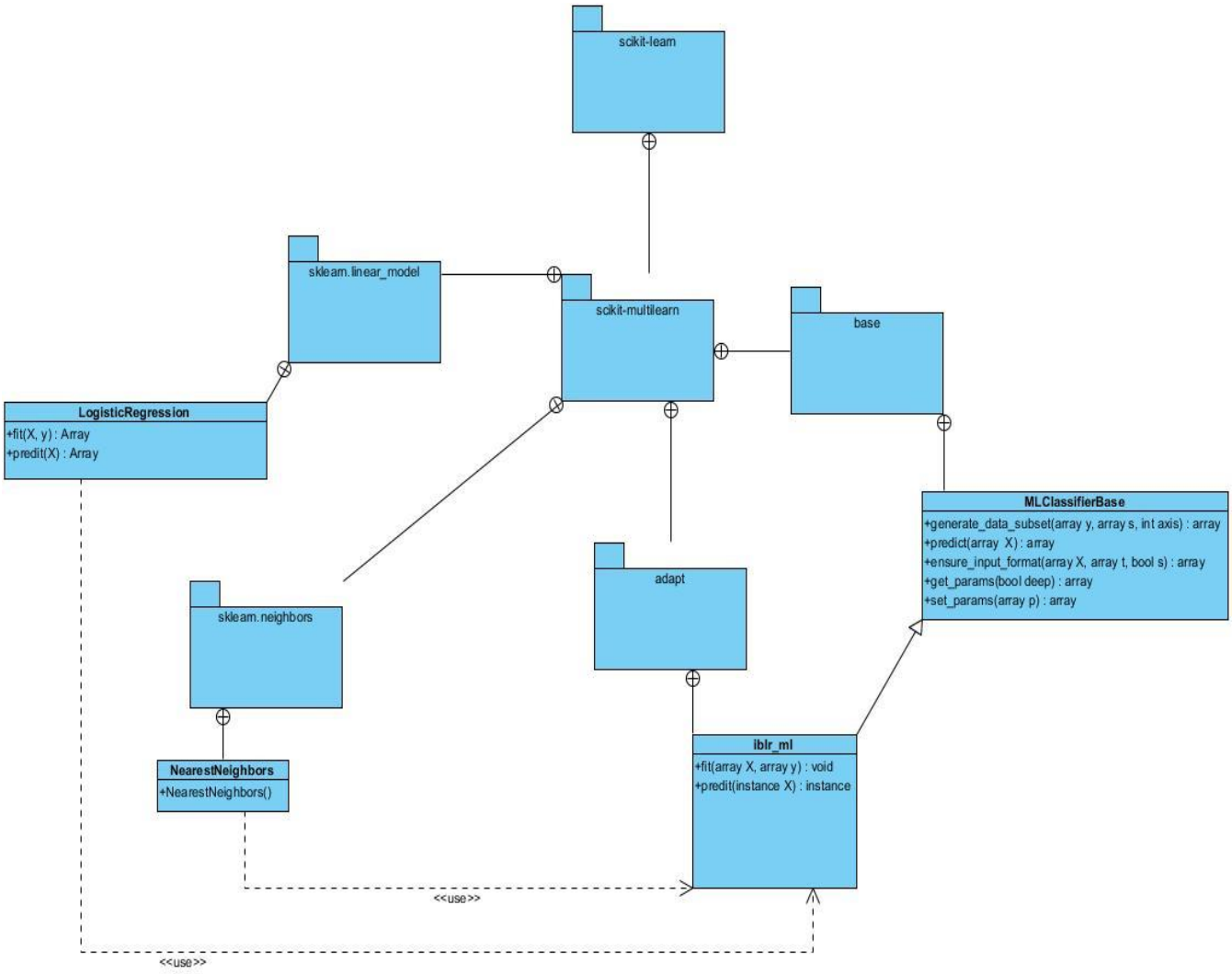


Figura 3. Diagrama de clases

Para poder utilizar el método IBLR-ML desde el paquete scikit-multilearn.adapt que es donde se encuentran los métodos de adaptación hace falta integrarlo a la librería. En la figura 4 se ilustran los códigos desarrollados en la clase __init__, de ese paquete para poder realizar esta integración.

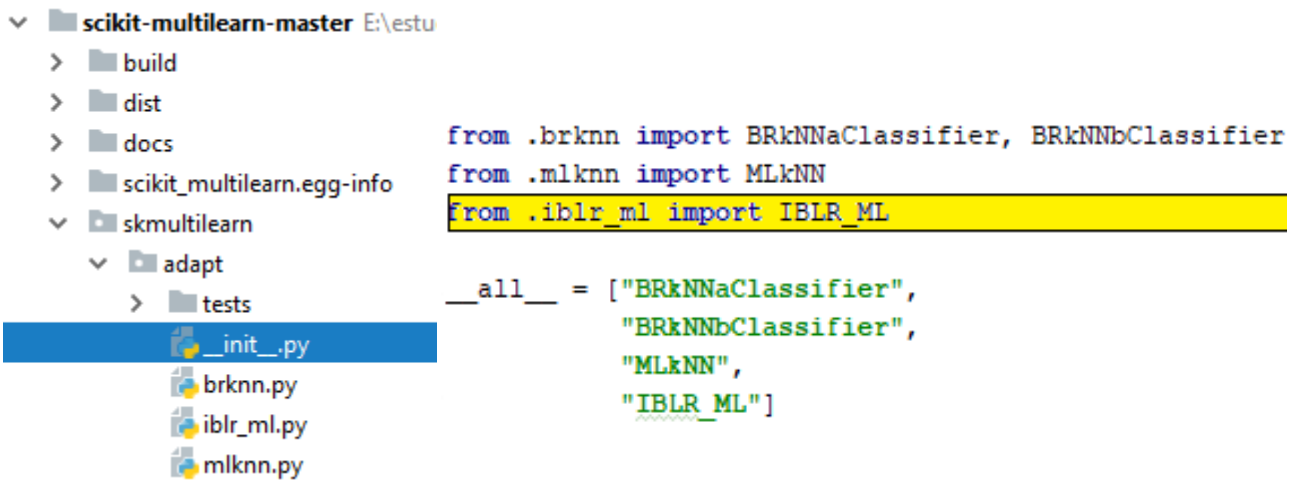


Figura 4. Códigos de integración

2.6 Estándares de codificación

Los estándares de código, son parte de las llamadas buenas practicas o mejores prácticas, estas son un conjunto no formal de reglas, que han ido surgiendo en las distintas comunidades de desarrolladores con el paso del tiempo y las cuales, bien aplicadas pueden incrementar la calidad de tu código, notablemente (Merkury 2017). La comunidad de usuarios de Python ha adoptado una guía de estilo que facilita la lectura del código y la consistencia entre programas de distintos usuarios. Esta guía no es de seguimiento obligatorio, pero es altamente recomendable. El documento completo se denomina PEP 8(Python Enhancement Proposal). Con el uso de este modelo se verán los siguientes estándares:

- 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. Por ejemplo;

```
# 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)
```

- 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',
       )
```

- Entre otras reglas que se pueden conocer con (A.E 2013)

2.7 Conclusiones parciales del capítulo

En el desarrollo del presente capítulo se:

a. Realizó la descripción matemática al algoritmo IBLR-ML y de sus métodos bases.
