UNIVERSIDAD CENTRAL "MARTA ABREU" DE LAS VILLAS
FACULTAD DE MATEMÁTICA, FÍSICA Y COMPUTACIÓN
CENTRO DE ESTUDIOS DE INFORMÁTICA
DPTO. INTELIGENCIA ARTIFICIAL

EN OPCIÓN AL TÍTULO DE MÁSTER EN CIENCIA DE LA COMPUTACIÓN

GMLkNN: MODELO BASADO EN INSTANCIAS PARA EL APRENDIZAJE MULTI-ETIQUETA UTILIZANDO LA DISTANCIA VDM

AUTOR: LIC. ADIS PERLA MARIÑO RIVERO
TUTOR: DRA. YANET RODRÍGUEZ SARABIA

RESUMEN

En los últimos años el aprendizaje multi-etiqueta ha cobrado gran interés en los investigadores debido a la gran cantidad de aplicaciones modernas que presentan estos tipos de datos. Los algoritmos perezosos ML*k*NN, BR*k*NN y IBLR-ML utilizan el tradicional *k*NN pero dirigido al trabajo con datos multi-etiqueta, su capacidad de generalización depende en gran medida de la definición de la función de distancia que se utiliza para comparar la instancia de prueba y las instancias entrenadas. El algoritmo *k*NN utiliza HEOM como función de distancia, esta solo considera una superficial diferencia entre los objetos comparados y es demasiado simplista para el manejo de atributos nominales ya que no hace uso de la información adicional proporcionada por los valores de atributos nominales que pueden ayudar en la generalización mientras que la función VDM reduce el impacto de atributos irrelevantes en la precisión de la clasificación sin necesidad del pre-procesamiento de los datos además de tener en cuenta la relación de cada rasgo con el rasgo objetivo. En este trabajo se presenta un modelo llamado GML*k*NN el cual utiliza la distancia VDM en la búsqueda de los vecinos, este se compara con los algoritmos perezosos dichos anteriormente y obtiene mejores resultados en 12 de los 15 conjuntos de datos de la experimentación. Se redefine GMLkNN teniendo en cuenta la dependencia entre las etiquetas, una valiosa información que se perdía anteriormente y se compara con IBLR-ML el cual también tiene esta característica y su comportamiento es superior a este último modelo. En una tercera etapa se inserta ruido en los conjuntos de datos y se aprecia como el modelo GML*k*NN es bastante robusto y estable.

ABSTRACT

During the last years, multi-label learning has gained considerable interest among researchers due to the large number of modern applications these data types present. The lazy learning algorithms MLkNN, BRkNN and IBLR-ML use the traditional k –NN, but oriented to work with multi - label data, its generalization ability depends largely on the definition of the distance function used to compare the test instance and the trained ones. The traditional k -NN algorithm uses Heom as distance function, it only considers a superficial difference between the compared objects and it is too simple to handle nominal attributes as it does not use the additional information provided by their values, while the VDM function reduces the impact of irrelevant attributes in the classification accuracy without the need of pre - processing the data. In this work we present a model called GMLkNN which uses the VDM distance in the process of finding the neighbors, this model is compared with the lazy learning algorithms mentioned before and it i sable to obtain better results for 12 of the 15 datasets used for the experiments. GMLkNN is redefined taking into account the dependency between labels, a valuable information that was lost earlier and it is compared with IBLR-ML, which also has this feature and its performance is superior to the latter model. In a third phase noise is inserted into the data sets and it is appreciated how the GMLkNN model is quite robust and stable.

Dedicatoria

A mis padres:Arelis y Mariño, gracias a ellos me he convertido en la persona que soy.

A mi abuela por siempre estar ahí en mis momentos difíciles.

A mi hermano Adonis, te quiero y te necesito mucho.

A mi tutora por tener el tiempo de atenderme.

A mi profe Marilin, por confiar en mi, por darme esperanzas y siempre tener un tiempo y una sonrisa.

A Geily por acompañarme es estos momentos de tristeza.

A todo mi departamento.

A mis amigos que no los nombro porque pudiera quedarse alguno.

ÍNDICE

INTRODUCCIÓN

El volumen de información almacenada en el mundo ha ido creciendo exponencialmente en las últimas décadas, y este crecimiento no tiene visos de detenerse. En este contexto, en el que se calcula que cada 20 meses se duplica el volumen de la información almacenada en bases de datos digitales [1] y en el que dispositivos electrónicos almacenan nuestras decisiones, sea en el supermercado, navegando por Internet o solicitando un crédito, es necesario tener mecanismos que permitan no sólo almacenar y recuperar eficientemente los datos, sino también procedimientos que permitan obtener información y conocimiento útil a partir de estos datos en bruto.

El hecho de que las bases de datos estén en muchas ocasiones saturadas de información, trae consigo que aumente la brecha entre el almacenamiento de esta información y su utilización real, es decir, la proporción entre datos e información útil decrece. Además, encerrada entre esta gran cantidad de datos sin duda hay mucho conocimiento útil, desaprovechado, y que simplemente está ahí, rodeado entre terabytes de ruido, en espera de ser hallado.

La Minería de Datos (*Data Mining*, DM) 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. El proceso idealmente debe ser automático y el objetivo principal es que los patrones descubiertos deben tener algún significado en el sentido de que puedan ser utilizados de alguna forma como conocimiento útil [1] (actualmente suele significar económicamente útil). Estos patrones, o dicho de manera más adecuada, estos conceptos o modelos nos permitirán hacer predicciones sobre nuevos datos que se almacenen.

Además, idealmente estos modelos deben estar expresados de manera inteligible, para que nos permitan no solamente hacer predicciones, sino también comprender por qué los datos están estructurados de dicha manera y no de otra.

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*, ML) [1].

Dentro del aprendizaje automático debemos destacar el Aprendizaje Inductivo, en el que se basan la mayoría de las tareas de la minería de datos [2].

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 [3]. Dentro del aprendizaje inductivo se suelen distinguir tres paradigmas clásicos de aprendizaje: el aprendizaje supervisado, el no supervisado y el aprendizaje con refuerzo [4].

El aprendizaje perezoso (*lazy learning*), al igual que en el aprendizaje inductivo, se aprende a partir de ejemplos, pero a diferencia de este:

- Se usa descripción extensional, sin generar descripciones intencionales.
- El proceso de aprendizaje y el proceso de usar el conocimiento aprendido para resolver nuevos problemas no se separan.
- El paso de generalización se retrasa para la fase de solución de problemas.

Cuando se resuelve un nuevo problema *P*, la solución de un problema viejo se transfiere (quizás transformada) a *P*. Hay una generalización implícita entre el viejo y el nuevo problema. Un método clásico de aprendizaje perezoso es el algoritmo de los *k* vecinos más cercanos (*k Nearest Neighbour*, *k*NN)[5]. Métodos de aprendizaje perezoso bien conocidos son: el aprendizaje basado en instancias, el razonamiento analógico, y el razonamiento basado en casos.

Los algoritmos de aprendizaje basado en instancias (*Instance-Based Learning*, IBL) pertenecen a la familia de algoritmos perezosos [6] [7], se caracterizan en [8] por tres comportamientos: almacenan todos los ejemplos de entrenamiento, difieren su procesamiento hasta que se presente una solicitud, y las solicitudes se responden por la combinación de ejemplos de entrenamiento seleccionados mediante un criterio de similitud y combinados en una función de predicción.

En el contexto del aprendizaje basado en instancias muchas propuestas se basan en propiedades estadísticas pre-computadas a partir de datos de entrenamiento disponible y puede ser interpretada en términos de probabilidades, en este aspecto la distancia más conocida se denominada VDM (*Value Difference Metric*)[7] la cual tiene en cuenta la relación de cada rasgo con el rasgo objetivo (distancia informada), uno de los criterios de similitud más sofisticados para comparar atributos nominales. Entre los modelos IBL podemos mencionar ConFuCiuS el cual se concibe como una variante del *k*NN utilizando como criterio de comparación a nivel de rasgo, algunas distancias basadas en la extensión y generalización de VDM son FVDM[9] y FADM[10].

Las técnicas de ML se han utilizado en minería de datos para resolver una gran cantidad de tareas. Estas tareas se suelen agrupar en dos grandes categorías: tareas descriptivas y predictivas.

Las tareas descriptivas son las que tratan de obtener informacion acerca de la estructura de los datos almacenados, entre ellas encontramos las tareas de Clustering y Asociación. Por otro lado,

las tareas predictivas tratan de predecir un comportamiento a partir de informacion ya almacenada. Entre las tareas predictivas encontramos las de Regresión y Clasificación.

La tarea de clasificación se definió originalmente considerando que la correspondencia entre instancias y clases es unívoca, sin embargo, hay muchos problemas en los que la restricción de una etiqueta por instancia no se cumple. Dentro de estos problemas merece especial atención el paradigma en el que un patrón puede estar asociado simultáneamente con más de una clase, la llamada Clasificación Multi-Etiqueta (*Multi-Label Classification*, MLC). Esta se ha aplicado para resolver múltiples problemas reales, en múltiples dominios, como categorización de documentos, bioinformática, clasificación de imágenes, o medicina entre otras.

Los algoritmos de aprendizaje perezosos se han utilizado con bastante éxito en este contexto de datos multi-etiqueta, en el que se puede mencionar el modelo ML*k*NN(*Multi-Label k-Nearest Neighbor*)[11] el cual es conocido y estudiado en gran medida en la comunidad de reconocimiento de patrones, tanto por su sencillez y su rendimiento. En este la definición de la función de distancia es fundamental para obtener una buena precisión en un conjunto de datos dado.

ML*k*NN utiliza la función HEOM (*Heterogeneous Euclidean-Overlap Metric*)[6] para comparar la instancia de prueba y la de entrenamiento en la búsqueda de los *k* vecinos más cercanos, pero su enfoque es demasiado simplista para el manejo de atributos nominales ya que no hace uso de la información adicional proporcionada por los valores de atributos nominales que pueden ayudar a su generalización.

Lo antes expuesto fundamenta como problema científico que al ser la clasificación multi-etiqueta un área de creciente interés dentro del aprendizaje automático se demanda la creación de nuevos modelos computacionales basados en instancias que propicien resolver con eficacia la diversidad de problemas que pueden ser abordados desde esta perspectiva.

El presente trabajo propone como objetivo general entonces: “desarrollar un modelo computacional basado en instancias para problemas de clasificación multi-etiqueta utilizando VDM.”

Para cumplir este objetivo general se plantean los objetivos específicos siguientes:

- Estudiar los modelos perezosos para clasificación multi-etiqueta.
- Extender el modelo utilizado en ConFuCiuS para clasificación multi-etiqueta en presencia de datos simbólicos.
- Implementar el modelo propuesto
- Validar la eficacia del modelo propuesto con datos sintéticos y en presencia de ruido.

Como preguntas de investigación nos planteamos:

1. ¿Es extensible el modelo ConFuCiuS a clasificación multi-etiqueta?.

2. ¿Existe un marco de trabajo que facilite la implementación del modelo propuesto?.

3. ¿El uso de una distancia informada mostrará ventajas en la clasificación multi-etiqueta con respecto a otras distancias utilizadas?.

Como hipótesis se plantea que la combinación de un enfoque IBL con una función de distancia informada mejora la eficacia de otros modelos similares para resolver problemas de clasificación multi-etiqueta.

ESTRUCTURA DE LA TESIS

El presente trabajo está estructurado en tres capítulos. El Capítulo1 introduce el problema de la clasificación multi-etiqueta, sus principales áreas de aplicación y una notación uniforme para el problema. Se hace un estudio de los principales algortimos perezosos adaptados para el trabajo con datos multi-etiqueta concentrándonos en tres de ellos y se detallan dos de las funciones de distancias más utilizadas.

En el Capítulo 2 después de revisar los principales algoritmos que hasta ahora se han propuesto, se presenta un nuevo modelo de aprendizaje para clasificación multi-etiqueta que utiliza la función de distancia VDM y después se redefine este modelo teniendo en cuenta la dependencia existente entre las etiquetas. Se observa en este capítulo la incorporación de la propuesta a la biblioteca de código abierto MULAN así como un análisis comparativo de la complejidad computacional de los dos modelos propuestos.

En el Capítulo 3 se hace una síntesis de las métricas más comúnmente utilizadas para medir el rendimiento de algoritmos de clasificación multi-etiqueta, ya que debido a sus características, distintas del problema de la clasificación clásica, el rendimiento de los algoritmos ha de ser determinando de forma diferente. Además se muestra un estudio de los conjuntos de datos que se utilizaron en la fase de experimentación, se hace un estudio experimental con algunos modelos perezosos utilizados en la literatura para este tipo de datos y la nueva propuesta descrita en el capítulo 2 en presencia de datos simbólicos. Al finalizar se muestran los resultados de la experimentación con dos modelos,ambos teniendo en cuenta la dependencia entre las etiquetas y su comportamiento en presencia de datos ruidosos.

Capítulo 1

CLASIFICACIÓN MULTI-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 hablamos de clasificación multiclase ($|L|>2$). 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.

Existen numerosos problemas de creciente actualidad en los que el requisito de que cada instancia pertenezca a una y sólo una clase es difícil de cumplir. Para poder tratar adecuadamente este tipo de problemas surge el paradigma de aprendizaje denominado MLC.

La clasificación multi-etiqueta se caracteriza porque los conjuntos de clases no son excluyentes entre sí, y por tanto puede haber instancias a los que se les asocie más de una etiqueta a la vez. De esta manera, si se utilizan técnicas de clasificación multi-etiqueta en lugar de técnicas de clasificación clásicas. En MLC las etiquetas asociadas a cada instancia son binarias, que indican si dicha instancia está asociada o no a la etiqueta.

