UNIVERSIDAD POLITECNICA DE VALENCIA

DSIC
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

TRABAJO FIN DE MASTER

Departamento de Sistemas Informáticos y Computación

Máster Universitario en Ingeniería y Tecnología de Sistemas Software

Estudio Experimental Comparativo de Diversos Métodos para Aprender Dominios Jerárquicos

Autor: Daniel Andres Silva Palacios
Tutor: César Ferri Ramirez
Tutor: María José Ramirez

Curso 2014-15

Valencia, 21 de Septiembre de 2015

AGRADECIMIENTOS

Primeramente quisiera agradecer a mis tutores en este proyecto de tesis María José Ramírez y Cesar Ferri por compartir sus ideas y conocimientos en el área, lo cual permitió realizar exitosamente este trabajo de investigación.

Quisiera agradecer a Ecuador ya que han creído en mis capacidades, y por haberme confiado una beca de estudios que me permitió aprender y aplicar todos mis conocimientos.

A mi familia que siempre me apoyó a la distancia en especial a mi madre y padre que siempre estuvieron pendientes de mí y listos para animarme a seguir mis sueños. A mi esposa que es mi compañera de vida y estuvo conmigo cada segundo de toda esta aventura.

Gracias a mis compañeros de clase y profesores que contribuyeron con su apoyo y conocimiento durante todo este tiempo que duro mis estudios.

Finalmente quisiera agradecer al creador por darme la vida y su bendición.

RESUMEN

Este proyecto de investigación trata de resolver el problema de la falta de una jerarquía de clases para el entrenamiento de un clasificador jerárquico. Ya que actualmente se necesita un experto o una base de datos que proporcione esta jerarquía, se ha buscado un método automático que pueda realizar la inferencia de la jerarquía de clases y permita la construcción de un clasificador jerárquico.

La investigación ha partido buscando información relacionada con las diferentes metodologías para la creación de clasificadores jerárquicos. Se han analizado los diferentes tipos de clasificadores, sus características, funcionamiento y algunos métodos utilizados para su evaluación.

Ya que es importante la inferencia de las jerarquías de clase, se ha propuesto una serie de funciones que ayuden en el cálculo de distancias entre clases y permitan construir jerarquías. También se realizó una investigación sobre los métodos de agrupamiento jerárquico más utilizados en la literatura, ya que fue una herramienta importante para encontrar las jerarquías de clase.

Posteriormente se realizó una serie de experimentos que utilizan los conceptos estudiados anteriormente, donde se realiza la inferencia de jerarquía a través de varios métodos de cálculo aplicado a tres bases de datos diferentes. Luego se realizó la construcción de un clasificador jerárquico por cada jerarquía de clase y se evaluó su rendimiento a través de indicadores de rendimiento. Finalmente se comparó estos indicadores con lo que se presentan en clasificadores planos con la intensión de encontrar las diferencias en los rendimientos de cada uno.

PALABRAS CLAVES

Aprendizaje automático, clasificación jerárquica, agrupamiento jerárquico, análisis de datos.

TABLA DE CONTENIDOS

LISTA DE TABLAS

LISTA DE FIGURAS

CAPÍTULO

1 INTRODUCCIÓN

1. Introducción

Este capítulo presenta una introducción al problema de investigación que será tratado en el resto de capítulos, los objetivos que persigue y los factores que motivaron la realización de este trabajo fin de máster.

1.1. Motivación

En esta era digital donde la información mueve la economía mundial, no es posible negar la importancia de la información y cómo afecta nuestras vidas. Plataformas de redes sociales y nuevos negocios aparecen cada minuto en todo el mundo. Nuevas tecnologías y protocolos que permiten el intercambio seguro de información a través del internet. El aumento de usuarios y mejores redes de comunicación han permitido el crecimiento desmedido de la información que se publica en internet.

Hoy en día estamos siempre conectados a través de nuestros equipos móviles, relojes y diferentes sensores al internet. Las empresas productoras de coches, juguetes y otros equipos de uso diario están incorporando en sus productos sistemas que les permitan conectarse a la red. El problema con este flujo creciente de información es su falta de estructura y organización, lo que ha llevado a que cada día sea más difícil seleccionar metodologías adecuadas para realizar una correcta clasificación automática de los datos.

Algunos de los problemas de clasificación se pueden tratar usando aprendizaje automático y por lo general se asume que las clases a predecir no tienen una estructura jerárquica, pero en problemas reales se ha podido observar que estas clases tienen relaciones y forman jerarquías de clases.

La jerarquización parece ser natural y coherente con la realidad y como se ha visto en muchas investigaciones en el campo de la biología, los objetos de estudio no se enmarcan únicamente dentro de una clase, si no dentro de una jerarquía

de clases. Si tomamos ventaja de esta característica, es posible la creación modelos de predicción que utilicen la información contenida en las jerarquías para mejorar los modelos utilizados en clasificación.

En la literatura se han visto diferentes aplicaciones de clasificadores jerárquicos en diferentes campos de conocimiento como es la clasificación de textos, clasificación de pájaros, clasificación de imágenes, etc. Actualmente existen estudios relacionados con los clasificadores jerárquicos y sus aplicaciones, y se han desarrollado varios tipos de estos clasificadores. La mayoría de estas aproximaciones usan jerarquías dadas por el usuario o extraídas de otras bases de datos.