b. Desarrolló el pseudo-código del algoritmo IBLR-ML para facilitar la comprensión de su funcionamiento y la implementación de este en cualquier plataforma.
c. Ilustró la ubicación del algoritmo IBLR-ML dentro de la herramienta scikit-multilearn y el diagrama de las clases que usa para poder ejecutarse.
d. Identificaron los estándares correctos para la codificación y de esta forma lograr una mejor calidad en el código.

Capítulo 3. Implementación y resultados

En el actual capítulo se describe la implementación el algoritmo IBLR-ML en la herramienta scikit-multilearn. Se seleccionan y se describen los datos con los que se trabajan. Se realizan las pruebas necesarias para evaluar la factibilidad del algoritmo implementado, en cuanto a tiempo de respuesta y la precisión de los datos de salida, en relación a los estudiados en la literatura científica revisada. Se utiliza como metodología de evaluación la Validación cruzada, y también las métricas Acierto (accuracy) y Pérdida Hamming.

3.1 Pruebas unitarias

Las pruebas unitarias no es más que el método que puede invocar al código que se prueba y determina si el resultado obtenido es el esperado. Si es igual, entonces la prueba es exitosa, sino, falla. Si estos métodos se pueden ejecutar con un solo clic, entonces podemos ejecutar miles de ellos en muy poco tiempo (oscarcenteno 2016). Estas pruebas poseen las siguientes características.

- **Funcionan en memoria,** por lo que son muy rápidas. Cada una tarda pocos milisegundos en ejecutarse. No acceden a recursos externos como archivos, bases de datos o *webservices*. No requieren configuraciones manuales, por lo que funcionan con un solo clic.
- **Son repetibles.** Para que se pueda confiar en ellas, las pruebas deben ejecutarse siempre de la misma manera. Para poder ejecutarlas durante el día con cada cambio que realicemos, o en el futuro cuando un programador (diferente a quien creó el Sistema) haga otro cambio. Para lograrlo, una prueba unitaria nunca debe depender de algo que cambie en el tiempo, como, por ejemplo, la fecha actual, ni de alguna función aleatoria. En el ejemplo anterior, el resultado esperado es un número ya dado, así que la prueba siempre ejecutará de la misma manera (oscarcenteno 2016).

En el caso de la presente investigación se realizan las pruebas unitarias automatizadas con el IDE PyCharm a través del método unit-test se aplica las pruebas al código. Por ejemplo, en la figura 5 que se le muestra a continuación es la implementación del método “predict” donde cuenta con un parámetro que es la etiqueta a clasificar y retorna la etiqueta clasificada.

```
129     def predict(self, X):
130         mlo=[]
131         conf_corrected =[]
132         confidences =[]
133 
134         neighbors = [a[self.ignore_first_neighbours:] for a in
135                      self.knn.kneighbors(X, self.k + self.ignore_first_neighbours, return_distance=False)]
136         for instance in range(X.shape[0]):#ver a que equivale shape[0]
137             deltas = self.train_labels[neighbors[instance],].sum(axis=0)
138             for label in range(self.num_labels):
139                 confidences.append(deltas[0, label]/self.k)
140         attvalue =[]
141 
142         if self.addFeatures:
143             #preguntar si estos son los numeros de atributos de la instancia pasados por parametros ver si lo uso asi o
144                                                     #es que todas tienen la misma cantidad de atributos
145             for m in range(self.train_numbers_attributes-self.num_labels):  # descargar el paquete mulan mara python
146                 self.attvalue[m] =X[m]
147             attvalue = confidences[self.train_numbers_attributes - self.num_labels:len(confidences)]
148         else:
149             attvalue =confidences[0:len(confidences)]
150         for j in range(self.num_labels):
151             attvalue[- 1] = X.value(self.train_numbers_attributes - self.num_labels + j)
152             newInst = (1, attvalue)
153             conf_corrected[j] = self.classifier[j].distribution_for_instance(newInst)[1] #preguntarle de donde cargar esto
154 
155         for i in range(self.num_labels):
156             if conf_corrected[i]>=0.5:
157                 mlo[i]= conf_corrected[i]
158         return mlo
159 
```

Figura 5. Método “predict”

3.2 Selección de datos

En esta sección se describen brevemente los conjuntos de datos multi-etiqueta que se han utilizado en la fase experimental en la mayoría de los artículos. Estos conjuntos pertenecen a gran cantidad de dominios, como clasificación de música y escenas, clasificación de proteínas y genes, diagnóstico médico o clasificación de textos, y han sido ampliamente utilizados en otros trabajos relacionados con clasificación multi-etiqueta. Además, presentan características muy dispares tanto en tipos de datos (nominales y numéricos) como en sus métricas de cardinalidad y densidad, así como en el número de combinaciones de etiquetas.