En las tablas 1.1 y se puede ver un conjunto de una sola etiqueta y multi-etiqueta respectivamente. De esta manera se observa que, mientras que el primero solo permite que una instancia este asociado a una sola clase, en el segundo cada instancia puede estar asociado a varias categorías simultáneamente.

La clasificación multi-etiqueta se enmarca dentro de un área mucho más amplia, el denominado Aprendizaje Multi-Etiqueta, (*Multi-Label Learning*, MLL). El aprendizaje multi-etiqueta abarca una gran variedad de tareas, entre las que podemos destacar, además de la clasificación multi-etiqueta, el ranking, la clasificación jerárquica multi-etiqueta, y clasificación multi-dimensional.

Ejemplo	Atributos	Etiqueta
1	X_1	Religión
2	X_2	Historia
3	X_3	Filosofía
4	X_4	Ciencia

Cuadro 1.1: Conjunto de una sola etiqueta

Ejemplo	Atributo	Etiquetas			
		Religión	Historia	Filosofía	Ciencia
1	X_1	T	F	F	T
2	X_2	T	T	F	T
3	X_3	F	F	T	T
4	X_4	T	F	F	F

Cuadro 1.2: Conjunto multi-etiqueta

1.1. NOTACIÓN

A continuación se define una notación para el problema de la clasificación multi-etiqueta. Esta notación trata de sintetizar la que se viene utilizando en la mayor parte de trabajos publicados. Para una formalización de la tarea de clasificación multi-etiqueta se considerará $L = \{\lambda_i : i = 1...n\}$ al conjunto de etiquetas presente en un problema determinado siendo n el número máximo de etiquetas que puede tener asociadas una instancia.

Por otro lado, se denominará al conjunto $D = \{D_j : j = 1...m\}$ conjunto de datos multi-etiqueta. Cada elemento del conjunto de datos D, también denominado instancia multi-etiqueta, estará compuesta por un par de elementos $D_j = (X_j, Y_j)$siendo X_j el vector de atributos condicionales, siendo $|X|$ su tamaño e $Y_j \subseteq L$ el subconjunto de etiquetas o atributos de decisión asociados a la instancia j, por tanto $|Y_j|$ será el número de etiquetas asociado a la instancia j.

MLC está asociado con el aprendizaje de un modelo predictivo que proporcione como salida una bipartición del conjunto de etiquetas en relevantes e irrelevantes respecto a una instancia de consulta.

Los problemas tradicionales de clasificación binaria y multi-clase pueden ser considerados como casos específicos de MLC, considerada esta última por su generalidad una tarea más compleja de resolver que las otras dos [11]. De esta manera se define la tarea de MLC como encontrar una función $h : X \rightarrow 2^L$ tal que h maximice o minimice un parámetro determinado.

1.2. MÉTODOS DESARROLLADOS PARA LA CLASIFICACIÓN MULTI-ETIQUETA

Los métodos 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 de problemas tratan de modificar los datos de entrada multi-etiqueta para convertirlos en datos de una sola etiqueta, a los que aplicar una técnica de clasificación clásica, y por otro lado, se han tratado de crear nuevas técnicas que puedan trabajar directamente con los datos multi-etiqueta [12].

1.2.1. Métodos de transformación de problemas

Estos métodos de transformación de problemas convierten un problema multi-etiqueta para obtener uno o varios conjuntos de datos de una sola etiqueta. Esta perspectiva puede tener algunas desventajas ya que en primer lugar, puede implicar una pérdida de información, bien en forma de etiquetas que se desprecian, o sencillamente por el hecho de que se pierdan las correlaciones entre etiquetas al generarse nuevos conjuntos de datos de una sola etiqueta.

La primera familia de estos métodos de transformación de problemas es basado en la "copia" o "selección"[13] [14]. La transformación por copia transforma todas las instancias de k etiquetas en k instancia de una sola etiqueta (cada etiqueta es un único elemento dentro de las k etiquetas). Un peso constante de $1/k$ también puede asignarse a cada uno de los casos copiados ("*copy-weight*"). Alternativamente, uno puede usar una estrategia de selección para reemplazar las múltiples etiquetas de cada instancia por una simple etiqueta que es la más frecuente ("*select-max*") o la menos frecuente ("*select-min*") o seleccionarlas aleatoriamente ("*select-random*").

Otra simple transformación es sólo usar los datos con una sola etiqueta ("*ignore*"). La segunda familia se le llama *Label Powerset*(LP)y sus variantes [15][16]. Los métodos LP tratan cada conjunto de etiquetas en el conjunto de entrenamiento como una nueva y simple etiqueta, y reconstruyen el conjunto de entrenamiento con la nueva definición de las etiquetas.

La tercera familia es basada en *Binary Relavance*(BR) [17] en este para cada etiqueta es entrenado un clasificador binario, es decir, se construye un nuevo conjunto de datos de etiqueta simple, donde cada ejemplo del conjunto original se fija como positivo si tiene asociado la etiqueta y se fija como negativo en caso contrario. Cuando se realiza la predicción, cada clasificador binario predice si su etiqueta asociada es relevante o no. Además puede llegar a ser poco práctica para determinados conjuntos de datos por el coste que supone su transformación. Como contrapartida, este enfoque permite utilizar técnicas de clasificación clásica sobre estos datos, más ampliamente probadas y aceptadas que sus equivalentes multi-etiqueta. Este método puede llegar a ser poco práctica para determinados conjuntos de datos por el coste que supone su transformación.

Como contrapartida, este enfoque permite utilizar técnicas de clasificación clásica sobre estos

datos, más ampliamente probadas y aceptadas que sus equivalentes multi-etiqueta.

1.2.2. Métodos de Adaptación de problemas

El segundo método es el de adaptación de algoritmos, ya que adaptan técnicas de clasificación clásica a la clasificación multi-etiqueta.

La mayor parte de las técnicas utilizadas en clasificación clásica han sido adaptadas para tratar con datos multi-etiqueta directamente. La principal diferencia entre los métodos de transformación de problemas y los de adaptación de algoritmos es que los primeros son independientes del paradigma de clasificación que se utilice posteriormente, pero los segundos no, aunque algunos métodos, internamente, modifiquen los datos de entrada de manera similar a alguna técnica de transformación.

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[18][19], Árboles de Decisión [20][21], Redes Neuronales [22][23], Clasificación Asociativa [24], Métodos Probabilísticos [25], Algoritmos Bioinspirados [26], y el k vecino más cercano .

1.3. ALGORITMOS PEREZOSOS EN DATOS MULTI-ETIQUETA

Los algoritmos perezosos son métodos basados en instancias que 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. De hecho, muchas técnicas construyen solo una aproximación local de la función de salida que se aplica en la vecindad de una nueva instancia y nunca construyen una aproximación diseñada para tener un buen rendimiento sobre todo el espacio de instancias de entrada.

Los algoritmos de aprendizaje perezosos se han utilizado con bastante éxito en este contexto de datos multi-etiqueta, entre los que podemos mencionar:

- El DML-KNN (*Dependent Multi-Label K-nearest neighbor*)[27] es una generalización del modelo anterior basado en el enfoque de clasificación de problemas multi-etiqueta donde la dependencia entre las etiquetas es considerada. Este usa un principio de *máximo a posteriori*

global ya que considera la frecuencia de todas las etiquetas en los k vecinos más cercanos de todas las instancias de entrenamiento, en lugar de considerar solamente la cantidad de vecinos de la etiqueta a ser asignada.

- El Mr.KNN (*Margin-Ratio k*NN)[28] es un novedoso enfoque de aprendizaje multi-etiqueta que integra los métodos de transformación y de adaptación de problemas. Este modelo introduce una nueva estrategia llamada relevancia débil, donde a cada instancia le es asignada una relevancia respecto a sus etiquetas asociadas. La relevancia débil está basada en la aplicación del algoritmo FCM (*fuzzy c-mean*s, algoritmo no supervisado). En otro paso esta relevancia es empleada en la función de votación usada por el clasificador *k*NN.

- Otro modelo es FSKNN(*Multi-label text categorization based on fuzzy similarity and k-nearest neighbors*) [29] que utiliza la medida de similitud borrosa y los k vecinos más cercanos para clasificación de textos multi-etiquetas. Esta medida de similitud borrosa se emplea para calcular la similitud entre un documento y cada categoría (etiquetas) de los documentos de entrenamiento y agrupar el conjunto de instancias de entrenamiento en clusters. Uno de los problemas asociados con el enfoque *k*NN es el costo computacional que demanda la búsqueda de los k vecinos más cercanos para todas las instancias de entrenamiento por lo que se emplea la similitud borrosa para hacer una preselección en la comparación. Este modelo utiliza la estimación de "*maximum a posteriori*".

- MLC-WkNN (*Multi-Label Weighted k -Nearest Neighbor Classifier with Adaptive Weight Estimation*)[30]: este método no introduce ningún otro parámetro a ajustar que la cantidad k de vecinos a considerar, a diferencia de IBLR-ML. Extiende la versión de *k*NN pesada (W*k*NN) para el contexto multietiqueta haciendo uso del teorema de Bayes. Una instancia de prueba es aproximada solamente mediante la suma pesada de los k vecinos más cercanos, lo cual resulta en un problema de programación cuadrática con una simple restricción para determinar los pesos.

1.3.1. ML*k*NN *(Multi-Label k-Nearest Neighbor)*

El modelo ML*k*NN es un algoritmo perezoso derivado del tradicional algoritmo *k*NN, este es uno de los más ampliamente utilizados en clasificación clásica. Este algoritmo determina si una instancia pertenece a una clase determinando por las k instancias más cercanas del conjunto de entrenamiento y examinando la clase a la que pertenecen estas.

Veremos algunas notaciones para entrar en detalles con el algoritmo: dada una instancia x y sus etiquetas asociadas $Y \subseteq \Upsilon$, tenemos que $\overrightarrow{y_x}$ es el vector categoría para x donde la l-ésima componente de $\overrightarrow{y_x}(l)$ $(l \in \Upsilon)$ toma el valor 1 si $l \in Y$ y 0 en otro caso. Denotaremos $N(x)$ 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:

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

Donde $\overrightarrow{C_x}(l)$ 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 $E_j^l(j \in \Upsilon)$ 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 $\overrightarrow{C_x}(l)$ el vector categoría $\overrightarrow{y_t}$ es determinado usando el siguiente principio "*maximum a posteriori*":

$$\overrightarrow{y_t}(l) = arg\,max_{b\varepsilon\{0,1\}} P(H_b^l \mid E^l_{\overrightarrow{C_t}(l)}), l\varepsilon\Upsilon \tag{1.1}$$

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

$$\overrightarrow{y_t}(l) = arg\,max_{b\varepsilon\{0,1\}} \frac{P(H_b^l)P\left(E^l_{\overrightarrow{C_t}(l)} \mid H_b^l\right)}{P(E^l_{\overrightarrow{C_t}(l)})} \tag{1.2}$$

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.

Este método de clasificación es conocido y estudiado en gran medida en la comunidad de reconocimiento de patrones, tanto por su sencillez y su rendimiento. La definición de la función de distancia es fundamental para obtener una buena precisión en un conjunto de datos dado y diferentes funciones de distancia se han propuesto para aumentar el rendimiento.

1.3.2. BRkNN(*Binary Relevance in conjunction with the kNN algorithm)*

El modelo BRkNN[17] 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 algortimo kNN, pero es $|L|$ más rápido. Se identifican dos extensiones de este modelo que mejoran su rendimiento de predicción general.

BR es uno de los más conocidos métodos de transformación de problemas para datos multi-etiqueta. Este aprende un clasificador binario $h_\lambda : X \longrightarrow \{\neg\lambda, \lambda\}$ para cada etiqueta diferente $\lambda\varepsilon L$. BR transforma el conjunto de datos originales en $|L|$ conjunto de datos D_λ que contienen todos ejemplos del conjunto de datos original, etiquetanddo 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 BR*k*NN pudieramos haber utilizado implementaciones existentes de BR y *k*NN. Sin embargo, el problema de emparejar BR con *k*NN 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 redundantes, BR*k*NN extiende el algoritmo *k*NN para que las predicciones independientes se hagan para cada etiqueta, después de una sola búsqueda de los k vecinos más cercanos. De esta manera BR*k*NN es $|L|$ veces más rápido que BR más *k*NN 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, suponemos que Y_j, j=l...k es el conjunto de etiquetas de los k vecinos más cercanos de una nueva instancia x. La confidencia c_λ de una etiqueta $\lambda \varepsilon L$ es igual a:

$$c_\lambda = \frac{1}{k}\sum_{j=1}^{k} I_{Y_j}(\lambda) \tag{1.3}$$

donde $I_{Y_j} : L \longrightarrow \{0,1\}$ 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.

1.3.3. IBLR-ML(*Combining instance-based learning and logistic regression*)

El modelo IBLR-ML [31] 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 instancia vecinas a una instancia de consulta como una característica de esa instancia, 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.

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 dependencia 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.

1.4. FUNCIONES DE DISTANCIA

Las funciones de distancia se clasifican en dependencia del tipo de dato que puedan manejar entre las que podemos ver están las que trabajan con datos ordinales (continuos o discretos) de entrada, los datos nominal (o simbólicos) y datos heterogéneos que consisten tanto en datos ordinal y nominales. Ahora veremos dos de las más utilizadas en la literatura.