Este proyecto de investigación busca inferir jerarquías de clases partiendo de datos a través de varios procedimientos, para luego crear una serie de clasificadores jerárquicos que utilicen la información de la jerarquía para realizar predicciones. Con los resultados obtenidos se podrá realizar una comparación entre el rendimiento de un clasificador que no utilice una jerarquía y un clasificador que sí. Además se podrá buscar el procedimiento óptimo que nos permita inferir jerarquías de problemas similares a los planteados en los experimentos.

1.2. Objetivos

Este trabajo se encuentra en el marco de la clasificación multi-clase en general y dentro de la clasificación jerárquica en particular. A nivel teórico se ha realizado un estudio de los clasificadores jerárquicos y se han propuesto varios procedimientos para la inferencia automática de jerarquías de clases. A nivel práctico, se han realizado varios experimentos que aplican estos métodos a varios dominios de aplicación, y evalúan los modelos de clasificación jerárquicos obtenidos para comprobar las prestaciones del modelo resultante.

1.2.1. Objetivos Específicos

1. Revisión del campo clasificación jerárquica
2. Plantear métodos para realizar la inferencia de estructuras jerárquicas.
3. Realizar experimentos para generar modelos utilizando los procedimientos propuestos en diferentes dominios de aplicación.
4. Evaluar los modelos con las métricas propuestas.

1.3. Estructura del documento

Este documento se estructura de la siguiente manera:

En el Capítulo 2 se realiza una revisión del campo de los clasificadores jerárquicos y las taxonomías de conceptos.

En el Capítulo 3 se realiza un análisis de las metodologías para inferir jerarquías partiendo de matrices de confusión.

En el Capítulo 4 se describen los experimentos realizados, mostrando los resultados obtenidos.

En el Capítulo 5 se establecen las conclusiones obtenidas del presente trabajo de investigación y se proponen temas de investigación relacionados.

CAPÍTULO

2 CLASIFICACIÓN JERÁRQUICA

2. Clasificación Jerárquica

En este capítulo se profundizará en el campo de la clasificación jerárquica. Para lo cual se realizará una revisión de conceptos básicos que intervienen en la clasificación jerárquica, metodologías de implementación, métodos de evaluación y se presentará los últimos trabajos y publicaciones relacionadas.

2.1. Conceptos Previos

2.1.1. Proceso de Descubrimiento de Conocimiento en Bases de Datos (KDD)

El descubrimiento de Conocimiento en Bases de Datos (en inglés: *Knowledge Discovery in Databases*, KDD) es el proceso de descubrir información útil de los datos. Este procesamiento incluye: definir el propósito/objetivo del análisis, recopilar el conjunto de datos inicial, limpiar y pre procesar esta información, seleccionar el algoritmo de datamining para la búsqueda de patrones, ejecutar la búsqueda de patrones, realizar la interpretación de los patrones y usar esta información (Usama Fayyad, 1996).
Por ejemplo si nos interesa predecir las ventas en un supermercado de cierto cereal en el próximo año, el primer paso es la extracción exitosa de datos desde los datos históricos de ventas recopilados en el supermercado, además es necesario crear procedimientos que limpien y eliminen anomalías e inconsistencias como son ventas equivocadas o incompletas. Éste es un problema de regresión, ya que tenemos que predecir un valor numérico, por lo que podríamos aplicar una regresión lineal. Finalmente es necesario poder interpretar los resultados obtenidos para tomar decisiones. Por ejemplo, si se pronostica una menor venta de cereal es necesario disminuir la adquisición de este producto del proveedor o incluir un descuento dentro del catálogo de cupones del supermercado para incentivar su venta.

El procesamiento previo y limpieza de datos es una de las tareas más complejas ya que muchas veces los datos poseen ruido y valores atípicos. Una vez obtenido una vista minable es posible crear modelos que representen los datos.

El proceso de minería de datos parte desde que se formula las preguntas sobre un problema en particular, la creación de un modelo para responderlas, hasta la implementación del modelo en un entorno de trabajo (Microsoft SQL Server, 2014). Este es un proceso iterativo y como se puede observar en la Figura 1, es un proceso repetitivo que busca mejorar continuamente el modelo generado y con cada paso redefinir el conocimiento que se tiene del problema. Los pasos básicos son los siguientes:

1. Definir el problema.
2. Preparar los datos.
3. Explorar los datos.
4. Generar los modelos.
5. Explorar y validar los modelos.
6. Implementar y actualizar los modelos.