El conjunto corel5K (Duygulu et al. 2002) contiene los datos utilizados para el artículo ECCV 2002 "Object Recognition as Machine Translation", de Pinar Duygulu, Kobus Barnard, Nando hacer Freitas, y David Forsyth. El dato está muy improvisado y tiene algunas anomalías. Cada segmento de la imagen está representado por 36 características. Dado que cada imagen tiene un número diferente de segmentos, que indique el número de segmentos utilizados en archivos separados.

El conjunto de datos medical (Sasaki, Rea y Ananiadou 2007) es un conjunto de datos desarrollado para una competición internacional, el 2007 International Challenge: Classifying Clinical Free Text Using Natural Language Processing, en la que se trataba de categorizar textos médicos por enfermedades. Es un conjunto nominal y destaca por el pequeño valor de densidad que posee.

El conjunto genbase (Diplaris et al. 2005) contiene información sobre bioinformática, sobre proteínas y la función que desempeñan. Genbase es un conjunto pequeño en número de instancias, además de no presentar un elevado número de combinaciones de etiquetas.

El conjunto de datos bibtext (Vlahavas, Tsoumakas y Katakis 2008) trata sobre clasificación textual, en este caso se trata de datos sobre etiquetado o tagging. La información presente es sobre numerosos documentos web etiquetados por usuarios. Se trata de un conjunto de datos nominal y sus características más reseñables son el elevado número de etiquetas, así como de combinaciones de etiquetas que presenta, ambos son los más elevados de los conjuntos presentados.

El conjunto enron (Keila y Skillicorn 2005) es un conjunto de datos también sobre textos, en concreto correos electrónicos hechos públicos a raíz del caso Enron y que fueron manualmente clasificados en categorías en la Universidad de Berkeley a lo largo del proyecto Enron Email Analysis. Es un conjunto nominal en el que cada etiqueta representa una de las categorías a las que se ha asociado el correo correspondiente.

Se trabaja con datos ya normalizados (están listos para entrenar el algoritmo). Los DataSet que se utilizan para entrenar el algoritmo son los que contiene la herramienta Mulan, se seleccionan 5. La tabla 1 muestra los DataSet seleccionados, los cuales se describen con los siguientes parámetros: número de etiquetas (L), número de instancias (N), número de características (F), número promedio de etiquetas (LC), LC/L densidad de etiquetas (LC/L).

Tabla 1. Descripción de los DataSet

	L	N	F	LC	LC/L
Corel5k	374	5000	499	3.522	0.94%
Bibtex	159	7395	1836	2.402	1.51%
Enron	53	1702	1001	3.378	6.37%
Medical	45	978	1449	1.245	2.76%
Genebase	27	662	1186	1.252	4.63%

Fuente: elaboración propia

3.3 Metodología de validación cruzada para la evaluación del algoritmo