1.4.1. HEOM (*Heterogeneous Euclidean-Overlap Metric*)

Una forma de manejar las aplicaciones con atributos tanto continuos como nominales es utilizar una función de distancia heterogénea que utiliza diferentes funciones de distancia en diferentes tipos de atributos. Pn algunos casos encontramos que estos atributos son mezclados por lo que se necesitan métricas más complejas para manipular estos datos heterogéneos.

La distancia HEOM se utiliza para calcular la distancia entre dos instancias, esta utiliza la métrica *overlap* para atributos simbólicos y la distancia Euclideana normalizada para atributos numéricos. Dados dos instancias con datos heterogéneos puede ser calculada como sigue:

$$D_{HEOM}(x,y) = \sqrt{\sum_{i=0}^{m} \delta(x_i,y_i)^2} \tag{1.4}$$

Aquí hace referencia a cada atributo y $\delta(x_i,y_i)$ es definido como:

$$\delta(x_i,y_i) = \begin{cases} 1 & si\, x_i = ?\, o\, y_i = ? \\ d_{overlap}(x_i,y_i) & si\, el\, atributo\, es\, símbolico \\ d_{nom1}(x_i,y_i) & si\, el\, atributo\, es\, numérico \end{cases} \tag{1.5}$$

Para atributos ausentes la función de distancia retorna su máximo valor. La $d_{overlap}$ se refiere a atributos simbólico donde no se puede definir un orden entre los valores asociados a sus categorías ontológicas e incluso a sus códigos formales.

$$d_{overlap}(x_i,y_i) = \begin{cases} 0 & si\, x_i = y_i \\ 1 & si\, e.o.c \end{cases} \tag{1.6}$$

La expresión 1.7 se refiere a la distancia Euclideana normalizada (local) y solamente puede ser utilizada si el atributo es numérico, donde l y u se refieren a los valores mínimo y máximo del dominio del atributo respectivamente.

$$d_{nom1}(x_i,y_i) = \frac{|x_i - y_i|}{l-u} \tag{1.7}$$

En este trabajo se trabaja con la distancia *overlap* ya que los conjuntos de datos que se utilizaron en la experimentación solo tienen presencia de datos símbolicos. La función de

distancia HEOM elimina los efectos del ordenamiento arbitrario de los valores nominales, pero su enfoque es demasiado simplista para el manejo de atributos nominales ya que no hace uso de la información adicional proporcionada por los valores de atributos nominales que pueden ayudar en la generalización.

1.4.2. VDM (*Value Difference Metric*)

Recientemente, en el contexto del aprendizaje basado en instancias muchas propuestas se basan en propiedades estadísticas pre-computadas a partir de datos de entrenamiento disponible y puede ser interpretada en términos de probabilidades. La distancia más conocida de este tipo fue propuesta por [7], denominada VDM.

La función de distancia VDM se define en términos de valores de los atributos que están condicionados por las probabilidades posteriores de la clase. Esta función se basa en la equivalencia $A = B \Longleftrightarrow (\forall D)\,(A \Longrightarrow D) \Longleftrightarrow (B \Longrightarrow D)$ donde D denota un valor de un atributo de decisión. Es decir, dos valores A y B de un atributo de condición son iguales si ambas conducen a la misma decisión.

La expresión 1.8 muestra el criterio local de la distancia VDM para comparar dos valores A y B de un atributo en el contexto de MLC:

$$D_{VDM}(A,B) = \sqrt{\sum_{l=1}^{m}(P(\lambda_l/A) - P(\lambda_l/B))} \tag{1.8}$$

En la ecuación 1.8 la $P(\lambda \mid A)$cuantifica el grado en que se puede obtener la etiqueta λ conocido A. Estas probabilidades pueden ser estimadas desde el conjunto de entrenamiento empleando la ecuación 1.9:

$$P(\lambda \mid A) = \frac{\sum_i N_{x,\lambda,A}}{\sum_i N_{x,A}} \tag{1.9}$$

donde:

$N_{x,\lambda,A}$ es el número de casos del conjunto de entrenamiento que tienen el valor de etiqueta λ y A como atributo;

$N_{x,A}$ es el número de casos del conjunto de entrenamiento que tienen el valor A como atributo.

A diferencia de la distancia HEOM, la distancia VDM sólo es usada para comparar atributos simbólicos.

Nótese en su criterio de comparación local, que VDM tiene en cuenta la relación de cada rasgo con el rasgo objetivo; la cual se referencia en [8] como uno de los criterios de similitud más sofisticados para comparar atributos discretos.

Un estudio experimental [32] utilizando la regla de los k vecinos más cercanos y la variante no pesada (todos los rasgos influyen de igual manera en la similitud global) muestra porcientos de

clasificaciones correctas para VDM al menos tan buenos como con el C5.0 mientras que en [33] se constata que VDM reduce el impacto de atributos irrelevantes sobre la precisión de la clasificación sin necesidad de pre-procesamiento de los datos.

El criterio de comparación local de VDM tiene las ventajas siguientes: tiene como imagen un valor normalizado en el intervalo [0,1]; mientras la mayoría de las funciones de distancia sobre valores discretos requieren valores binarios o simplemente cuentan el número de no coincidencias, el resultado de VDM depende del valor individual que tome el rasgo (es más informada), y el valor ausente se trata como si éste fuese otra categoría de las posibles. Sin embargo, esta función solo es aplicable a problemas de aprendizaje supervisado con atributos simbólicos.

1.5. Aprendizaje Basado en Instancias(*Instance-Based Learning*, IBL)

Los IBL pertenecen a la familia de algoritmos perezosos estos utilizan directamente ejemplos del dominio de aplicación en la construcción del modelo, y tienen pocos parámetros a definir en el proceso de ingeniería del conocimiento con respecto a otros modelos. Sin embargo, el éxito de su aplicación en un dominio particular depende en buena medida del criterio de similitud a emplear, cuya formalización es una tarea difícil, dependiente del contexto, y para la cual sería conveniente aplicar también aprendizaje automático.

El modelo ConFuCiuS se concibe como una variante del *k*NN utilizando como criterio de comparación a nivel de rasgo una variante de VDM.

A continuación se describe este modelo atendiendo a cada uno de los aspectos que caracteriza el aprendizaje basado en instancias:

- Se tienen almacenadas instancias de entrenamiento que se representan como pares de atributo-valor.
- El valor para la clase del problema a resolver se corresponde con la clase que más aparece (más probable) en los *k* casos más similares recuperados (*majority vote*) [8], luego de calcular la probabilidad de cada clase atendiendo a las siguientes expresiones:

$$z = ArgMax_{l=1}^{c} P_r(C_l/q) \tag{1.10}$$

$$Pr^k(C/q) = \frac{\sum_{i \varepsilon Kq} C(z_i)}{|K_q|} \tag{1.11}$$

donde Kq es el conjunto de los k vecinos más cercanos, subconjunto del conjunto de instancias de entrenamiento; y que se seleccionan como resultado de comparar cada instancia de entrenamiento x con la solicitud q.

- Para comparar dos instancias (en el espacio m-dimensional) se utiliza la distancia Euclideana que emplea el k -NN estándar.

Capítulo 2

Descripción del modelo propuesto

El algoritmo de clasificación basado en el k vecino más cercano para datos multi-etiqueta MLkNN se deriva del tradicional algoritmo kNN, este utiliza el principio "*maximum a posteriori*" para determinar el conjunto de etiquetas de una instancia de prueba, basado en las probabilidades a priori y posteriori para el cálculo de la frecuencia de cada etiqueta en los k vecinos más cercanos. Su capacidad de generalización depende en gran medida de la definición de la función de distancia que se utiliza para comparar la instancia de prueba y las instancias entrenadas.

Este algoritmo utiliza HEOM como función de distancia, esta solo considera la diferencia superficial entre los objetos comparados, mientras que VDM reduce el impacto de atributos irrelevantes en la precisión de la clasificación sin necesidad del pre-procesamiento de los datos.

2.1. GML*k*NN (*Goal-related distance measure Multi-label k*NN)

Se puede pensar en un primer modelo llamado *GMLkNN*(*Goal-related distance measure and k*NN) el cual eliminaría las deficiencias del ML*k*NN al usar la distancia HEOM en la búsqueda de los vecinos. Este nuevo modelo tiene en cuenta la relación existente entre el rasgo y el objetivo ya que utiliza la función de distancia VDM a la hora de seleccionar los k vecinos más cercanos. GML*k*NN primero calcula las probabilidades priori y posteriori las cuales son directamente estimadas del conjunto de entrenamiento basados en el contador de frecuencia. Después estos valores son usados para cada instancia de prueba t:

1. Se identifican sus kNNs N_t en el conjunto de entrenamiento
2. Determinar el vector categoría $\overrightarrow{y_t}$

Este modelo es derivado del tradicional algoritmo kNN, usando en el paso 1 una distancia relaciona con el objetivo para comparar la instancia de prueba y la de entrenamiento, la cual es la variante de VDM para datos multi-etiquetas que fue previamente introducida. Este algoritmo pertenece al grupo de los métodos adaptativos. Notar que la asignación de la etiqueta λ a la instancia de prueba

t, acorde a 1.1 es dependiendo del evento $E^{\lambda}_{\overrightarrow{C}_t}(l)$. En otras palabras el MAP solo toma en cuenta el número de vecinos de la etiqueta a ser asignada en lugar de todas las etiquetas en el conjunto N_t.

Se espera que el algortirno GML*k*NN muestre mejores resultados que el ML*k*NN ya que el propuesto usa una distancia relacionada con el objetivo profunda basada en el conocimiento en constraste con la previa que solo considera una similitud superficial. No obstante, este método puede mejorarse si se toma en cuenta la información que una etiqueta proporciona sobre otra.

2.2. GMLkNN teniendo en cuenta la dependencia entre las etiquetas

En el modelo anterior no se tiene en cuenta una información importante como es la dependencia entre las etiquetas por lo que se redefine el GML*k*NN para no perder esa valiosa información .

En este nuevo modelo lo que se propone redefinir el procedimiento para determinar el vector de categoría $\overrightarrow{y_t}$ en aras de obtener una mayor ganancia de información.

Ahora se deben obtener soluciones más exactas, desde que una instancia pertenezca a una clase con un cierto grado en el sentido de la teoría de conjuntos borrosos(un mapa borroso del espacio de salida) permitiendo adoptar aproximación difusa para estimar una función de puntuación basado en *k*NN y sus valores de distancia para poner a prueba la instancia *t*. Vale la pena mencionar que las ideas similares han sido explotados en [29] en clasificación de documentos.

2.2.1. Usando relación borrosa para estimar la función de puntuación para una instancia de prueba

Dado que el objetivo en problemas MLC es predecir un conjunto de etiqueta, puede ser útil medir la distribución de una etiqueta en un conjunto de ellas. Sólo se está teniendo en cuenta la instancia de entrenamiento x cuyo conjunto de etiqueta L_x incluye la etiqueta especificada. Siguiendo nuevamente el enfoque bayesiano definimos:

Suponemos que λ es una etiqueta y que L un conjunto de ellas, la distribución de una etiqueta λ en un conjunto L es definida como $dist : \mathscr{L} \times 2^{\mathscr{L}} \rightarrow [0,1]$ para cuantificar el grado en que la etiqueta λ se distribuye en todo el conjunto de etiquetas en un conjunto de instancias de entrenamiento, que se puede estimar según:

$$dist(\lambda, L) = P(L/\lambda) \tag{2.1}$$

$$dist(\lambda, L) = \frac{P(L)P(\lambda/L)}{P(\lambda)} \tag{2.2}$$

$$dist(\lambda, L) = P(L/\lambda)\frac{\sum_{i=1}^{m} L(x_i)\lambda(x_i)}{\sum_{i=1}^{m} \lambda(x_i)} \tag{2.3}$$

Observando que si la etiqueta λ tiene lugar en la instancia de entrenamiento x donde $L_x = L$ el valor de $dist(\lambda, L)$ será 1.0. Por el contrario, la penalización será dada a una etiqueta que ocurre de manera eventual en el conjunto de etiquetas de todas las instancias, y en este caso el valor de $dist(\lambda, L)$ será muy bajo.

La relevancia de una etiqueta es, en efecto gradual en el sentido de la lógica difusa y no incierta en el sentido de la teoría de la probabilidad. Generalizando el ajuste anterior de la clasificación muti-etiqueta, se asume que cada instancia $x \in \chi$ puede pertenecer a cada clase λ con un cierto grado definido en el mapa difuso del espacio de salida. En otras palabras, el conjunto L_x de etiquetas relevantes es ahora un subconjunto borroso de $\mathscr{L}$. No vamos a distinguir entre el conjunto borroso L_x y sus funciones miembro y denotamos a $L_x(\lambda)$ el grado de pertenencia de la etiqueta λ en el conjunto borroso L_x. Así, en lugar de utilizar un subconjunto de atributos m representado en el vector como presencia (1) / ausencia (0), usamos como atributo de decisión un valor de grado membresía $L_x(\lambda_j) \in [0,1]$, que se calcula para cada instancia x basada en la instancia de entrenamiento x_i usando:

$$L_x(\lambda) = \frac{dist(\lambda, L)}{max_i dist(\lambda, L_{x_i})},\ i = 1..m \tag{2.4}$$

Tenga en cuenta que, el valor de $dist(\lambda, L)$ es utilizado, la información global de las probabilidades a posterior del conjunto de entrenamiento se debe estimar, pero sólo una vez. Es decir, para cada etiqueta $\lambda \in \{\lambda_1, \lambda_2, ..., \lambda_n\}$ y $L \in \{L_{x_1}, L_{x_2}, ..., L_{x_n}\}$. Por lo tanto, $L(\lambda)$es básicamente la distribución de la etiqueta λ sobre todos los conjuntos de etiquetas posibles presentadas en el conjunto de entrenamiento.