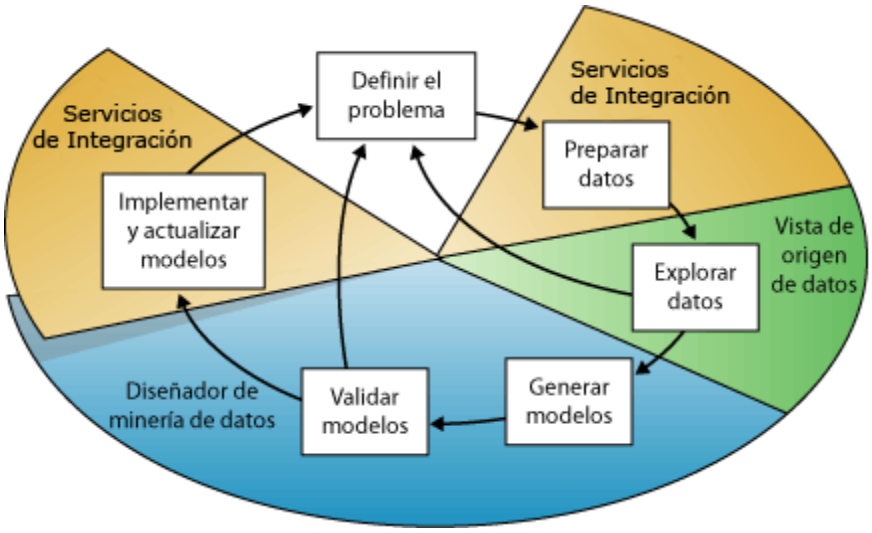


Figura 1. Proceso de minería de datos.
(Microsoft SQL Server, 2014)

Cuando logramos que un computador pueda aprender automáticamente tareas y como se mencionó anteriormente crear modelos y reproducir tareas complejas nos referimos al aprendizaje automático el cual será desarrollado en la siguiente sección.

2.1.2. Aprendizaje Automático

Aprendizaje Automático (En inglés: *Machine Learning*) se enfoca en la construcción de sistemas que requieren de una mínima intervención humana para aprender de los datos. No es un campo nuevo de estudio si no que se remonta a 1958 cuando se introduce la primera técnica de aprendizaje neuronal (Hall, Dean, & Kabul , 2014), la misma que no tuvo un alto impacto debido a sus limitaciones tecnológicas y costoso procesamiento.

En 1970s aparecieron lenguajes de computación como LISP[1] y PROLOG[2] que permitieron resolver problemas específicos para ciertos dominios. En 1980 aparecieron los llamados sistemas expertos (Waterman & Hayes-Roth, 1978) y también se intensificó el estudio de redes neuronales con la aparición de los perceptores multicapa.

Los métodos de clasificación y regresión fueron muy estudiados en los años ochenta, en especial los métodos de regresión a través de árboles (Breiman, Friedman, & Olshen, 1984).

[1] LISP.- Es una familia de lenguajes de programación multiparadigma especificado originalmente en 1958 y fue especialmente utilizado para en la investigación de inteligencia artificial.
[2] PROLOG.- Lenguaje para programar artefactos electrónicos mediante el paradigma lógico con técnicas de producción final interpretada.

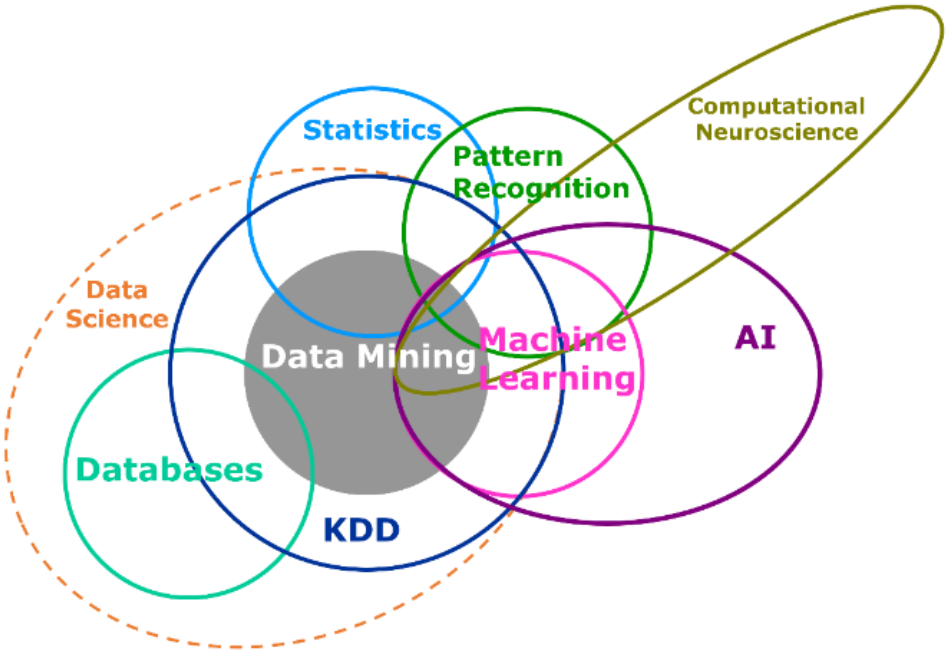


Figura 2. Naturaleza multidisciplinar del aprendizaje automático.
(Hall, Dean, & Kabul , 2014)

Como se puede observar en la Figura 2, el aprendizaje automático es el resultado de la unión de un conjunto de disciplinas como son: estadística, reconocimiento de patrones, minería de datos, ciencia de datos, inteligencia artificial, computación neuronal y forma parte del proceso de descubrimiento de conocimiento. Hoy en día hay muchos algoritmos de aprendizaje automático aplicados en diferentes campos como son: medicina, robótica y finanzas. (Hall, Dean, & Kabul , 2014)

Cabe mencionar que dentro del aprendizaje se distingue entre problemas predictivos y descriptivos. Los primeros tienen como objetivo predecir el valor de una variable objetivo, mientras que las tareas descriptivas tienen como objetivo describir los datos. Dentro del aprendizaje supervisado distinguimos entre clasificación y regresión. En la clasificación el atributo a predecir puede tomar un valor de un conjunto de valores nominales (que denominamos clases), mientras que el problema es de regresión cuando el objetivo es predecir valores de una variable continua.