El método de validación cruzada se utiliza para evaluar los resultados de un análisis estadístico y garantizar que son independientes de la partición, entre datos de entrenamiento y prueba. En la presente investigación se utiliza este método comparándolo con cuatro algoritmos que son los planteado en el estudio del estado del arte (son los algoritmos MLKNN, BRKNNa, BRKNNb y IBLR-ML), donde los datos se dividen en 10 particiones (k=10) y se utilizan tres experimentos que van a coincidir con las dos métricas que se le aplican a los algoritmos.

3.3.1 Experimento 1

El experimento 1, se basa en la métrica de Acierto (accuracy)que no es más que la clasificación binaria, que es la que calcula el porcentaje de ocurrencia de las etiquetas relevantes predefinidas en el subconjunto que creo el clasificador más relevante. A continuación, se muestran la tabla 2 y la figura 6 con los resultados después de haber ejecutado la métrica en los algoritmos.

Tabla 2. Resultados de ejecución de los algoritmos del experimento 1

	MLkNN	BRkNNa	BRkNNb	IBLR-ML
Corel5k	0,002000	0,002200	0,000000	0,002300
Bibtex	0,027435	0,055849	0,000270	0,002799
Enron	0,389542	0,251469	0,000588	0,624559
Medical	0,747444	0,708589	0,000000	0,759932
Genebase	0,913897	0,996979	0,000000	0,996982

Fuente: elaboración propia

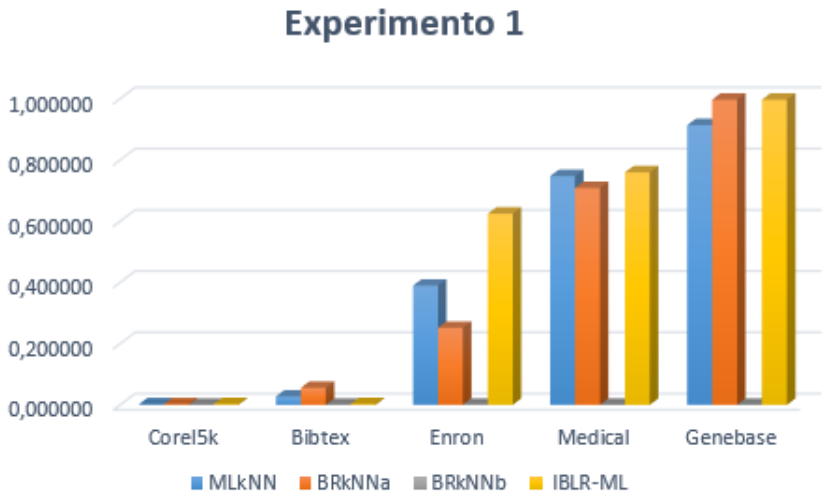


Figura 6. Gráfico de barra del resultado del experimento 1

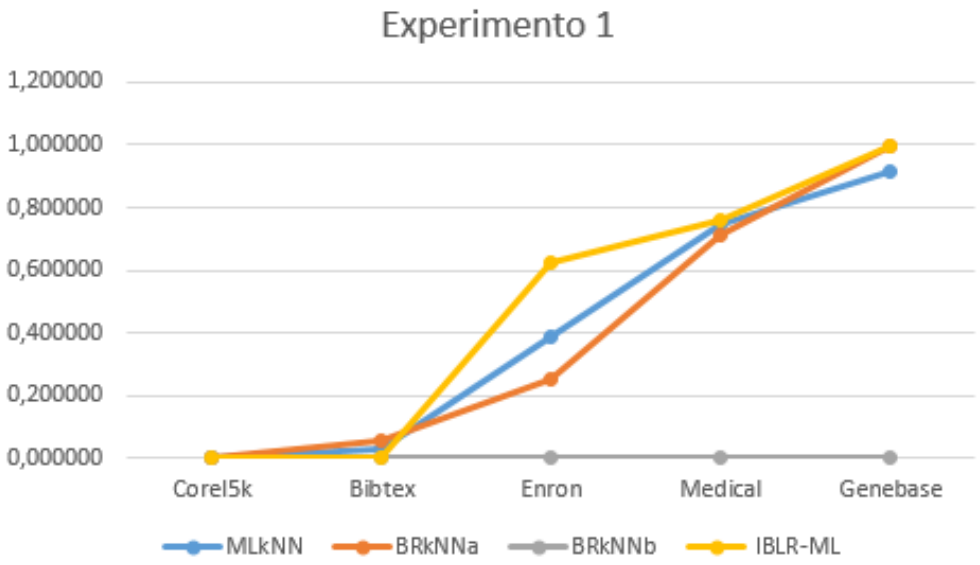


Figura 7. Gráfico del resultado del experimento 1

Los resultados arrojados después de ejecutar los algoritmos para el experimento 1, que se visualiza en las figuras 6 y 7 se puede afirmar que el algoritmo IBLR-ML de la muestra de 5 base de datos en 4 es mayor el porcentaje de ocurrencia de las etiquetas relevantes.

3.3.2 Experimento 2

El experimento 2, se basa en la métrica de Hamming loss, que no es más que la evaluación, que es la que calcula la fracción de etiquetas erróneas con respecto al total de etiquetas. A continuación, se muestra la tabla 3 y la figura 8 con los resultados después de haber ejecutado la métrica en los algoritmos.

Tabla 3. Resultados de ejecución de los algoritmos del experimento 2

	MLkNN	BRkNNa	BRkNNb	IBLR-ML
Corel5k	0,051704	0,016748	0,034386	0,015778
Bibtex	0,055674	0,055583	0,145686	0,065674
Enron	0,028734	0,08066	0,551538	0,012299
Medical	0,006771	0,007885	0,852738	0,006224
Genebase	0,002283	0,000112	0,999888	0,00011

Fuente: elaboración propia

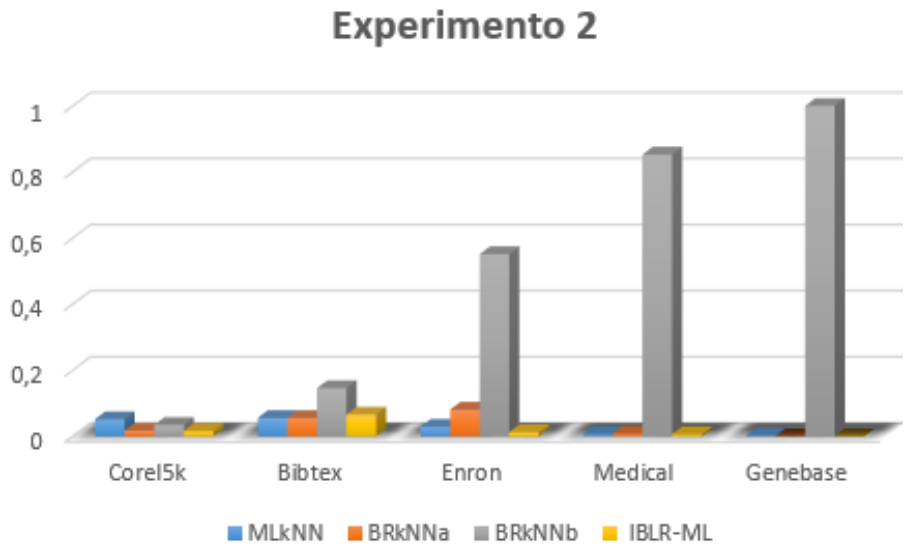


Figura 8. Gráfico de barra del resultado del experimento 2

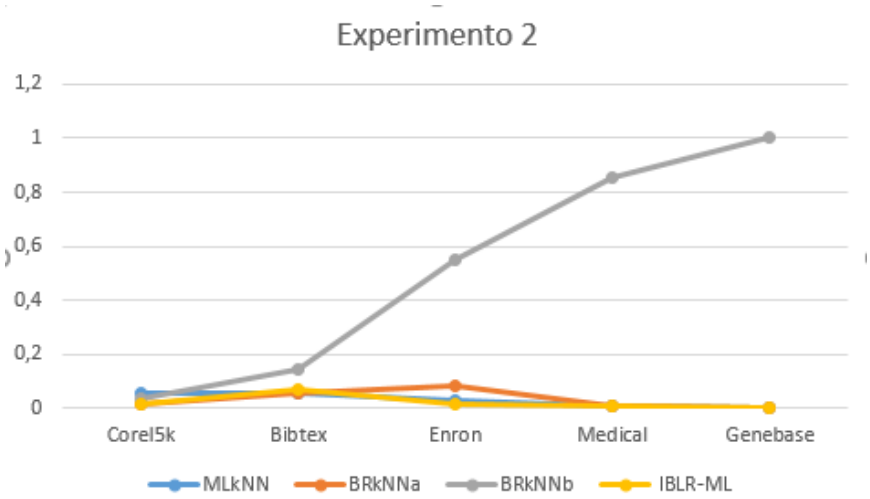


Figura 9. Gráfico del resultado del experimento 2

Los resultados arrojados después de ejecutar los algoritmos para el experimento 1, que se visualiza en las figuras 8 y 9 se puede afirmar que el algoritmo IBLR-ML de la muestra de 5 base de datos en 4 es menor el porcentaje de etiquetas erróneas con respecto al total de etiquetas.