Proponemos para estimar $f(t, \lambda)$ para la instancia de prueba t y clase λ basándose en sus kNNs y relaciones borrosas el uso de:

$$f(t, \lambda) = w_1 \otimes L_{x_1}(\lambda) \oplus w_2 \otimes L_{x_2}(\lambda) \oplus .. \oplus w_k \otimes L_{x_{/N_t/}}(\lambda) \tag{2.5}$$

donde:

- N_t es el conjunto de instancias del conjunto de entrenamiento seleccionadas como kNN para la instancia de prueba t.
- $L_x(\lambda)$ es el grado de membresía de la etiqueta λ en el conjunto borroso L_x.
- $\otimes$ y $\oplus$ son los operadores de agregación definidos en las relaciones borrosas
- w_k es el peso del kth vecino calculado considerando $d(x_k, t)$ como sigue:

$$w_k = \frac{1}{d(x_k, t)} \tag{2.6}$$

Notar que $f(t, \lambda_l)$ es una aproximación de que la etiqueta l sea relevante para la instancia t considerando adicionalemte la información local proveniente de sus kNNs. De hecho, se utilizan los prototipos borrosos, donde la membresía ofrece un nivel útil de la confianza de la clasificación[34][et al. (1985)].

El principio implícito del método borrosos kNN expresado por Dubois et al. (1997) como: "cuanto más similares son la descripción de los atributos del problema, más similares son los atributos resultantes". Para relajar las restricciones, Dubois et al. (1997) reformuló el principio un poco y utilizó conjuntos difusos para modificar la restricción: "Cuanto más similares sea s1 y s2, mas similares serán t1 y t2".

2.2.2. Considerando dependencia entre las etiquetas

En GMLkNN se tiene presente la depencia existente entre las etiquetas la cual se logra como: dado una instancia de entrenamiento, primero calcula las probabilidades priori y posteriori, que pueden todas ser directamente estimadas por el conjunto de entrenamiento basada en la función de puntuación y estos valores son usados para cada instancia de prueba t.

Se detalla a continuación el procedimiento:

1. Se identifican sus kNNs N_t en el conjunto de entrenamiento
2. Para cada $x \in N_t$ (vecinos de t) hacer
   a) Calcular los pesos w_x usando la expresión 2.6
   b) Calcular $L_x(\lambda_l)$ para cada etiqueta λ_l, $l = 1..n$ usando 2.4
3. Estimando el umbral $f(t, \lambda_l)$ para cada etiqueta λ_l, $l = 1..m$ basado en sus kNNs donde el producto y el máximo como operadores de agregación son usados, la función $f(.)$ para usar es:

$$f(t, \lambda) = Max_{x \in N_t}(w_x \cdot L_x(\lambda)) \tag{2.7}$$

Note que:

$$kNN(t) = max_l P(\lambda_l / t),\ l = 1..m$$

kNN difiere de otros clasificadores en cómo define estas clases probabilidades posteriores:

$$P(\lambda / t) = \frac{\sum_{x \in N_t} \lambda(x) \cdot K(d(x,t))}{\sum_k d(x_k, t)}$$

estandar kNN usa $\lambda = \arg\max_{l=1}^{m} \vec{C_t}(l)$:

$$P(\lambda/t) = \frac{\sum_{x \in N_t} 1\left[\lambda\left(x\right) = 1\right]}{/N_t/}$$

$$K(.) = 1$$

Se propone usar :

$$P(\lambda/t) = \frac{\sum_{x \in N_t} K\left(d(x,t), L_x(\lambda)\right)}{\sum_{x \in N_t} d(x,t)}$$

$$K\left(d(x,t), L_x(\lambda)\right) = \frac{L_x(\lambda)}{d(x,t)}$$

$$f(t,\lambda) = max_k(w_k \cdot L_{x_k}(\lambda))$$

En general, esto es:

$$K(d(x,t), L_x(\lambda), \oplus, \otimes) = w_1 \otimes L_{x_1}(\lambda) \oplus w_2 \otimes L_{x_2}(\lambda) \oplus .. \oplus w_k \otimes L_{x_{/N_t/}}(\lambda)$$

4. Determinar el vector categoría $\vec{y_t}$ utilizando el umbral:

$$\vec{y_t}\left(l\right) = \begin{cases} 1 & f(t,\lambda) \geq 0{,}5 \end{cases} \tag{2.8}$$

La idea principal, en constraste con el procedimiento que sigue 1.1, es considerar la etiqueta de las instancias vecinas como "rasgos" (prototipos). Más, el valor de puede ser visto como una medida de la relevancia de λ tomando en cuenta su relación con los otros del conjunto de entrenamiento, desde el punto de vista del *k-th* vecino. Por último, una agregación ponderada utilizando los pessos w_k, se hace para cada etiqueta. Tener en cuenta que el valor de $f(t,\lambda)$ puede considerarse naturalmente como una puntuación para la etiqueta λ. Por otra parte, una predicción multi-etiqueta pura para la etiqueta λ se deriva de $f(t,\lambda_l)\, l = 1..m$ a través del umbral α =0.5.

2.3. Implementación en MULAN

MULAN es una biblioteca de código abierto que trabaja con conjuntos de datos multi-etiqueta los cuales constan de ejemplos de entrenamiento de una función objetivo que tiene varias variables de destino binarios. Esto significa que cada elemento de un conjunto de datos multi-etiqueta puede ser miembro de varias categorías o anotado por muchas etiquetas (clases). Esto es en realidad la naturaleza de muchos problemas del mundo real, tales como la anotación semántica de imágenes y de vídeo, página web categorización, el marketing directo, la genómica funcional y la categorización de la música en géneros y emociones. Una introducción a la minería de datos multi-etiqueta se proporciona en [35]

Actualmente, la biblioteca incluye una variedad de algoritmos del estado del arte para realizar las siguientes tareas principales de aprendizaje multi-etiqueta:

- Clasificación. Esta tarea se refiere a la salida de una bipartición de las etiquetas en los más relevantes e irrelevantes para una instancia determinada entrada.
- Ranking. Esta tarea se refiere a la salida de una ordenación de las etiquetas, de acuerdo con su importancia para un elemento de datos dado.
- Clasificación y ranking. Una combinación de las dos tareas mencionadas anteriormente.

Además, la biblioteca ofrece las siguientes características:

- La selección de características. Métodos de referencia simples son soportados actualmente.
- Evaluación. Las clases que calculan una gran variedad de medidas de evaluación a través de la evaluación de retención y validación cruzada.

Como ya se mencionó, MULAN es una biblioteca. Como tal, ofrece sólo API programático a los usuarios de la biblioteca. No hay ninguna interfaz gráfica de usuario (GUI) disponibles. La posibilidad de utilizar la biblioteca a través de línea de comandos, es también en la actualidad no es compatible.

En esta biblioteca fue incorporado nuestro modelo GML*k*NN descrito en la sección 2.1, para ello creamos una nueva clase en el paquete *mulan.classifier.lazy* que entienda de la clase *MultiLabelKNN* la cual llama a la función de distancia VDM incorporada al WEKA[1], se puede observar en 2.1.

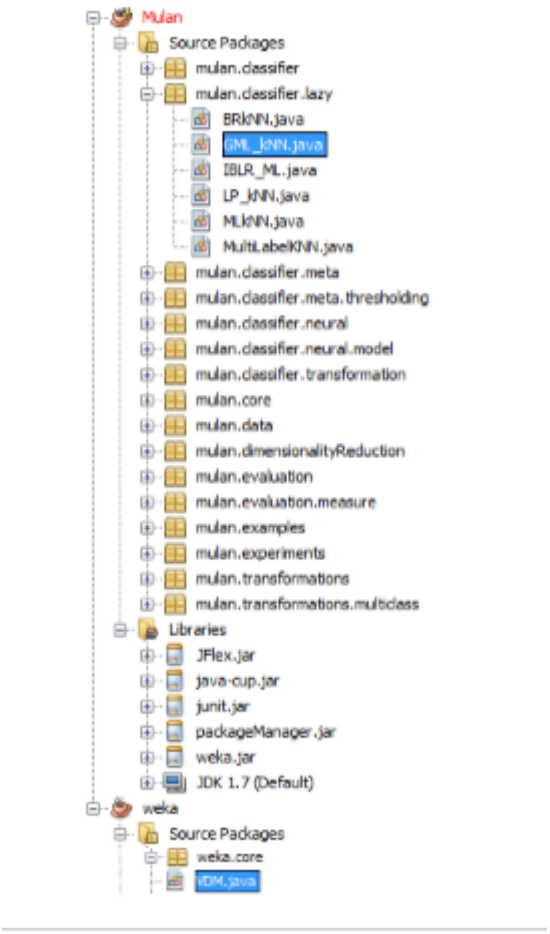

Figura 2.1: Incorporación de GMLkNN al MULAN y VDM en el WEKA.

2.4. Análisis de la complejidad computacional

En GMLkNN en su primera versión en la fase de entrenamiento busca las probabilidades a priori de cada una de las instancias como lo hace MLkNN, como se muestra en el algoritmo 2.1. En este caso tiene una complejidad computacional de$O(nm)$, donde n es el número de etiquetas y m es número de instancias.

Algoritmo 2.1 Probabilidades a priori GMLkNN variante 1

for $l \in L$ **do**
$P(H_1^l) = (s + \sum_{i=1}^{m} \vec{y}_{x_i}(l))/(s \times 2 + m); P(H_0^l) = 1 - P(H_1^l);$
end for

Las probabilidades a posteriori son calculadas como se muestra en el algoritmo 2.2 y tiene una complejidad computacional de $O(nkm)$, donde n es la cardinalidad del conjunto de etiquetas, m es el número de instancias y k es el número de vecinos. Para el cálculo de los k vecinos más cercanos se emplea la ecuación 1.8 .

Algoritmo 2.2 Probabilidades a posteriori GMLkNN variante 1

Identificar $N(x_i)$, $i \in \{1,2,\ldots,m\}$ utilizando VDM
for $l \in L$ **do**
  for $j \in \{0,1,\ldots,k\}$ **do**
    $c[j] = 0; c^{'}[j] = 0;$
  end for
  for $i \in \{1,2,\ldots,m\}$ **do**
    $\delta = \vec{C}_{x_i}(l) = \sum_{a \in N(x_i)} \vec{y}_a(l);$
    if $(\vec{y}_{x_i}(l) == 1)$ **then**
      $c[\delta] = c[\delta] + 1;$
    else
      $c^{'}[\delta] = c^{'}[\delta] + 1;$
    end if
  end for
  for $j \in \{0,1,\ldots,k\}$ **do**
    $P(E_j^l|H_1^l) = (s + c[j])/(s \times (k+1) + \sum_{p=0}^{k} c[p]);$
    $P(E_j^l|H_0^l) = (s + c^{'}[j])/(s \times (k+1) + \sum_{p=0}^{k} c^{'}[p]);$
  end for
end for

En la fase de predicción se emplean las probabilidades a priori y posteriori precalculadas, para ello se determina el conjunto de etiquetas asociadas a la instancia de consulta basado en las probabilidades estimadas, como se muestra en el algoritmo 2.3. Este tiene una complejidad computacional de $O(max(m,nk))$ para cada instancia.

Algoritmo 2.3 Fase de predicción GMLkNN variante 1

Identificar $N(t)$ utilizando VDM
for $l \in L$ **do**
  $\vec{C}_t(l) = \sum_{a \in N(t)} \vec{y}_a l;$
  $\vec{y}_t(l) = argmax_{b \in \{0,1\}} P(H_b^l) P(E^l_{\vec{C}_t(l)}|H_b^l);$
  $\vec{r}_t(l) = P(H_1^l|E^l_{\vec{C}_t(l)}) = (P(H_1^l)P(E^l_{\vec{C}_t(l)}|H_1^l))/P(E^l_{\vec{C}_t(l)})$
    $= (P(H_1^l)P(E^l_{\vec{C}_t(l)}|H_1^l))/(\sum_{b \in \{0,1\}} P(H_b^1)P(E^l_{\vec{C}_t(l)}|H_b^l))$
end for

El algoritmo GMLkNN en su variante 1 tiene la misma complejidad computacional que el algoritmo MLkNN, donde la diferencia entre ambos algoritmos radica en el cálculo de los k vecinos más cercanos donde MLkNN utiliza 1.4 y GMLkNN los calcula utilizando 1.8, la complejidad general del algoritmo es $O(nkm)$.

Por otra parte la variante 2 de GMLkNN en la fase de entrenamiento para cada una de las instancias busca en el conjunto de entenamiento el número de instancias que tienen la combinación de este conjunto de etiquetas en la clasificación y además la presencia de estas, es decir , la distribución de una etiqueta en un conjunto de ellas, como se muestra en el algoritmo 2.4.

Algoritmo 2.4 Fase de entrenamiento GML-kNN variante 2

for $i \in \{0,1,\ldots,n\}$ **do**
  for $j \in \{1,2,\ldots,m\}$ **do**
    if $(x_j[i] == 1)$ **then**
      $c[i]++;$
    end if
  end for
end for
for $i \in \{1,2,\ldots,m\}$ **do**
  for $l \in L$ **do**
    Calcular la distribución $dist(l, L_{x_i})$
  end for
end for

En esta fase de entrenamiento el algoritmo tiene una complejidad computacional de $O(m^2n)$. A diferencia de la primera variante del GMLkNN y el MLkNN este algoritmo tiene una mayor complejidad computacional debido a que el número de instancias siempre es mayor que el número de vecinos seleccionados.