2.2. Clasificación Jerárquica

Un área típica dentro del campo de aprendizaje automático se enfoca en el estudio de clasificadores planos que se caracterizan porque los valores o clases a predecir son independientes entre sí (es decir, no existe ninguna relación entre ellas), en este grupo se puede encontrar los clasificadores multiclase (más de dos clases) y binarios (únicamente dos clases). Pero además de éstos existen los conocidos como problemas de clasificación jerárquica, donde las clases a ser predichas están organizadas en jerarquías, es decir no son independientes entre sí. La clasificación jerárquica resuelve problemas de clasificación donde la salida del algoritmo está definido sobre una taxonomía de clases (Silla & Freitas, 2011).

Debido a que existe mucha literatura en la cual se utiliza diferente terminología, en este trabajo de investigación nos basamos en los conceptos y taxonomía especificada por Carlos Silla y Alex Freitas en su publicación "Una encuesta sobre la clasificación jerárquica a través de diferentes dominios de aplicación" (Silla & Freitas, 2011). Esta trata de organizar la información relevante sobre los diferentes tipos de clasificadores jerárquicos.

2.2.1. Problemas de Clasificación Jerárquica

Como ya se mencionó los problemas de clasificación jerárquicos se diferencian de los problemas de clasificación plana en el hecho que las clases están organizadas bajo una estructura jerárquica o taxonómica.

Las taxonomías de clase son estructuras en forma de árbol definidas por el par ordenado $(C, \prec)$, donde la C es un conjunto finito que incluye todas de las clases de un dominio de aplicación y el símbolo $\prec$ representa la relación "IS-A": una clase c_i es un subtipo de la clase c_j, y lo representamos como $c_i \prec c_j$, donde c_i y c_j representa dos clases en diferentes posiciones dentro de la jerarquía. (Wu et al, 2005) define la relación IS-A con las siguientes propiedades:

- Ningún elemento puede ser mayor a la raíz del árbol.
- $\forall\, c_i, c_j \in C, if\ c_i \prec c_j\ then\ c_j \nprec c_i.$

- $\forall\, c_i \in C, c_i \prec c_i.$
- $\forall\, c_i, c_j, c_k \in C, c_i \prec c_j \; and \;\; c_j \prec c_k \; imply \; c_i \prec c_k.$

Las cuales permiten construir las taxonomías de clases, bien como estructuras arbóreas o bien como grafos acíclicos directos (DAG: Direct Acyclic Graph). La principal diferencia entre ambas representaciones es que en un DAG cada nodo interno puede tener uno o más ancestros directos, mientras que en la estructura en forma de árbol solamente tiene uno, como se puede observar en la Figura 3.

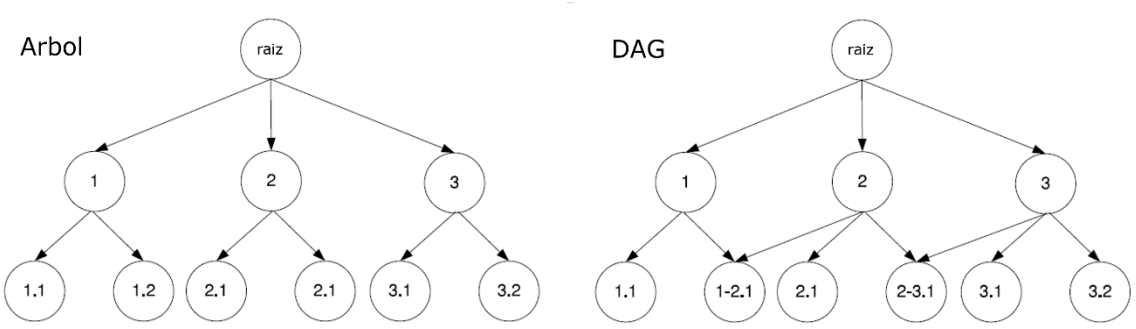


Figura 3. Tipos de estructuras jerárquicas.
(Bronoski, Silla, & Nievola, 2013)

En la jerarquía en forma de árbol (figura de la izquierda) todos los nodos intermedios y hojas solamente tienen un nodo ancestro, mientras que en la gráfica de la derecha un nodo intermedio puede relacionarse con más de un nodo ancestro. Por ejemplo, el caso del nodo 2–3.1 que es hijo de los nodos 2 y 3.