3.4 Prueba de Friedman

Se aplica la prueba de Friedman para la variable por ciento de clasificaciones correctas, con un intervalo de confianza de 95%. La hipótesis nula consiste en asumir que todos los algoritmos son estadísticamente equivalentes. La prueba se realiza con un p-value que arroja el método Friedman y si el valor está por debajo de 0.05 se debe rechazar la hipótesis nula y por tanto se puede afirmar que existen diferencias significativas entre los algoritmos planteados. Para detectarlas se aplica la prueba post hoc con la corrección de Finner (García et al. 2010).

3.4.1 Prueba de Friedman para el experimento 1

Se realiza la prueba de Friedman para la variable por ciento de clasificaciones correctas. Se plantea la hipótesis nula que consiste que en asumir que no hay diferencia significativa entre los algoritmos, después se calcula el p-value donde es igual a 0.001452. Por lo que se puede concluir que hay diferencia significativa entre los algoritmos planteados por lo que se rechaza la hipótesis nula. En la figura 10 (ver anexo 1) se puede observar que la distancia crítica que marca el límite de significación no incluye al algoritmo BRKNNb. Por otra parte, las diferencias no son significativas con respecto a los demás algoritmos, pero aun así el algoritmo IBLR-ML es mejor los resultados teniendo en cuenta el criterio de Accuracy.

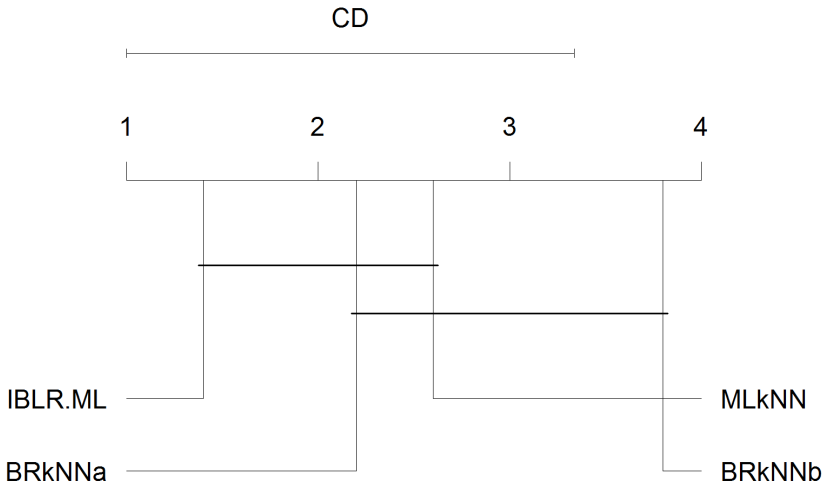


Figura 10. Gráfica que ilustra las diferencias significativas del experimento 1

3.4.2 Prueba de Friedman para el experimento 2

Se realiza la prueba de Friedman para la variable por ciento de clasificaciones correctas. Se plantea la hipótesis nula que consiste que en asumir que no hay diferencia significativa entre los algoritmos, después se calcula el p-value donde es igual a 0.009731. Por lo que se puede concluir que hay diferencia significativa entre los algoritmos planteados por lo que se rechaza la hipótesis nula. En la figura 11 (ver anexo 2) se puede observar que la distancia crítica que marca el límite de significación no incluye al algoritmo IBLR-ML, por otra parte, las diferencias no son significativas con respecto a los demás algoritmos, pero aun así el algoritmo IBLR-ML es mejor los resultados ya que tiene menor porciento de etiquetas clasificadas incorrectamente por ciento calculado con la métrica Hamming loss.

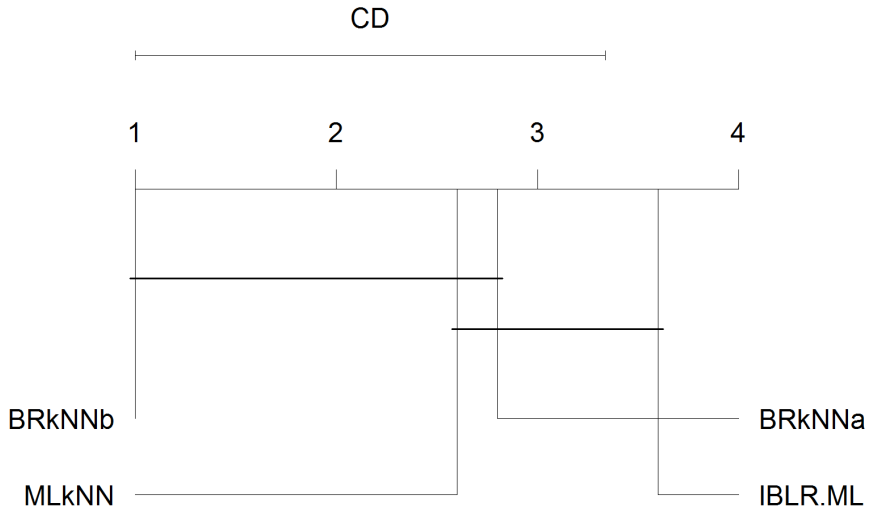


Figura 11. Gráfica que ilustra las diferencias significativas del experimento 2

3.5 Conclusiones parciales del capítulo

En este capítulo se realizaron las pruebas para la validación del algoritmo IBLR-ML resultando que:

a. Al aplicar la validación cruzada a los algoritmos del estado del arte en las bases de datos seleccionadas se calcularon los porcientos de acierto y de error de etiquetas asignadas y se demostró que el algoritmo IBLR-ML mejora en cuanto a porcientos de etiquetas asignadas correctamente y disminuye el porciento de etiquetas asignadas incorrectamente.
b. Se realiza la prueba de Friedman a través de la prueba de post hoc con la corrección de Finner donde en el experimento 1 clasifica mejor el algoritmo IBLR-ML teniendo diferencia significativa y para el experimento 2, de clasificación de etiquetas incorrectas, da resultados favorables sin diferencias significativas.