En la fase de predicción esta segunda variante calcula las distribuciones de una etiqueta en un conjunto de ellas empleando la expresión 2.4, como se muestra en el algoritmo 2.5. Esta fase tiene una complejidad computacional de $O(knm)$. En comparación con la variante 1 de GMLkNN y MLkNN es más compleja.

Algoritmo 2.5 Fase de predicción GMLkNN variante 2

Identificar $N(t)$ utilizando VDM
for $x \in N(t)$ **do**
  $w_x = 1/d(x,t);$
  Calcular $L_x(\lambda)$
end for
$f(t,\lambda) = max_k(w_k \cdot L_{x_k}(\lambda));$

La complejidad de esta variante de GMLkNN es $O(m^2n)$, la cual es mayor a la variante 1 de este modelo y MLkNN.

Capítulo 3

Validación del modelo propuesto

En este capítulo se muestran los experimentos realizados con los algoritmos perezosos ML*k*NN, BR*k*NN, IBLR-ML y GLM*k*NN utilizando la biblioteca de código abierto en java MULAN utilizando algunos conjuntos de datos multi-etiqueta más utilizados en la literatura.

3.1. Conjuntos de Datos Multi-Etiqueta

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 de datos *medical* [36] es un conjunto de datos desarrollado para una competición internacional, el 2007 *International Challenge: Classifying Clinical Free Text Using Natural Language Processin*g, 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[37]* 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* [38]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* [39]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.

EL conjunto *delicious* [40]contiene datos textuales de las páginas web, junto con sus *tag*, y se utiliza para entrenar un clasificador multietiqueta para la sugerencia de *tag* automatizado.

El conjunto *corel5K* [41]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. Los datos 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. Se entrenan utilizando 4.500 imágenes corel.

El conjunto *Corel 10K(10 samples)* contiene los datos utilizados para el artículo JMLR "*Matching Words and Pictures*", por Kobus Barnard, Pinar Duygulu, Nando de Freitas, David Forsyth, DavidBlei, y Michael I. Jordan, en *Journal of Machine Learning Research*. Los datos está muy improvisado y tiene algunas anomalías. Cada segmento de la imagen está representado por 46 características. Dado que cada imagen tiene un número diferente de segmentos, que indique el número de segmentos utilizados en archivos separados. Los segmentos para una determinada imagen se enumeran en orden descendente de tamaño y están separados de los de la imagen siguiente de varios espacios. Para el artículo JMLR se utilizaron los 10 segmentos más grandes.

3.2. MÉTRICAS PARA LA DESCRIPCIÓN DE LOS CONJUNTOS DE DATOS

A la hora de plantear un desarrollo experimental sobre un modelo que genere clasificadores multi-etiqueta, así como de discutir los resultados obtenidos por éste, es importante determinar cómo de multi-etiqueta es el conjunto de datos con el que se valida. Para ello se han propuesto varias métricas.

La métrica más sencilla es el denominado *distinct* [42], el cual es el número de combinaciones de distintas etiquetas presente en un conjunto de datos. Además del número de combinaciones, se han propuesto otras dos medidas, la cardinalidad y la densidad [42]. Si consideramos $|Y_i|$ el número de etiquetas de la instancia i, la cardinalidad de un conjunto de datos D viene dada por el promedio de etiquetas asociadas a cada instancia.

$$cardinalidad(D) = \frac{1}{m}\sum_{i=0}^{m}|Y_i| \tag{3.1}$$

Sin embargo intuitivamente parece razonable considerar también el número total de etiquetas

posible. Por ejemplo podemos tener dos conjuntos de cardinalidades similares, pero puede ocurrir que el número de posibles etiquetas máximo que pueda tener asignado una instancia varíe enormemente. Para evitar la distorsión que supone comparar cardinalidades entre conjuntos con diversos número de etiquetas se define la densidad como la cardinalidad entre el número total de etiquetas $|L|$.

$$densidad\,(D) = \frac{1}{m}\sum_{i=1}^{m}\frac{|Y_i|}{|L|} \tag{3.2}$$

3.2.1. MÉTRICAS PARA DATOS MULTI-ETIQUETA

El hecho de que la clasificación multi-etiqueta tenga características propias hace que sea necesario definir métricas específicas para determinar el rendimiento de los clasificadores multi-etiqueta. Se han propuesto diversas formas de determinar el rendimiento de un clasificador multi-etiqueta. Básicamente las métricas que existen se agrupan en métricas basadas en ejemplos (*example-based*) y métricas basadas en etiquetas (*label-based*)[43]. Las métricas basadas en etiquetas son métricas de clasificación binaria clásicas y se calculan para cada etiqueta, siendo posteriormente promediadas para obtener un valor único. Por contra las métricas basadas en ejemplos están específicamente concebidas para problemas multi-etiqueta, y por esta razón se calculan para cada ejemplo teniendo en cuenta todo el conjunto de etiquetas predicho y real.

En esta sección se definen las principales métricas basadas en ejemplos, que se pueden utilizar para medir el rendimiento de un algoritmo de aprendizaje multi-etiqueta. Las métricas basadas en ejemplos toman como entrada la respuesta completa del clasificador para cada instancia. Posteriormente se promedia el valor de la métrica para obtener el valor medio de esta sobre todo el conjunto de entrenamiento.

Dentro de las métricas basadas en ejemplos encontramos las denominadas métricas binarias, que son utilizadas para medir cualquier algoritmo de clasificación multi-etiqueta y las llamadas métricas de ranking que se utilizan en el contexto de algoritmos de ranking.

3.2.1.1. MÉTRICAS BINARIAS

La métrica más sencilla de calcular es la denominada subset accuracy, la cual calcula el porcentaje de ejemplos que el clasificador ha acertado en todas sus etiquetas asociadas. Esta métrica tiene la característica de que es muy estricta, ya que en el momento en que el conjunto real y el predicho no sean exactamente iguales, la instancia no puntúa en absoluto. Por esta razón no suele ser utilizada para evaluar clasificadores multi-etiqueta.

$$subset\,accuracy = \frac{1}{m}\sum_{i=1}^{m}\delta(P_i = Y_i) \tag{3.3}$$

Siendo

$$\delta(x) = \begin{cases} 1 & six = verdadero \\ 0 & si = falso \end{cases} \tag{3.4}$$

Una métrica basada en ejemplos muy usada para evaluar clasificadores multi-etiqueta es la denominada pérdida de *Hamming* (*Hamming loss*). Esta métrica considera tanto los errores de clasificación (el hecho que una etiqueta incorrecta sea predicha) como los errores por omisión (cuando una etiqueta que deberá estar presente en la salida del clasificador no lo está). La *Hamming loss* es una medida de la diferencia entre el conjunto de etiquetas predicha y el conjunto de etiquetas real, siendo por tanto más cercana a 0 cuanto mejor sea el resultado del clasificador. En la ecuación 1.3 aparece la expresión de la métrica, siendo $|P_i \Delta Y_i|$ la diferencia simétrica entre los dos conjuntos de etiquetas predichas y reales.

$$Hamming\,loss = \frac{1}{m}\sum_{i=1}^{m}\frac{|P_i \triangle Y_i|}{n} \tag{3.5}$$

Godbole y Sarawagi, en [44], han redefinido las métricas *accuracy*, la *precision* y el *recall*, para que tuvieran en cuenta la respuesta completa del clasificador, y por tanto han de ser consideradas como métricas basadas en ejemplos. Estos autores definen la *precision*3.6 como el promedio de etiquetas acertadas frente a las predichas, el *recall3.7* como el promedio de etiquetas acertadas frente a las reales y la *accuracy* 3.12 como el promedio de la fracción de aciertos del clasificador frente a las unión de etiquetas reales y predichas.

$$precision = \frac{1}{m}\sum_{i=1}^{m}\frac{|Y_i \cap P_i|}{|P_i|} \tag{3.6}$$

$$recall = \frac{1}{m}\sum_{i=1}^{m}\frac{|Y_i \cap P_i|}{|Y_i|} \tag{3.7}$$

$$accuracy = \frac{1}{m}\sum_{i=1}^{m}\frac{|Y_i \cap P_i|}{|Y_i \cup P_i|} \tag{3.8}$$

Otra forma, basada en ejemplos, de evaluar el rendimiento de un clasificador multi-etiqueta es la α-*evaluation*, propuesta por Boutell et al.[45]. En esta evaluación se puntúa cada predicción realizada por el clasificador según la ecuación 3.9 .

$$score(P_i) = \left(\frac{|Y_i \cap P_i|}{|Y_i \cup P_i|}\right)^{\alpha} (\alpha \geq 0) \tag{3.9}$$

Al parámetro α la ecuación se denomina forgiveness rate porque refleja el modo en el que la métrica penaliza errores en la clasificación. Un pequeño α implica que la medida es menos sensible a errores de clasificación, y uno grande hace que la medida sea muy sensible a fallos del clasificador. Llevado al extremo, cuando α= 0, basta con que el clasificador asigne una etiqueta correcta para

obtener la máxima puntuación, y en el caso opuesto, cuando α =1 el clasificador solamente obtiene puntuación si predice correctamente todas las etiquetas. Utilizando la puntuación de la ecuación 3.9, se definen las medidas de precision, recall y accuracy para un subconjunto de clases $C \subseteq L$ según las ecuaciones 3.12, 3.11 y 3.10 respectivamente.

$$precision_C = \frac{1}{D_C} \sum_{i \varepsilon D_c} score(P_i) \tag{3.10}$$

Siendo $D_C = \{D_j \mid C = P_i\}$

$$recall = \frac{1}{|D_C|} \sum_{i \varepsilon D_c} score(P_i) \tag{3.11}$$

Siendo $D_C = \{D_j \mid C = Y_i\}$

$$accuracy_C = \frac{1}{|D|} \sum_{i \varepsilon D_c} score(P_i) \tag{3.12}$$

La *precision*, mide la fracción de instancias que contienen realmente el subconjunto de etiquetas *C* frente los que el clasificador le ha asociado el subconjunto *C*, y el *recall* la fracción de instancias que son clasificados como pertenecientes al subconjunto C frente a los que realmente lo contienen. La *accuracy* proporciona una medida del rendimiento global del clasificador, en función del *forgiveness rate*

.

3.2.1.2. MÉTRICAS DE RANKING

Las métricas utilizadas para medir el rendimiento de un algoritmo de ranking también pueden utilizarse para determinar el rendimiento de un clasificador multi-etiqueta que esté basado en técnicas de ranking. Para su definición consideraremos que *r(λ)* es la posición en el ranking de la etiqueta λ, es decir, en la etiqueta con mejor posición en el ranking, *r(λ)* = 1, la siguiente, *r(λ)* = 2 y así sucesivamente hasta la última, *r(λ)* =*n*.

La métrica one error mide el promedio de cuantas veces la etiqueta que el clasificador considera como más relevante no se encuentra en el conjunto de etiquetas asociado a la instancia3.13.

$$one\,error = \frac{1}{m} \sum_{i=1}^{m} \delta(arg\,min_{\lambda \varepsilon L}\, r_i(\lambda)) \tag{3.13}$$

Siendo

$$\delta(x) = \begin{cases} 1 & si \lambda \notin Y_i \\ 0 & si\,no \end{cases} \tag{3.14}$$

La métrica de cobertura (*coverage*) evalúa hasta qué punto hay que avanzar en la lista de ranking devuelta por el clasificador para cubrir todas las etiquetas asociadas a un ejemplo3.15.

$$coverage = \frac{1}{m}\sum_{i=1}^{m}(max_{\lambda \varepsilon Y_i} r_i(\lambda) - 1) \tag{3.15}$$

Ranking loss es una medida que nos indica el número promedio de veces en que las etiquetas irrelevantes en el ejemplo aparecen en el ranking antes que etiquetas relevantes3.16.

$$rloss = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{|Y_i||\overline{Y}_i|}|\{(\lambda_a, \lambda_b) : r_i(\lambda_a) > r_i(\lambda)\} \tag{3.16}$$

Y por último reseñar la precisión media (*average precision*) que nos indica el promedio de etiquetas con un ranking superior a una etiqueta $\lambda \in Y_i$ 3.17, es decir, nos indica cuanto hay que avanzar en promedio en el ranking hasta encontrar una etiqueta determinada.

$$avgprec = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{|Y_i|}\sum_{\lambda \varepsilon Y_i}\frac{|\{\lambda' \varepsilon Y_i : r_i(\lambda' \leq r_i(\lambda))\}|}{r_i(\lambda)} \tag{3.17}$$

3.3. Comportamiento de algoritmos perezosos en presencia de datos simbólicos

Con el objetivo de observar el comportamiento de los algoritmo perezosos ML*k*NN, BR*k*NN, IBLR-ML y GML*k*NN(sin tener en cuenta la dependencia entre las etiequetas), se implementó la función de distancia VDM para datos multi-etiqueta en el WEKA para ser llamada desde MULAN. En la experimentación, se utilizaron un número de vecinos igual a 10, un factor de *smoothing* igual a 1 y se trabajó con la muestra ya particionada en entrenamiento y prueba del conjunto de datos que se encuentra publicada en el mismo repositorio del MULAN. Se utilizaron las métricas basadas en ejemplos más comúnmente utilizadas para medir el rendimiento de los algoritmos de clasificación multi-etiqueta: *Hamming Loss*, *Average Precision, Coverage*, *One-Error* y *Ranking Loss*; ya que debido a sus características, distintas del problema de la clasificación clásica, el rendimiento de los algoritmos ha de ser determinando de forma diferente.

La tabla 3.1 muestra los conjuntos de datos utilizados en la experimentación. En cada uno de ellos se observa el número de instancias, la cantidad de atributos nominales, el número de etiquetas, el número de combinaciones de etiquetas distintas, la cardinalidad y la densidad.