La implementación de clasificadores para estructuras DAG es más compleja que para estructuras en forma de árbol, ya que es más probable encontrarse con inconsistencias. El problema se da por ejemplo, en el caso del nodo 1-2.1 de la Figura 3 (derecha), que al tener como padres los nodos 1 y 2 es difícil determinar si este nodo es el resultado del camino Raíz->1->1-2.1 (por lo que le asignaríamos las clases 1 y 1-2.1) o el camino Raíz->2->1-2.1 (por lo que le asignaríamos las clases 2 y 1-2.1). Por lo tanto, es posible asignar a cada instancia múltiples caminos de predicción. Un camino es el resultado del recorrido que realiza el clasificador desde la raíz hasta obtener el nodo hoja buscado.

Los métodos de clasificación jerárquica se diferencian por la estructura que soportan bajo su análisis, y por la profundidad en el árbol a la que hacemos la predicción, ya que el método de clasificación jerárquica podría llegar hasta el nodo hoja o no. También es importante el enfoque del clasificador que es explicado en la siguiente sección.

2.3. Enfoques de Clasificación Jerárquica

Existe varios enfoques para el desarrollo de clasificadores jerárquicos. A continuación se hace una revisión de cada uno:

2.3.1. Enfoque de Clasificador Plano

Consiste en ignorar completamente la jerarquía de las clases y se realizan predicciones solamente sobre las clases de los nodos hojas, ignorando los nodos (clase) internos de la jerarquía. La desventaja se presenta en el hecho que el clasificador deberá distinguir entre una mayor cantidad de clases, sin explorar la información contenida en cada nivel de la jerarquía (Silla & Freitas, 2011).

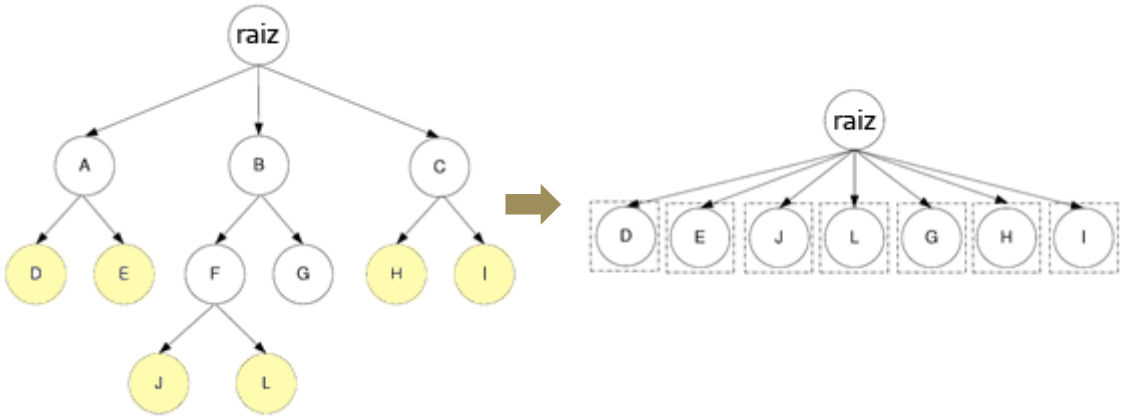


Figura 4. Clasificador plano aplicado a un problema de clasificación jerárquica. **(Bronoski, Silla, & Nievola, 2013)**

La Figura 4 (izquierda), muestra una jerarquía completa en la que los nodos amarillos (las hojas) son los valores a predecir mientras que en la imagen de la derecha se muestra la taxonomía considerada por el enfoque del clasificador plano. Una vez aprendido el clasificador, se puede hacer uso de la jerarquía

inicial para, a partir de las hojas asignar las etiquetas de clase internas. Por ejemplo, siguiendo con la figura 4 si una instancia es clasificada como de clase D entonces también podemos asignarle la clase A ya que D es desciende de A.

2.3.2. Enfoque de Clasificador Local

Existen varios tipos de clasificadores locales los cuales se basan en la forma en que utilizan la información de los niveles de jerarquía y cómo construyen el clasificador. A continuación se presentan los tipos de clasificadores locales más habituales en la literatura:

2.3.2.1. Clasificador Local por Nodo

Consiste en entrenar un clasificador binario para cada nodo de la jerarquía con excepción del nodo raíz. La desventaja de esta metodología es que si el número de clases es muy grande se obtendrá una gran cantidad de clasificadores. Además el resultado puede ser inconsistente ya que no existe garantía de que se respete la jerarquía, dado que cada clasificador se entrena aisladamente sin tener en cuenta la estructura jerárquica. En la figura 5, las líneas puntuadas representan la existencia de un clasificador en un nodo, es posible observar que se ha entrenado un clasificador binario para cada nodo exceptuando la raíz, el nivel más bajo nos dirá si la el ejemplo pertenece o no a la clase.