Conclusiones

A partir de la sistematización de los principales referentes teóricos que sustenta la investigación se confirma que la herramienta computacional scikit-multilearn posee ventajas sobre las otras existentes y que el algoritmo IBLR-ML mejora en cuanto a precisión, a los algoritmos de métodos de adaptación basados en instancias que contiene esta herramienta, por lo que se evidencia la necesidad de implementar dicho algoritmo en la herramienta scikit-multilearn.

Con el desarrollo del diseño teórico se logró una descripción legible del algoritmo para comprender mejor su funcionamiento y facilitar la implementación. Se definió la ubicación, que clases necesitó el algoritmo IBLR-ML para su creación y los estándares de codificación que se utilizaron con el fin de lograr un código limpio y de fácil entendimiento.

Los métodos científicos empleados para la validación del algoritmo IBLR-ML permitieron comprobar que la solución propuesta contribuye a mejorar la eficacia en la clasificación de las etiquetas con respecto a los diferentes algoritmos analizado en la investigación.

Recomendaciones

Implementar otros algoritmos de métodos de adaptación basados en instancias en la herramienta computacional scikit-multilearn, tales como: GMLkNN y DMLkNN.

Realizar un componente a la herramienta computacional scikit-multilearn para la transformación de las bases de datos antes de aplicar los algoritmos de métodos de adaptación basados en instancias.

Referencias Bibliográficas

ALEJANDROCASSIS, 2015. Aprendizaje Supervisado. *Inteligencia Artificial 101* [en línea]. [Consulta: 20 junio 2018]. Disponible en: https://inteligenciaartificial101.wordpress.com/2015/10/20/aprendizaje-supervisado/.

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

BATISTA, R. y RAMÍREZ PÉREZ, J., 2015. Fig. 3: Etapas del Descubrimiento de Conocimiento en Base de Datos... *ResearchGate* [en línea]. [Consulta: 22 junio 2018]. Disponible en: https://www.researchgate.net/figure/Etapas-del-Descubrimiento-de-Conocimiento-en-Base-de-Datos-KDD-por-sus-siglas-en_fig4_316212474.

CALONSO, 2014. BasadosEnInstancias-1.pd. [en línea]. [Consulta: 24 enero 2018]. Disponible en: file:///C:/Users/Mandy/AppData/Local/Temp/BasadosEnInstancias-1.pdf.

CHENG, W. y HÜLLERMEIER, E., 2009. Combining instance-based learning and logistic regression for multilabel classification. *Machine Learning*, vol. 76, no. 2-3, pp. 211-225. ISSN 0885-6125, 1573-0565. DOI 10.1007/s10994-009-5127-5.

COVER, T. y HART, P., 2006. Nearest Neighbor Pattern Classification. *IEEE Trans. Inf. Theor.*, vol. 13, no. 1, pp. 21–27. ISSN 0018-9448. DOI 10.1109/TIT.1967.1053964.

DATA, M., 2004. KDD: Knowledge Discovery in Databases. *Minerva* [en línea]. [Consulta: 8 junio 2018]. Disponible en: http://mnrva.io/kdd-platform.html.

DIPLARIS, S., TSOUMAKAS, G., MITKAS, P.A. y VLAHAVAS, I., 2005. Protein Classification with Multiple Algorithms. En: DOI: 10.1007/11573036_42, *Advances in Informatics* [en línea]. S.l.: Springer, Berlin, Heidelberg, pp. 448-456. [Consulta: 21 junio 2018]. Disponible en: https://link.springer.com/chapter/10.1007/11573036_42.

DUYGULU, P., BARNARD, K., FREITAS, J.F.G. de y FORSYTH, D.A., 2002. Object Recognition as Machine Translation: Learning a Lexicon for a Fixed Image Vocabulary. En: DOI: 10.1007/3-540-47979-1_7, *Computer Vision — ECCV 2002* [en línea]. S.l.: Springer, Berlin, Heidelberg, pp. 97-112. [Consulta: 21 junio 2018]. Disponible en: https://link.springer.com/chapter/10.1007/3-540-47979-1_7.

FAYYAD, U.M., SHAPIRO, G.P. y SMYTH, P., 1996. From Data Mining to Knowledge Discovery in Databases - 1131. [en línea]. [Consulta: 8 junio 2018]. Disponible en: https://www.aaai.org/ojs/index.php/aimagazine/article/viewFile/1230/1131.

GIL MARTÍNEZ, 2011. Una mejora de los mo delos apilados para la clasi_cación multi-etiqueta. [en línea]. [Consulta: 4 mayo 2018]. Disponible en: http://www.aic.uniovi.es/~juanjo/Homepage_Juan_Jose_del_Coz_Velasco/juanjo_files/DBR-CAEPIA11-spanish.pdf.

GOLD, K. y PETROSINO, A., 2010. Using information gain to build meaningful decision forests for multilabel classification. [en línea]. S.l.: IEEE, pp. 58-63. ISBN 978-1-4244-6900-0. DOI 10.1109/DEVLRN.2010.5578864. Disponible en: http://ieeexplore.ieee.org/document/5578864/.

GONZÁLEZ, A., 2014. ¿Qué es Machine Learning? [en línea]. [Consulta: 13 junio 2018]. Disponible en: http://cleverdata.io/que-es-machine-learning-big-data/.

GONZÁLEZ, G., 2017. Python se ha convertido en el lenguaje de programación que crece más rápido. *Genbeta* [en línea]. [Consulta: 1 junio 2018]. Disponible en: https://www.genbeta.com/actualidad/python-se-ha-convertido-en-el-lenguaje-de-programacion-que-crece-mas-rapido.

HALL, M., FRANK, E., HOLMES, G., PFAHRINGER, B., REUTEMANN, P. y WITTEN, I.H., 2009. The WEKA Data Mining Software: An Update. *SIGKDD Explor. Newsl.*, vol. 11, no. 1, pp. 10–18. ISSN 1931-0145. DOI 10.1145/1656274.1656278.