N. Conjunto	|D|	Nominal	|L|	Distinct	Cardinalidad	Densidad	G. Correlación
medical	978	1449	45	94	0.245	0.028	0.0403929
genbase	662	1186	27	32	1.252	0,046	0.1038025
enron	1702	1001	53	753	3.378	0.064	0.0528559
bibtex	7395	1836	159	2856	2.402	0.015	0.0267905
delicious	16105	500	983	15806	19.020	0.019	0.0159126
corel5k	5000	499	374	3175	3.522	0.009	0.0105028
corel16k001	13811	500	153	4937	2.867	0.018	0.0234106
corel16k002	13761	500	164	4934	2.890	0.019	0.0221377
corel16k003	13760	500	154	4899	2.899	0.020	0.0229098
corel16k004	13811	500	153	4937	2.867	0.022	0.0234798
corel16k005	13761	500	164	4934	2.890	0.024	0.0225796
corel16k006	13760	500	160	4899	2.899	0.017	0.0223778
corel16k007	13811	500	153	4937	2.867	0.017	0.0229708
corel16k008	13761	500	164	4934	2.890	0.018	0.0225097
corel16k0010	13760	500	156	4899	2.899	0.019	0.0226123

Cuadro 3.1: Conjunto de Datos de la Experimentación

Los resultados experimentales de los algoritmos perezosos medido en las diferentes métricas mencionadas anteriormente se muestran en las tablas 3.2, 3.3,3.4 , 3.5 y 3.6. Aparecen en negrita los mejores resultados para cada conjunto de dato analizado en cada una de las métricas y en la última fila de cada tabla se observa el número de conjuntos de datos con mejor comportamiento para cada modelo.

Para las medidas *Hamming Loss*, *Coverage*, *One-Error* y *Ranking Loss* un modelo obtiene mejor resultado mientras más pequeño sea su valor en la métrica, en el caso de *Average Precision* es todo lo contrario.

Conjunto de datos	***Hamming Loss***			
	MLkNN	**BRkNN**	**IBLR-ML**	**GMLkNN**
medical	0.0188	0.0210	**0.0146**	0.0207
genbase	0.0052	**0.0014**	0.0015	0.0797
enron	**0.0514**	0.0584	0.0572	0.0538
bibtex	0.0140	0.0147	0.0189	**0.0133**
delicious	0.0183	0.0184	0.0184	**0.0182**
corel5k	0.0093	0.0094	0.0231	**0.0093**
corel16k001	0.0200	0.0201	0.0235	**0.0199**
corel16k002	0.0191	0.0195	0.0231	**0.0190**
corel16k003	0.0201	0.0204	0.0236	**0.0203**
corel16k004	0.0189	0.0193	0.0220	**0.0188**
corel16k005	0.0192	0.0195	0.0228	**0.0191**
corel16k006	0.0193	0.0200	0.0238	**0.0192**
corel16k007	0.0176	0.0177	0.0225	**0.0174**
corel16k008	0.0183	0.0188	0.0230	**0.0184**
corel16k0010	0.0214	0.0215	0.0242	**0.0213**
# win	1	1	1	12

Cuadro 3.2: Resultados de la experimentación en la métrica *Hamming Loss*

Conjunto de datos	***Average Precision***			
	MLkNN	**BRkNN**	**IBLR-ML**	**GMLkNN**
medical	0.6266	0.6719	**0.7049**	0.5842
genbase	0.9931	0.9876	**0.9944**	0.2881
enron	0.6345	0.5583	**0.6420**	0.6197
bibtex	0.3489	0.2681	0.3349	**0.4182**
delicious	0.3261	0.3190	0.3190	**0.3354**
corel5k	0.2656	0.2099	0.1624	**0.2713**
corel16k001	0.2667	0.1955	0.2368	**0.2851**
corel16k002	0.2831	0.1973	0.2387	**0.2986**
corel16k003	0.2849	0.2220	0.2435	**0.3045**
corel16k004	0.2687	0.1992	0.2331	**0.2835**
corel16k005	0.2843	0.2287	0.2436	**0.2966**
corel16k006	0.2838	0.2085	0.2491	**0.3076**
corel16k007	0.2545	0.2086	0.2101	**0.2832**
corel16k008	0.2870	0.2143	0.2448	**0.2974**
corel16k0010	0.2967	0.2315	0.2752	**0.3159**
# win	0	0	3	12

Cuadro 3.3: Resultados de la experimentación en la métrica *Average Precision*

Conjunto de datos	*Coverage*			
	MLkNN	**BRkNN**	**IBLR-ML**	**GMLkNN**
medical	3.5442	4.3736	**3.5438**	6.3643
genbase	0.5779	0.5367	**0.5025**	5.4070
enron	**12.1813**	20.6149	12.9551	13.2418
bibtex	56.2660	68.4755	48.7797	**47.4922**
delicious	589.8986	673.7381	673.7381	**582.4688**
corel5k	113.0460	222.3260	113.0800	**112.0260**
corel16k001	60.9931	80.5717	64.3360	**59.0631**
corel16k002	60.4481	80.4941	66.0578	**58.6052**
corel16k003	59.5847	78.0256	62.9388	**56.8864**
corel16k004	61.6592	82.6340	65.7514	**60.0143**
corel16k005	63.6088	78.2263	64.8156	**59.8087**
corel16k006	76.2447	77.9211	64.7599	**60.1503**
corel16k007	66.3114	88.3864	74.2341	**64.0401**
corel16k008	77.8579	80.3759	66.9756	**62.0777**
corel16k0010	70.8842	70.2226	54.4355	**50.9364**
# win	1	0	2	12

Cuadro 3.4: Resultados de la experimentación en la métrica *Coverage*

Conjunto de datos	*One-Error*			
	MLkNN	**BRkNN**	**IBLR-ML**	**GMLkNN**
medical	**0.3535**	0.4233	0.5256	0.4667
genbase	**0.0000**	0.0101	**0.0000**	0.9598
enron	**0.2798**	0.4629	0.3834	0.3178
bibtex	0.5757	0.6803	0.6294	**0.4994**
delicious	0.4157	0.4163	0.4163	**0.3915**
corel5k	0.7084	0.7440	0.8820	**0.7080**
corel16k001	0.7078	0.7959	0.8005	**0.7076**
corel16k002	0.7140	0.7762	0.7919	**0.6809**
corel16k003	0.7030	0.7577	0.7985	**0.6838**
corel16k004	0.7457	0.7961	0.8207	**0.7205**
corel16k005	0.7036	0.7668	0.7938	**0.6829**
corel16k006	0.7104	0.7904	0.7939	**0.6891**
corel16k007	0.7567	0.8031	0.8386	**0.7092**
corel16k008	0.7390	0.7622	0.8057	**0.7019**
corel16k0010	0.7058	0.7780	0.7598	**0.6672**
# win	3	0	1	12

Cuadro 3.5: Resultados de la experimentación en la métrica *One-Error.*

Conjunto de datos	*Ranking Loss*			
	MLkNN	**BRkNN**	**IBLR-ML**	**GMLkNN**
medical	0.0586	0.0763	**0.1029**	0.1217
genbase	0.0062	**0.0029**	0.0037	0.1846
enron	**0.0934**	0.1570	0.1124	0.0943
bibtex	0.2173	0.2965	0.1961	**0.1792**
delicious	0.1285	0.1582	0.1582	**0.1254**
corel5k	0.1297	0.2941	0.2525	**0.1283**
corel16k001	0.1995	0.2931	0.2184	**0.1928**
corel16k002	0.1786	0.2666	0.2010	**0.1711**
corel16k003	0.1872	0.2644	0.2012	**0.1761**
corel16k004	0.1936	0.2793	0.2098	**0.1878**
corel16k005	0.1896	0.2578	0.2085	**0.1866**
corel16k006	0.1917	0.2582	0.2025	**0.1811**
corel16k007	0.1935	0.2736	0.2237	**0.1860**
corel16k008	0.1813	0.2566	0.0005	**0.1803**
corel16k0010	0.1852	0.2664	0.1926	**0.1759**
# win	1	1	1	12

Cuadro 3.6: Resultados de la experimentación en la métrica *Ranking Loss*

Se observa en los resultados de los modelos ML*k*NN, BR*k*NN, IBLR-ML y GML*k*NN para las cinco métricas analizadas que el número de conjunto de datos con un comportamiento mejor utilizando GML*k*NN es superior significativamente a los otros modelos y dichos conjuntos coinciden en todas las métricas. Se observa que el modelo GMLkNN es el que obtiene un número mayor de conjuntos de datos con mejor comportamiento, seguido por IBLR-ML, después el ML*k*NN y por último el BR*k*NN.

Al analizar los conjuntos que tienen un buen comportamiento con el modelo GML*k*NN nos percatamos que son conjuntos con un gran número de instancias, no teniendo esta característica los tres primeros conjuntos *medical,genbase* y *enron* para los cuales los resultados no son buenos. Otra característica importante es que en estos conjuntos de datos el grado de correlación entre las etiquetas(dependencia incondicional) es pequeña en comparación con la correlación de los tres primeros conjuntos que no tienen buen comportamiento con GML*k*NN.

Los resultados de los modelos se han comparados estadísticamente utilizando el test de Friedman[46]]. Aplicando este test a los resultados experimentales de cada métrica podemos ver en la tabla3.7cada uno de los modelos con su puntuación promedio y en la última columna el valor de significación dado por este test.

Métrica	ML*k*NN	BR*k*NN	IBLR-ML	GML*k*NN	*p* de Fridman
Hamming Loss	1.83333	3.03333	3.56666	**1.56666**	1.83616E-5
Average Precision	2.06666	3.69999	2.7	**1.533334**	3.30084E-5
Coverage	2.06666	3.76666	2.63333	**1.533333**	1.78391E-5
One-Error	1.83333	3.16666	3.59999	**1.4**	1.53525E-6
Ranking Loss	2.06666	3.63333	2.76666	**1.53333**	5.64958E-5

Cuadro 3.7: Aplicar el test de Friedman a los resultados experimentales en cada una de las métricas
.

Se observa que en todas las métricas el modelo GML*k*NN tiene mejor puntuación promedio y el valor *p* calculado por este test nos indica que hay diferencias significas en el comportamiento de los modelos.

Se procede entonces a realizar la prueba multi-paramétrica de Holm con una significación α=0.05 y teniendo como mejor ranqueado al algoritmo GML*k*NN, obteniendo como resultados en cada una de las métrica que las diferencias están entre nuestra propuesta y los algoritmos ML*k*NN y BR*k*NN, alcanzando resultados similares GML*k*NN e IBLR-ML.

3.4. Resultados experimentales entre IBLR-ML y el modelo GMLkNN

En esta sección se muestran los resultados experimentales entre los modelos IBLR-ML y el modelo GML*k*NN teniendo en cuenta la dependencia entre las etiquetas, este último se describió en el epígrafe 2.2.2. Se decidó escoger el algoritmo IBLR-ML para este estudio ya que en los experimentos anteriores mostró obtener buenos resultados y además por tener en cuenta la dependencia existente entre las etiquetas como nuestra propuesta.

Las tablas 3.8 y 3.9 muestran los resultados de la experimentación en las métricas utilizadas anteriormente, las tablas tienen las mismas características que en la experimentación del epígrafe 3.3.

Conjunto de datos	*Hamming Loss*		*Average Precision*		*Coverage*	
	IBLR-ML	**GMLkNN**	**IBLR-ML**	**GMLkNN**	**IBLR-ML**	**GMLkNN**
medical	**0.0146**	0.0206	**0.7049**	0.5862	**3.5438**	6.3647
genbase	**0.1247**	0.0786	**0.9944**	0.2897	**0.5025**	4.4450
enron	0.0586	**0.0505**	0.6420	**0.6433**	12.9551	**12.1118**
bibtex	0.0178	**0.0121**	0.3349	**0.4789**	48.7797	**46.4892**
delicious	0.0184	**0.0178**	0.3190	**0.3456**	673.7381	**572.4678**
corel5k	0.0222	**0.0088**	0.1624	**0.2811**	199.0800	**110.0770**
corel16k001	0.0236	**0.0191**	0.2368	**0.2800**	64.3360	**56.0731**
corel16k002	0.0212	**0.0188**	0.2387	**0.2956**	66.0578	**57.6882**
corel16k003	0.0250	**0.0198**	0.2435	**0.3077**	62.9388	**55.8454**
corel16k004	0.0209	**0.0176**	0.2331	**0.2853**	65.7514	**60.0103**
corel16k005	0.0210	**0.0189**	0.2436	**0.2978**	64.8156	**55.4687**
corel16k006	0.0233	**0.0181**	0.2491	**0.3087**	64.7599	**59.9903**
corel16k007	0.0211	**0.0169**	0.2101	**0.2834**	74.2341	**65.0001**
corel16k008	0.0216	**0.0174**	0.2448	**0.2999**	66.9756	**61.04477**
corel16k0010	0.0258	**0.0203**	0.2752	**0.3186**	54.4355	**51.0064**
# win	2	13	2	13	2	13

Cuadro 3.8: Resultado de la experimentación en las métricas *Hamming Loss*, *Average Precision* y *Coverage*

Conjunto de datos	*One-Error*		*Ranking Loss*	
	IBLR-ML	**GMLkNN**	**IBLR-ML**	**GMLkNN**
medical	0.5256	**0.3667**	**0.1029**	0.1209
genbase	**0.0000**	0.9198	**0.0027**	0.1046
enron	0.3834	**0.1278**	0.1124	**0.0932**
bibtex	0.6294	**0.4900**	0.1961	**0.1793**
delicious	0.4163	**0.2905**	0.1582	**0.1094**
corel5k	0.8820	**0.7000**	0.2525	**0.1223**
corel16k001	0.8005	**0.6006**	0.2184	**0.1888**
corel16k002	0.7919	**0.5805**	0.2010	**0.1091**
corel16k003	0.7985	**0.5538**	0.2012	**0.1541**
corel16k004	0.8207	**0.6705**	0.2098	**0.1768**
corel16k005	0.7938	**0.6009**	0.2085	**0.1346**
corel16k006	0.7939	**0.5191**	0.2025	**0.1451**
corel16k007	0.8386	**0.6792**	0.2237	**0.0960**
corel16k008	0.8057	**0.6529**	0.2005	**0.0903**
corel16k0010	0.7598	**0.6022**	0.1926	**0.1549**
# win	1	14	2	13

Cuadro 3.9: Resultado de la experimentación en las métricas *One Error* y *Ranking Loss*

Se puede observar que los dos primeros conjuntos de datos: *medical* y *genbase* obtienen resultados satisfactorios en todas las métricas utilizando el modelo IBLR-ML mientras que los otros conjuntos obtienen resultados mejores con GML*k*NN.