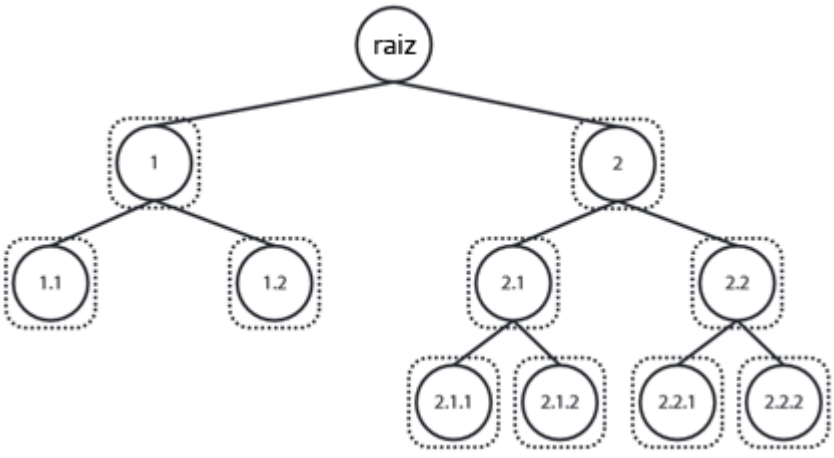


Figura 5. Clasificador local por nodo.

2.3.2.2. Clasificador Local por Nodo Padre

Consiste en entrenar un clasificador por cada nodo interno de la jerarquía de clases como un clasificador multiclase teniendo en cuenta únicamente las clases que están directamente relacionadas. Por lo general se utiliza un solo tipo de clasificador para todos los nodos internos, aunque esto podría variar por ejemplo al utilizar diferentes algoritmos para cada nivel. La elección en este caso del clasificador podría ser el que tenga el mayor indicador de exactitud. La desventaja con este método es que no puede ser utilizado con estructuras DAG ya que los conjuntos de entrenamiento local podrían ser altamente redundantes.

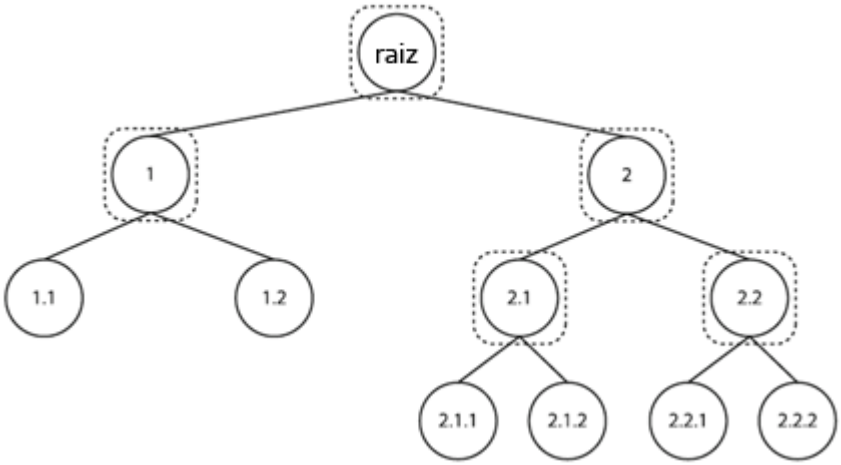


Figura 6. Clasificador local por nodo padre.

Como se puede observar en la figura 6, los nodos que presentan líneas punteadas son clasificadores multiclase. El proceso de predicción se realiza recorriendo el árbol de arriba hacia abajo y el nodo que se encuentra en un nivel superior al nodo hoja es el encargado de clasificar a cuál de las clases que representan los nodos hojas pertenece el ejemplo.

2.3.3. Enfoque de Clasificador Global

El clasificador global también se lo conoce como clasificador "big bag", el tamaño es menor que en comparación con los clasificadores locales. En este clasificador un único modelo de clasificación es creado a partir del conjunto de datos de entrenamiento. Tomando la jerarquía de clases como una sola

durante la ejecución del algoritmo de clasificación. La principal desventaja es la creciente complejidad en el desarrollo de estos clasificadores. Una vez creado el modelo este clasificador podría asignar una clase en cualquier nivel de la jerarquía.

2.3.4. Categorización de Algoritmos Jerárquicos

Es posible la formalización de los diferentes tipos de algoritmos a través de una tupla de 4 componentes: (Δ ; Ξ ; Ω ; Θ), donde:

- Δ dice si el algoritmo puede predecir etiquetas en uno o múltiples caminos de la jerarquía. Cuando el algoritmo puede asignar a cada instancia un camino se dice que es un algoritmo de predicción de camino simple, y si puede asignar a cada instancia múltiples caminos de etiquetas predichas se dice que es un algoritmo de predicción de múltiples caminos.
- Ξ representa la profundidad del algoritmo. Este puede llegar hasta el nodo hoja (MLNP: *Mandatory leaf-node prediction*) o puede asignar clases a cualquier nivel (NMLNP: *Non-mandatory leaf-node prediction*).
- Ω es la estructura taxonómica que soporta el algoritmo, como se vio anteriormente existen dos formas de árbol (T: *Tree*) o un grafo directo acíclicos (D: *DAG*).
- Θ representa la clasificación del algoritmo bajo la taxonomía propuesta. Según lo dicho anteriormente los posibles valores son: clasificador local por nodo (LCL: *Local Classifier per Nodo*), clasificador local por nodo padre (LCPN: *Local Classifier per Parent Node*) o clasificador global (GC: *Global Classifier*).

En el Anexo 1 se adjunta una tabla con los algoritmos de clasificación jerárquica caracterizados por los parámetros antes mencionados.