JOANNEUM, f, 2005. *Cross-Validation Explained*. 2005. S.l.: s.n.

KEILA, P.S. y SKILLICORN, D.B., 2005. Structure in the Enron Email Dataset. *Comput. Math. Organ. Theory*, vol. 11, no. 3, pp. 183–199. ISSN 1381-298X. DOI 10.1007/s10588-005-5379-y.

LANDA, J., 2016. ¿Qué es KDD y Minería de Datos? –. [en línea]. [Consulta: 6 junio 2018]. Disponible en: //fcojlanda.me/es/sin-categoria-es/kdd-y-mineria-de-datos-espanol/.

LARA TORRALBO, J.A., 2010. Tesis Doctoral - JUAN_ALFONSO_LARA_TORRALBO.pdf. [en línea]. [Consulta: 8 junio 2018]. Disponible en: http://oa.upm.es/5729/1/JUAN_ALFONSO_LARA_TORRALBO.pdf.

LARRAÑAGA, P., LI, G. y BIELZA, C., 2011. Multi-dimensional classification with Bayesian networks. *International Journal of Approximate Reasoning*, vol. 52, no. 6, pp. 705-727. ISSN 0888-613X. DOI 10.1016/j.ijar.2011.01.007.

LINGZHANGZHI-HUAZHOU, M. y HUAZHOU, Z., 2007. ML-KNN: A lazy learning approach to multi-label learning. *Pattern Recognition*, vol. 40, no. 7, pp. 2038-2048. ISSN 0031-3203. DOI 10.1016/j.patcog.2006.12.019.

MARIA JOSE, RAMIREZ QUINTANA, C. y RAMIREZ, F., 2004. Alianza SIDALC. [en línea]. [Consulta: 22 junio 2018]. Disponible en: http://www.sidalc.net/cgi-bin/wxis.exe/?IsisScript=UCC.xis&method=post&formato=2&cantidad=1&expresion=mfn=064234.

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

MOORE, 2001. crossvalidation_slides_Moore_CMU.pdf. [en línea]. [Consulta: 19 junio 2018]. Disponible en: https://clm.utexas.edu/fietelab/QuantNeuro/readings/crossvalidation_slides_Moore_CMU.pdf.

MORALES, E., 2017. Clasificación Multi-Etiqueta - multilabel.pdf. [en línea]. [Consulta: 18 junio 2018]. Disponible en: https://ccc.inaoep.mx/~emorales/Cursos/Aprendizaje2/Acetatos/multilabel.pdf.

NILSSON, N.J., 1996. Introduction to machine learning. [en línea]. [Consulta: 22 junio 2018]. Disponible en: http://thuvien.thanglong.edu.vn:8081/dspace/bitstream/DHTL_123456789/4037/5/cs516-2.pdf.

OSCARCENTENO, 2016. Pruebas unitarias: Definición y sus dos características. *Software Mantenible.com* [en línea]. [Consulta: 9 junio 2018]. Disponible en: https://softwaremantenible.com/2016/09/06/pruebas-unitarias-definicion-y-caracteristicas/.

PASILLAS, A., 2017. ¿Qué es machine learning? [Guía completa para principiantes]. [en línea]. [Consulta: 13 junio 2018]. Disponible en: https://blog.adext.com/es/machine-learning-guia-completa.

PEP 8 en Español - Guía de estilo para el código Python | Recursos Python - pep8es.pdf. [en línea], 2013. [Consulta: 9 junio 2018]. Disponible en: http://recursospython.com/pep8es.pdf.

PRESSMAN, R., 2007. Ingenieria de Software Un Enfoque Practico.6th.edicion-.Roger.pressman.1. *Scribd* [en línea]. [Consulta: 19 junio 2018]. Disponible en: https://es.scribd.com/doc/27182020/Ingenieria-de-Software-Un-Enfoque-Practico-6th-edicion-Roger-pressman-1.

RAK, R., KURGAN, L. y REFORMAT, M., 2008. A tree-projection-based algorithm for multi-label recurrent-item associative-classification rule generation. *Data & Knowledge Engineering*, vol. 64, no. 1, pp. 171-197. ISSN 0169-023X. DOI 10.1016/j.datak.2007.05.006.

RIVERO, M. y PERLA, A., 2015. *GMLKNN: modelo basado en instancias para el aprendizaje multi-etiqueta utilizando la distancia VDM* [en línea]. Thesis. S.l.: Universidad Central «Marta Abreu» de Las Villas. Facultad de Matemática, Física y Computación. Departamento Ciencias de la Computación. [Consulta: 7 diciembre 2017]. Disponible en: http://dspace.uclv.edu.cu:8089/xmlui/handle/123456789/7551.

RUIZ, S., 2017. El algoritmo K-NN y su importancia en el modelado de datos. *Analítica web* [en línea]. [Consulta: 1 junio 2018]. Disponible en: https://www.analiticaweb.es/algoritmo-knn-modelado-datos/.

SASAKI, Y., REA, B. y ANANIADOU, S., 2007. Multi-topic Aspects in Clinical Text Classification. *Proceedings of the 2007 IEEE International Conference on Bioinformatics and Biomedicine* [en línea]. Washington, DC, USA: IEEE Computer Society, pp. 62–70. ISBN 978-0-7695-3031-4. DOI 10.1109/BIBM.2007.53. Disponible en: https://doi.org/10.1109/BIBM.2007.53.

SCHNEIDER, J. y MOORE, A., 1997. A locally weighted learning tutorial using vizier. [en línea]. [Consulta: 19 junio 2018]. Disponible en: https://www.ri.cmu.edu/pub_files/pub2/schneider_jeff_2000_1/schneider_jeff_2000_1.pdf.

SEIJAS CANDELAS, L., 2016. Microsoft Word - libro digital RIUMA Actas Cap1 - Libro de actas RIUMA Cap41.pdf. [en línea]. [Consulta: 18 junio 2018]. Disponible en: https://riuma.uma.es/xmlui/bitstream/handle/10630/12478/Libro%20de%20actas%20RIUMA%20Cap41.pdf?sequence=1&isAllowed=y.