Destacar que los 13 conjuntos de datos donde el modelo GML*k*NN tiene mejores resultados presentan la característica que el promedio de etiquetas asociadas a cada instancia es mayor en comparación con *medical* y *genbase* donde el modelo IBLR-ML tiene un buen comportamiento, es decir, a el modelo GML*k*NN captura mejor la dependencia entre las etiquetas que el IBLR-ML.

Estos resultados se comparan estadísticamente utilizando el test de Friedman y las puntuaciones promedios así com el valor de *p* calculado por este test se observan en la tabla 3.10.

Métrica	**IBLR-ML**	**GMLkNN**	***p* de Fridman**
Hamming Loss	1.93333	**1.06666**	7.89112E-4
Average Precision	1.86666	**1.13333**	0.00450
Coverage	1.86666	**1.13333**	0.00450
One-Error	1.93333	**1.06666**	7.89112E-4
Ranking Loss	1.86666	**1.13333**	0.00450

Cuadro 3.10: Resultados de aplicar el test de Fridman a los resultados experimentales en cada una de las métricas .

Al observar la tabla nos percatamos que el modelo de mejor puntuación promedio es GML*k*NN. En todas las métricas el valor de significacion ofrecido por este test es menor que 0.05 por lo que se rechaza la hipotesis de igualdad obteniendo diferencias significativas entre los modelos, destacando que nuestra propuesta, GML*k*NN, obtiene para todas las métricas el mejor valor de *ranking*.

3.5. Experimentos con Datos Ruidosos

Una buena medida para observar el comportamiento de un algoritmo es ver la resistencia y robustez ante la presencia de ruido. Con este propósito se modificaron un 5 %, 10 %, 15 %, 20 % y 25 % de los valores de los rasgos que describen los objetos en cada conjunto de datos. Con la ayuda del MATLAB (abreviatura de *MATrix LABoratory*, "laboratorio de matrices") se realizó un programa para la inserción de ruido en los conjuntos de datos con los porcientos mencionados y así observar el comportamiento de estos.

Se analizaron los modelos IBLR-ML y GMLkNN ante diferentes perturbaciones de ruido. Se realizó el experimento con cuatro conjuntos de datos *medical, genbase*, *bibtex* y *corel5K* a los cuales se le modificaron un 5 %, 10 %, 15 %, 20 % y 25 % los valores de los atributos. En la experimentación de estos conjuntos se observó que el comportamiento de estos coincidía y se decidió mostrar solamente los resultados para el conjunto datos *medical*.

Las siguientes tablas muestran el comportamiento del conjunto de datos medical para las diferentes métricas, comparándolo con el valor original del modelo, es decir con ningún nivel de ruido frente a diferentes porcientos de inserción de ruido.

```
% Generación aleatoria de ruido en conjuntos de datos
% instrumento de medición
matrix=dlmread('datos.txt');
[N,M] = size(matrix);
%crea una lista de 1X5 de matrices vacias
ListNoiseMatrix = cell(1,5);
TNum = N*M;
Inc = 0;
for I=1:5
    Inc = Inc + 5;
    NoiseMatrix = matrix;
    % Número de valores a los que se le introducirá ruido
    PNum = round((Inc*TNum)/100) ;
    %genera una matriz uniformemente distribuida de tamanyo PNumX1 en el rango de 1 a N
    RowList = randint(PNum,1,[1,N]);
    ColList = randint(PNum,1,[1,M]);
    % Ciclo para introducir ruido a las PNum celdas
    for J=1:PNum
        %obtengo el valor que esta en esa posicion
        OrigValue = NoiseMatrix(RowList(J),ColList(J));
         %altero los valores de las celdas
          if  NoiseMatrix(RowList(J),ColList(J))==0
           NoiseMatrix(RowList(J),ColList(J)) = 1;
            else
             NoiseMatrix(RowList(J),ColList(J)) = 0;
          end
    end
    ListNoiseMatrix{I} = NoiseMatrix;
    % fichero + numero
    n_STR = num2str(Inc); % 'i' es la variable del ciclo for
    Name = ['datosRuidosos' n_STR '.txt'];
    dlmwrite(Name,NoiseMatrix) ;
end
```

Figura 3.1: Código en Matlab para la generación de ruido

Medical	***Hamming Loss***		***Average Precision***		**Coverage**	
	IBLR-ML	**GMLkNN**	**IBLR-ML**	**GMLkNN**	**IBLR-ML**	GMLkNN
Original	0.0146	0.0207	0.7049	0.5842	3.5438	6.3643
5 %	0.0149	0.0205	0.6299	0.5840	3.5900	6.3641
10 %	0.0182	0.0203	0.7004	0.5842	3.6053	6.3646
15 %	0.0187	0.0212	0.7121	0.5866	3.7472	6.3653
20 %	0.0178	0.0210	0.7203	0.5854	3.8084	6.3652
25 %	0.0191	0.0211	0.7107	0.5851	3.8513	6.3640

Cuadro 3.11: Resultado de la experimentación con datos ruidosos en las métricas *Hamming Loss, Average Precision* y *Coverage* .

Medical	***One-Error***		***Ranking Loss***	
	IBLR-ML	**GMLkNN**	**IBLR-ML**	**GMLkNN**
Original	0.5256	0.4667	0.1029	0.1217
5 %	0.5613	0.4662	0.1596	0.1278
10 %	0.5388	0.467	0.1696	0.1288
15 %	0.5003	0.4673	0.1072	0.1297
20 %	0.5595	0.4632	0.1946	0.1263
25 %	0.5423	0.4676	0.1306	0.1256

Cuadro 3.12: Resultado de la experimentación con datos ruidosos en las métricas *One-Error* y *Ranking Loss.*

Estos resultados se mostrarán a continuación utilizando tablas de dispersión para observar que robusto es cada modelo ante diferentes valores de ruido. Vemos los dos modelos ML*k*NN y GML*k*NN, en el eje vertical los valores de la métrica y en el horizontal los diferentes niveles de ruido que le fueron insertado a los conjunto de datos.

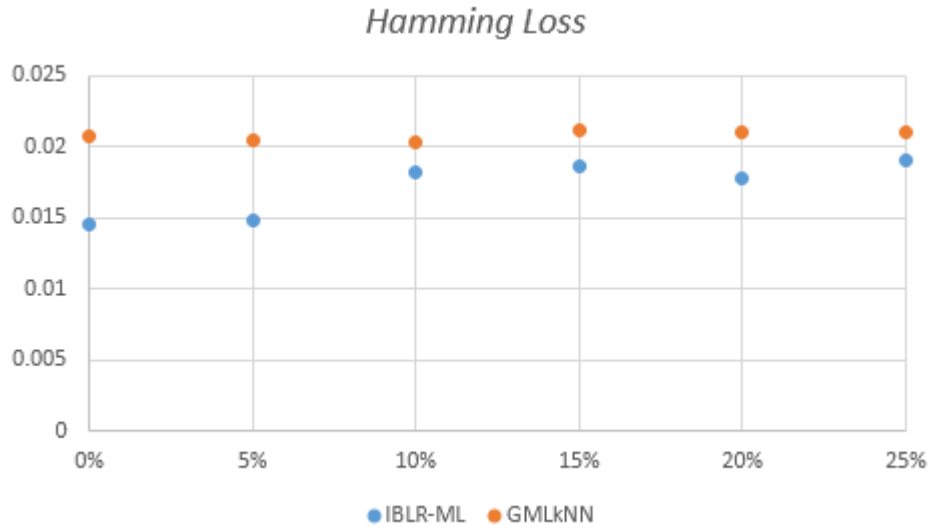

Figura 3.2: Comportamiento del conjunto de dato medical en los moedelo MlkNN y GMLkNN en la métrica *Hamming Loss*.

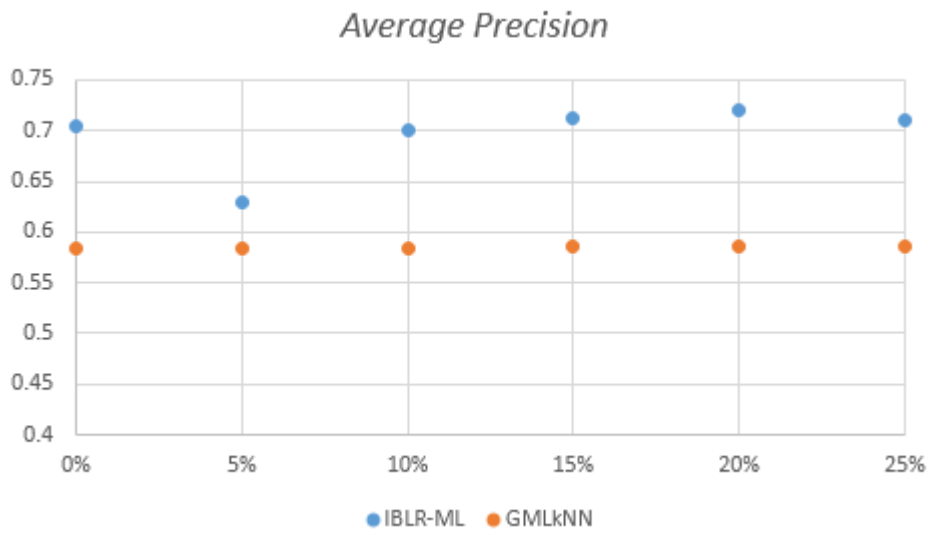

Figura 3.3: Comportamiento del conjunto de dato medical en los moedelo MlkNN y GMLkNN en la métrica *Average Precision.*

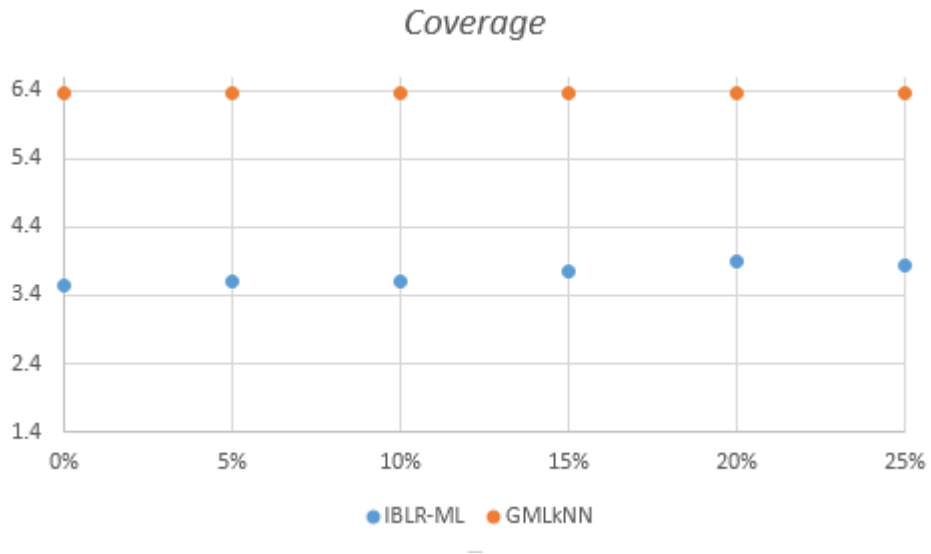

Figura 3.4: Comportamiento del conjunto de dato medical en los moedelo MlkNN y GMLkNN en la métrica *Coverage*.

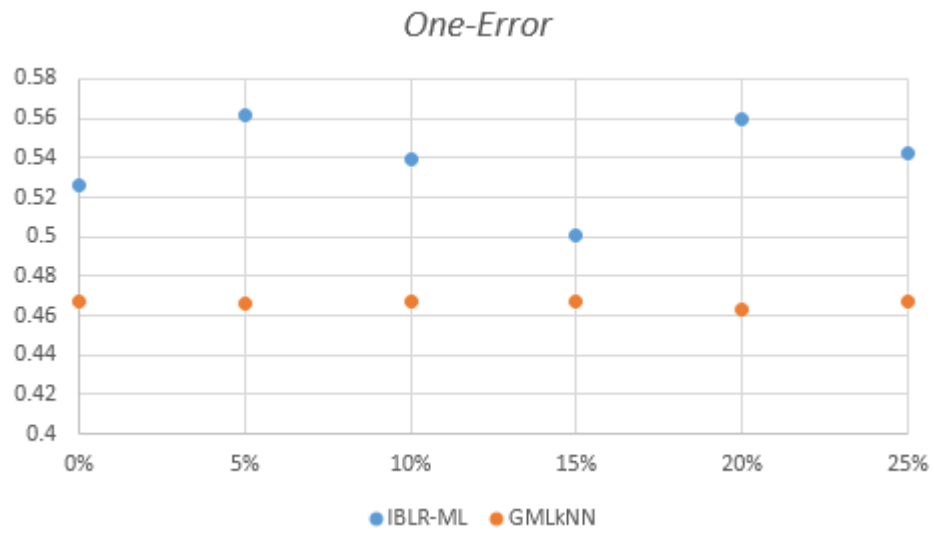

Figura 3.5: Comportamiento del conjunto de dato medical en los moedelo MlkNN y GMLkNN en la métrica *One-Error*.