2.4. Medidas de Rendimiento

La evaluación de clasificadores se puede realizar a través de la utilización de varias medidas de rendimiento. Esto depende del tipo de problema y del objetivo del experimento. A continuación se realizará una revisión de las principales medidas de evaluación de rendimiento y validación, estas medidas serán utilizadas para la evaluación de los experimentos en los capítulos posteriores.

2.4.1. Validación Cruzada

La validación cruzada es utilizada cuando se tiene un número limitado de datos y es necesario verificar que todos los datos están siendo representados. El primer paso es incluir casos que no aparecen en el conjunto de datos pero que son muy probables en la realidad. Una vez que los datos estén completos se puede aplicar la técnica de validación cruzada. El procedimiento de validación realiza una serie de repeticiones de entrenamiento y prueba con diferentes selecciones aleatorias de los conjuntos de datos a utilizar en cada repetición. Por lo general se realiza una división del conjunto de datos en 10 partes del mismo tamaño y se toman 9 para el entrenamiento y 1 para la prueba. Con cada iteración se calcula una tasa de error y finalmente se promedia para calcular el error estimado de todo el conjunto de datos. Experimentos en varios conjuntos de datos con diferentes técnicas han mostrado que 10 se acerca al número correcto de divisiones (Witten, Eibe, & Hall, 2011).

En la Figura 7, se puede visualizar como se realiza la segmentación de los datos los cuales son divididos en dos conjuntos de datos: uno para el entrenamiento y otro para la prueba. Como se puede observar los datos de prueba son siempre menores que los datos de entrenamiento y a través de cada iteración no se repite los mismos conjuntos de datos seleccionados.

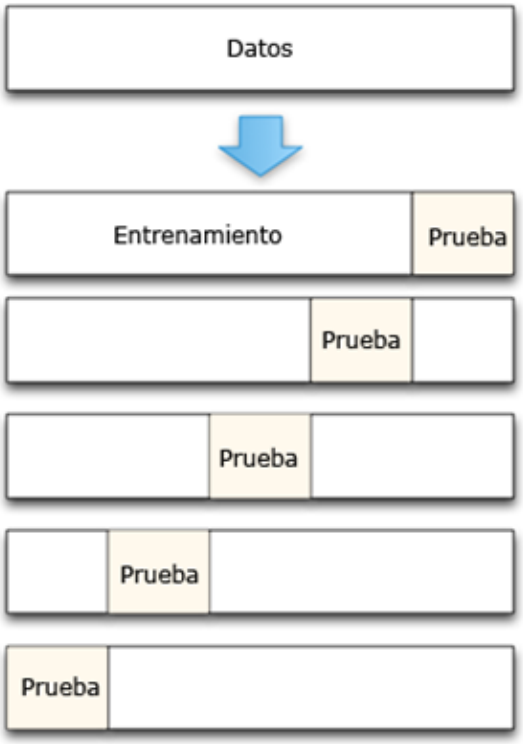


Figura 7. Ejemplo ilustrativo de una validación cruzada.
(Markham, 2015)

2.4.2. Matriz de Confusión

La matriz de confusión es una herramienta que nos permite visualizar en una tabla los aciertos y errores que produjo un modelo. Con una matriz de confusión es posible evaluar el funcionamiento de un clasificador, además nos permite saber en qué clases en particular se está cometiendo errores de predicción con mayor frecuencia. Cada columna de la matriz representa las predicciones por clase, mientras que cada fila representa la clase real.

La figura 8 muestra un ejemplo de una matriz de confusión generada por un sistema de clasificación de imágenes que permite distinguir entre imágenes que contienen un pato, un león o un mono. En este ejemplo el clasificador distinguió que 3 de las fotografías con patos contenían realmente un pato y cometió 1 error al predecir un pato como un león. Además se puede observar que este sistema tiene una pobre capacidad de distinguir entre monos y leones. Ya que confundió 3 monos con leones y 4 leones con monos.

		Clase Predicha		
		Pato	**León**	**Mono**
Clase Real	**Pato**	3	1	0
	León	1	5	4
	Mono	0	3	7

Tabla 1. Ejemplo de una matriz de confusión.

Para el caso de dos clases (generalmente denominadas como positiva/negativa, si/no, 1/0) la matriz de confusión puede representarse en forma de tabla en la que las celdas de la matriz representan falsos positivos, falsos negativos, verdaderos positivos y verdaderos negativos como se muestra a continuación:

		Predicho	
		Si	**No**
Real	**Si**	Verdadero Positivo (VP)	Falso Negativo (FN)
	No	Falso Positivo (FP)	Verdadero Negativo (VN)

Tabla 1. Componentes de una tabla de confusión.

Existen varias medidas para evaluar el rendimiento de un clasificador a partir de la tabla o matriz de confusión. A continuación se presentan algunos de los indicadores utilizados en este proyecto de investigación.