SIERRA MARTÍNEZ, J. y PÉREZ GONZÁLEZ, P., 2017. *Algoritmo de transformación LNCMTR para problemas de predicción con salidas compuestas integrado a MULAN.* S.l.: s.n.

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_environment_for_performing_multi-label_classification.

TSOUMAKAS, G. y KATAKIS, I., 2007. A Review of Multi-Label Classification Methods - Multi-Label Classification: An Overview.pdf. [en línea]. [Consulta: 4 mayo 2018]. Disponible en: https://aetos.it.teithe.gr/~stoug/odep/papers/Multi-Label%20Classification:%20An%20Overview.pdf.

TSOUMAKAS, G., SPYROMITROS-XIOUFIS, E., VILCEK, J. y VLAHAVAS, I., 2011. MULAN: A Java Library for Multi-Label Learning. *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2411-2414. ISSN ISSN 1533-7928.

VALLIM, R.M.M., GOLDBERG, D.E., LLORÀ, X., DUQUE, T.S.P.C. y CARVALHO, A.C.P.L.F., 2008. A New Approach for Multi-label Classification Based on Default Hierarchies and Organizational Learning. *Proceedings of the 10th Annual Conference Companion on Genetic and Evolutionary Computation* [en línea]. New York, NY, USA: ACM, pp. 2017–2022. ISBN 978-1-60558-131-6. DOI 10.1145/1388969.1389015. Disponible en: http://doi.acm.org/10.1145/1388969.1389015.

VLAHAVAS, I., TSOUMAKAS, G. y KATAKIS, I., 2008. Multilabel Text Classification for Automated Tag Suggestion - Semantic Scholar. [en línea]. [Consulta: 21 junio 2018]. Disponible en: /paper/Multilabel-Text-Classification-for-Automated-Tag-Katakis-Tsoumakas/157099d6ffd3ffca8cfca7955aff7c5f1a979ac9.

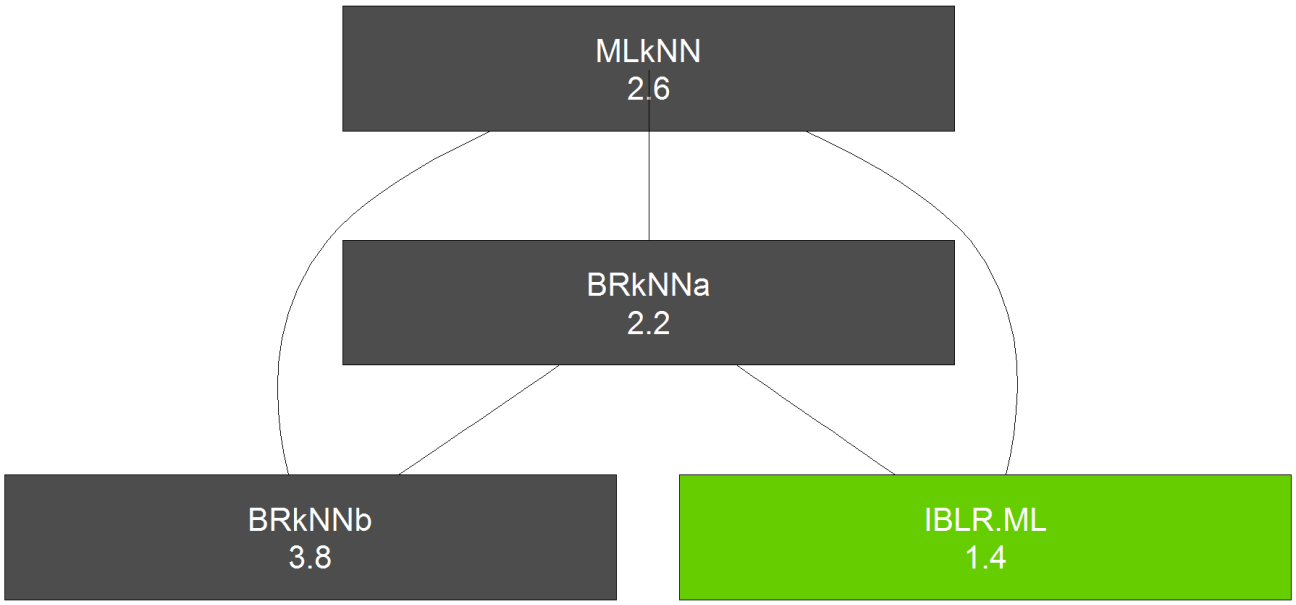
WITTEN, I.H., FRANK, E., TRIGG, L.E., HALL, M.A., HOLMES, G. y CUNNINGHAM, S.J., 2011. Weka: Practical machine learning tools and techniques with Java implementations. [en línea]. Working Paper. S.l.: [Consulta: 22 junio 2018]. Disponible en: https://researchcommons.waikato.ac.nz/handle/10289/1040.

XU, J., 2011. Multi-Label Weighted <Emphasis Type=«Italic»>k</Emphasis>-Nearest Neighbor Classifier with Adaptive Weight Estimation. En: DOI: 10.1007/978-3-642-24958-7_10, *Neural Information Processing* [en línea]. S.l.: Springer, Berlin, Heidelberg, pp. 79-88. [Consulta: 14 diciembre 2017]. Disponible en: https://link.springer.com/chapter/10.1007/978-3-642-24958-7_10.

ZHANG, M.-L., 2009. M<Emphasis Type=«SmallCaps»>l-rbf</Emphasis>: RBF Neural Networks for Multi-Label Learning. *Neural Processing Letters*, vol. 29, no. 2, pp. 61-74. ISSN 1370-4621, 1573-773X. DOI 10.1007/s11063-009-9095-3.

Anexos

Anexo 1: Ranking de Friedman del experimento 1



Anexo 2: Ranking de Friedman del experimento 2