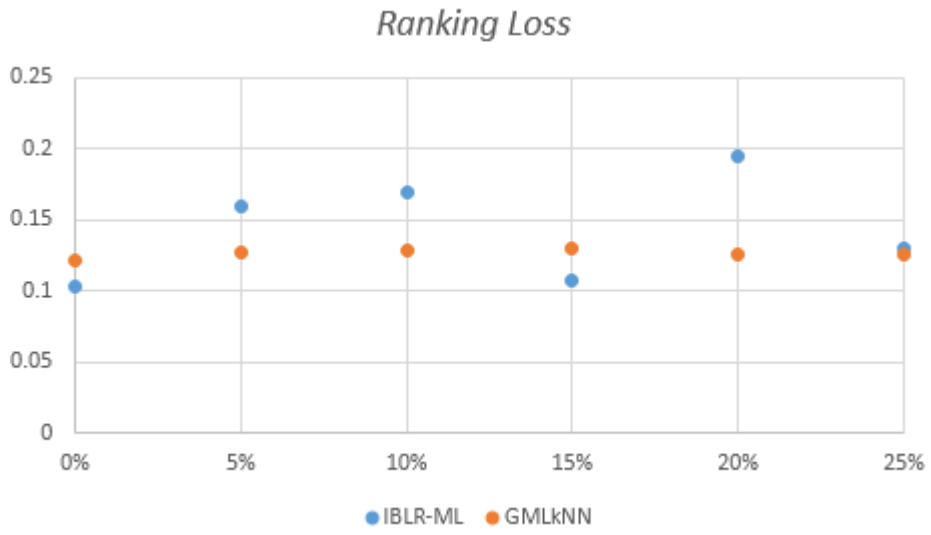

Figura 3.6: Comportamiento del conjunto de dato medical en los moedelo MlkNN y GMLkNN en la métrica *Ranking Loss*.

Al observar las tablas de dispersión podemos notar que ante diferentes perturbaciones de ruido el algoritmo IBLR-ML presenta inestabilidad. En las métricas *Hamming Loss* y *Coverage* el comportamiento de IBLR-ML empeora a medida que el nivel de ruido aumenta, para las métricas *Average Precision, One-Error* y *Ranking Loss* el comportamiento de este es bastante inestable mientras que para todas las métricas GMLkNN es bastante estable ante la presencia de ruido.

CONCLUSIONES

Los resultados obtenidos permiten concluir que:

- Se extiende el modelo ConFuCiuS a datos multi-etiquetas con la creación de un nuevo modelo llamado GML*k*NN el cual utiliza la función de distancia relacionada con el objetivo VDM para comparar la instancia de prueba y la de entrenamiento en la búsqueda de los *k* vecinos más cercanos.

- En la comparación de GML*k*NN con los algortimos perezosos presentes en la literatura se obtiene un mejor comportamiento en 12 de los 15 conjuntos de datos, siendo estadísticamente mejor este modelo, adecuado para trabajar con conjuntos de datos con un gran número de instancias.

- Al comparar la versión de GML*k*NN teniendo en cuenta la dependencia entre las etiquetas con el modelo IBLR-ML observamos que es más eficaz en 13 de los 15 conjunto de datos analizados, obteniendo diferencias siginificativas alcanzando nuestro modelo el mejor valor de *ranking*.

- Se observó la robustez del modelo GML*k*NN ya que su comportamiento en conjuntos de datos con presencia de ruido es bastante estable.

RECOMENDACIONES

Como recomendaciones

- Obtener una versión fuzzy de GML*k*NN ya que al resolver problemas de MLC, utilizando la modelación borrosa del espacio de salida, hace que la modelación del problema (vía de solución) sea más natural y cercana a un problema de clasificación.
- Obtener un modelo que tenga en cuenta múltiples etiquetas también para rasgos predictores en el espacio de entrada.

REFERENCIAS BIBLIOGRÁFICAS

[1] I. H. Witten, E. Frank, and M. A. Hall, *Data Mining: Practical Machine Learning Tools and Techniques, Third Edition*. Morgan Kaufmann, 3 ed., Jan. 2011.

[2] M. J. J. Hernández, Ramírez and C. Ferri, "Introducción a la minería de datos," *Pearson Prentice Hall*, 2004.

[3] M. A. H. Ian H. Witten, E.F., *Data mining : practical machine learning tools and techniques*. 2011.

[4] N. J. Nilsson, "Introduction to machine learning," *Stanford University*, 1996.

[5] P. H. T. M. COVER, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, pp. 21–27, 1967.

[6] M. AHA, D.W. D.K.; ALBERT, "Instance-based learning algorithms," *Machine Learning*, vol. 6, pp. 37–66, 1991.

[7] C. Stanfill and D. Waltz, "Toward memory-based reasoning," *Communications of the ACM*, vol. 29, no. 12, pp. 1213–1228, 1986.

[8] M. T. D. Wettschereck, Aha D. W., "A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms," *Artif. Intell. Rev.*, vol. 11, pp. 273–314, 1997.

[9] A. M. R. Morell, "ExtensiÓn a la clasificaciÓn de flujos de datos del modelo confucius," *II Conferencia Internacional de Ciencias Computacionales e Informática*, 2013.

[10] A. M. R. Morell, "ExtensiÓn del modelo de clasificaciÓn confucius a la clasificaciÓn de flujos de datos," *XVI Convención Científica de Ingeniería y Arquitectura*, 2012.

[11] M.-L. Zhang and Z.-H. Zhou, "ML-KNN: a lazy learning approach to multi-label learning," *Pattern Recognition*, vol. 40, pp. 2038–2048, July 2007.

[12] S. Parka and J. Furnkranz, "Efficient pairwise classification," *Proceedings of the 18th European Conference on Machine Learning*, pp. 658–665, 2007.

[13] M. R. Boutell, J. Luo, X. Shen, and C. M. Brown, "Learning multi-label scene classification," *Pattern recognition*, vol. 37, no. 9, pp. 1757–1771, 2004.

[14] Y. J. Z. B. C. Z. Chen, W. and Q. Yang, "Document transformation for multi-label feature selection in text categorization," *Proceedings of the Seventh IEEE International Conference on Data Mining , Omaha, NE*, pp. 451–456, 2007.

[15] J. Read, "A pruned problem transformation method for multi-label classification," *Proceedings of the New Zealand Computer Science Research Student Conference, Christchurch, New Zeaand*, 2008.

[16] G. Tsoumakas and I. R. k.-l. Vl ahavas, "An ensemble method for multilabel classification," *Machine Learning: ECML*, pp. 406–417, 2007.

[17] E. Spyromitros, G. Tsoumakas, and I. Vlahavas, "An empirical study of lazy multilabel classification algorithms," in *Artificial Intelligence: Theories, Models and Applications* (J. Darzentas, G. Vouros, S. Vosinakis, and A. Arnellos, eds.), vol. 5138 of *Lecture Notes in Computer Science*, pp. 401–406, Springer Berlin / Heidelberg, 2008.

[18] J. XU, "An extended one-versus-rest support vector machine for multi-label classification," *Neurocomputing*, vol. 74(17), pp. 3114–3124, 2011.

[19] M. MYINT, E. E.; PWINT, "An approach for mulit-label music mood classification," *3rd International Conference on Signal Processing Systems*, 2010.

[20] A. GOLD, K.; PETROSINO, "Using information gain to build meaningful decision forests for multilabel classification," *IEEE 9th International Conference on Development and Learning (ICDL)*, pp. 58–63, 2010.

[21] M. A.-J. J. L. V. S. GIBAJA, E.; VICTORIANO, "A tdidt technique for multi-label classification," *10th International Conference on. Intelligent Systems Design and Applications*, pp. 519–524, 2010.

[22] M.-L. ZHANG, "Ml-rbf: Rbf neural networks for multi-label learnin," *Neural Processing Letters*, vol. 29, pp. 61–74, 2009.

[23] M.-L. Z. X.-H. ZHOU, "Multilabel neural networks with applications to functional genomics and text categorization," *IEEE Transactions on Knowledge and Data Engineering,*, vol. 18, pp. 1338–1351, 2006.

[24] R. R. L. K. REFORMAT, "A tree-projection-based algorithm for multi-label recurrent-item associative-classification rule generation," *Data & Knowledge Engineering*, vol. 64, pp. 171–197, 2008.

[25] C. B. L. . P. LARRAÑAGA, "Multi-dimensional classification with bayesian networks," *International Journal of Approximate Reasoning*, vol. 52, pp. 705–727, 2011.

[26] R. V. D. G. X. L. D. A. CARVALHO, "A new approach for multi-label classification based on default hierarchies and organizational learning," *GECCO '08: Proceedings of the 2008 GECCO conference companion on Genetic and evolutionary computation*, vol. 2017-2022, 2008.

[27] Z. Younes, F. Abdallah, and T. DenÃžeux, "Multi-label classification algorithm derived from k-nearest neighbor rule with label dependencies," in *roceedings of the 16th European Signal Processing Conference*, 2008.

[28] X. Lin and X.-w. Chen, "Mr.KNN: soft relevance for multi-label classification," in *Proceedings of the 19th ACM international conference on Information and knowledge management*, CIKM '10, (New York, NY, USA), pp. 349–358, ACM, 2010.

[29] J.-Y. Jiang, S.-C. Tsai, and S.-J. Lee, "FSKNN: multi-label text categorization based on fuzzy similarity and k nearest neighbors," *Expert Systems with Applications*, vol. 39, pp. 2813–2821, Feb. 2012.

[30] J. Xu, "Multi-label weighted k-nearest neighbor classifier with adaptive weight estimation," in *Neural Information Processing* (B.-L. Lu, L. Zhang, and J. Kwok, eds.), vol. 7063 of *Lecture Notes in Computer Science*, pp. 79–88, Springer Berlin / Heidelberg, 2011.

[31] W. Cheng and E. HÃŒllermeier, "Combining instance-based learning and logistic regression for multilabel classification," *Machine Learning*, vol. 76, no. 2, pp. 211–225, 2009.

[32] W. A. G. Grzegorz, "Riona: A classifier combining rule induction and k-nn method with automated selection of optimal neighbourhood," *ECML*, 2002.

[33] E. P. T. R. Payne, "mplicit feature selection with the value difference metric," *European Conference on Artific ial Intelligence,Brighton, UK*, 1998.

[34] J. M. Keller, M. R. Gray, and J. A. Givens, "A fuzzy k-nearest neighbor algorithm," *Systems, Man and Cybernetics, IEEE Transactions on*, no. 4, pp. 580–585, 1985.

[35] G. Tsoumakas, E. Spyromitros-Xioufis, J. Vilcek, and I. Vlahavas, "MULAN: a java library for multi-label learning," *J. Mach. Learn. Res.*, vol. 12, pp. 2411–2414, July 2011.

[36] B. R. Y. Sasaki and S. Ananiadou, "Multi-topic aspects in clini-cal text classification," *in BIBM '07: Proceedings of the 2007 IEEE International Conference on Bioinformatics and Biomedicine, (Wa-shington, DC, USA)*, pp. 62–70, 2007.

[37] P. M. S. Diplaris, G. Tsoumakas and I. Vlahavas, "Protein classification with multiple algorithms," *Advances in Informatics*, pp. 448–456, 2005.

[38] G. T. I. Katakis and I. Vlahavas, "Multilabel text classification for automated tag suggestion," *in Proceedings of the ECML/PKDD 2008 Discovery Challenge*, pp. 75–83, 2008.

[39] P. Keila and D. Skillicorn, "Structure in the enron email dataset," *Computational & Mathematical Organization Theory*, vol. 11, pp. 183–199, october,2005.

[40] I. V. G. Tsoumakas, I. Katakis, "Effective and efficient multilabel classification in domains with large number of labels," *Proc. ECML/PKDD 2008 Workshop on Mining Multidimensional Data (MMD'08), Antwerp, Belgium*, 2008.

[41] N. d. F. Pinar Duygulu, Kobus Barnard and D. Forsyth, "Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary," *7th European Conference on Computer Vision*, pp. 97–112, 2002.

[42] G. Tsoumakas, I. Katakis, and I. Vlahavas, "Mining multi-label data," *Data mining and knowledge discovery handbook*, pp. 667–685, 2010.

[43] G. Tsoumakas, I. Katakis, and I. Vlahavas, "Random k-labelsets for multilabel classification," *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, pp. 1079 –1089, July 2011.

[44] X. Z. J. Chen and Z. Wu, "A multi-label chinese text categorization system based on boosting algorithm," *inCIT '04: Proceedings of the The Fourth International Conference on Computer and Infor-mation Technology, (Washington, DC, USA)*, pp. 1153–1158, 2004.

[45] X. S. M. R. Boutell, J. Luo and C. M. Brown, "Learning multi-label scene classification1," *Pattern Recognition*, vol. 37, pp. 1757–1771, Sept,2004.

[46] J. Demsar, "Statistical comparisons of classifiers over multiple datasets," *Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.