El acierto (Accuracy) mide el porcentaje de instancias correctamente clasificadas:

$$Accuracy\ (acierto) = \frac{VP + VN}{VP + VN + FP + FN}$$

Uno de los problemas con este indicador es que es insensible al costo de los errores de clasificación. Por ejemplo si quisiéramos diferenciar entre los correos spam y no spam, es más costoso perder un correo correcto que fallar en clasificar un correo como spam (Hernández Orallo, Ramirez Quintana, & Ferri Ramirez, 2004). También no se puede pensar que un error del 99% de exactitud es bueno ya que también depende del tamaño de la muestra, si la muestra es muy pequeña este valor no sería representativo. El acierto es el complementario de la tasa de error (error=1-accuracy).

Otros indicadores que son muy utilizados son:

El valor predictivo positivo nos dice el porcentaje de instancias que se predijeron como miembros de la clase positiva que son realmente de esa clase.

$$valor\ predictivo\ positivo\ (precision) = \frac{VP}{VP + FP}$$

El valor predictivo negativo nos dice el porcentaje de instancias que se predijeron como miembros de la clase negativa que son realmente de esa clase.

$$valor\ predictivo\ negativo\ = \frac{VN}{VN + FN}$$

La sensibilidad es la capacidad del modelo de predecir correctamente la clase positiva:

$$sensibilidad\ (recall) = \frac{VP}{VP + FN}$$

La especificidad es la capacidad del modelo de predecir correctamente la clase negativa:

$$especificidad = \frac{VN}{VN + FP}$$

2.4.3. Coeficiente Kappa

Este indicador nos permite evaluar el nivel de concordancia entre observadores. Es preferible a los otros índices incluye el porcentaje de acuerdo que se puede deberse al azar. En nuestro caso permite evaluar la relación que existe entre los valores reales y los valores predichos. El coeficiente Kappa es representado por K .

$$K = \frac{p_0 - p_e}{1 - p_e}$$

Siendo:

$$p_0 = \frac{numAciertos}{totalValoresPredicciones}$$

El $numAciertos$ es la suma de los valores que fueron predichos correctamente de la matriz de confusión y él $totalValoresPredicciones$ es la suma de todos los valores de la matriz de confusión.

$$P_e = \sum_{i=1}^{n} (p_{i1} \times p_{i2})$$

El valor de n representa el número de clases de la matriz de confusión, p_{i1} es la proporción de cada clase real con respecto al total de predicciones, calculada como la suma de los valores de una fila representada por i dividido para el total de la fila. El valor p_{i2} s la proporción de cada clase predicha con respecto al total de predicciones, es calculado como la suma de los valores de una columna representada por i dividido para el total de la columna. La fuerza del valor de concordancia se puede evaluar según los rangos que se especifican en la Tabla 2.

Coeficiente kappa	Fuerza de la concordancia
0,00	Pobre *(Poor)*
0,01 - 0,20	Leve *(Slight)*
0,21 - 0,40	Aceptable *(Fair)*
0,41 - 0,60	Moderada *(Moderate)*
0,61 - 0,80	Considerable *(Substantial)*
0,81 - 1,00	Casi perfecta *(Almost perfect)*

Tabla 2. Valoración del Coeficiente Kappa.

Por ejemplo, aplicando este indicador a nuestro ejemplo de la Tabla 1, el cálculo se realiza de la siguiente forma:

El primer paso es calcular el valor de la proporciones de la matriz de confusión tal y como se muestra en la Tabla 3.

		Clase Predicha			
		Pato	**León**	**Mono**	**Proporción**
Clase Real	**Pato**	3	1	0	4/24
	León	1	5	4	10/24
	Mono	0	3	7	10/24
	Proporción	4/24	9/24	11/24	24/24

Tabla 3. Aplicación de cálculo de estadístico Kappa.

$$p_0 = \frac{numAciertos}{totalValoresPredicciones} = \frac{3+5+7}{24} = 0.625$$

$$p_e = \left(\frac{4}{24} \times \frac{4}{24}\right) + \left(\frac{10}{24} \times \frac{9}{24}\right) + \left(\frac{10}{24} \times \frac{11}{24}\right) = 0.186$$

$$K = \frac{p_0 - p_e}{1 - p_e} = \frac{0.625 - 0.186}{1 - 0.186} = 0.539$$

2.4.4. Curvas ROC

La curva ROC es una gráfica en 2 dimensiones de un clasificador con la tasa de falsos positivos en el eje x y la tasa de verdaderos positivos en el eje y. Para el cálculo de esta curva es necesaria la proporción de verdaderos positivos, la cual es el porcentaje de positivos que nuestro clasificador acertó. Y

los falsos positivos podrían verse como las falsas alarmas donde nuestro clasificador predijo como positivos pero en realidad eran falsos (Foster Provost, 2013).

$$Proporción\,VP = \frac{VP}{VP + FN}$$

$$Proporción\,FP = \frac{FP}{FP + TN}$$

Dando un punto por cada clasificador se puede dibujar una curva como se muestra en la Figura 8. Aquí se puede observar que el mejor clasificador es el clasificador B ya que tiene el mayor porcentaje de aciertos y menor porcentaje de errores en la predicción.

También es posible dibujar una sola curva por clasificador usando un conjunto de ejemplo de entrenamiento. Para esto se debe obtener puntajes por cada ejemplo y variar los límites de decisión desde –∞ a +∞.

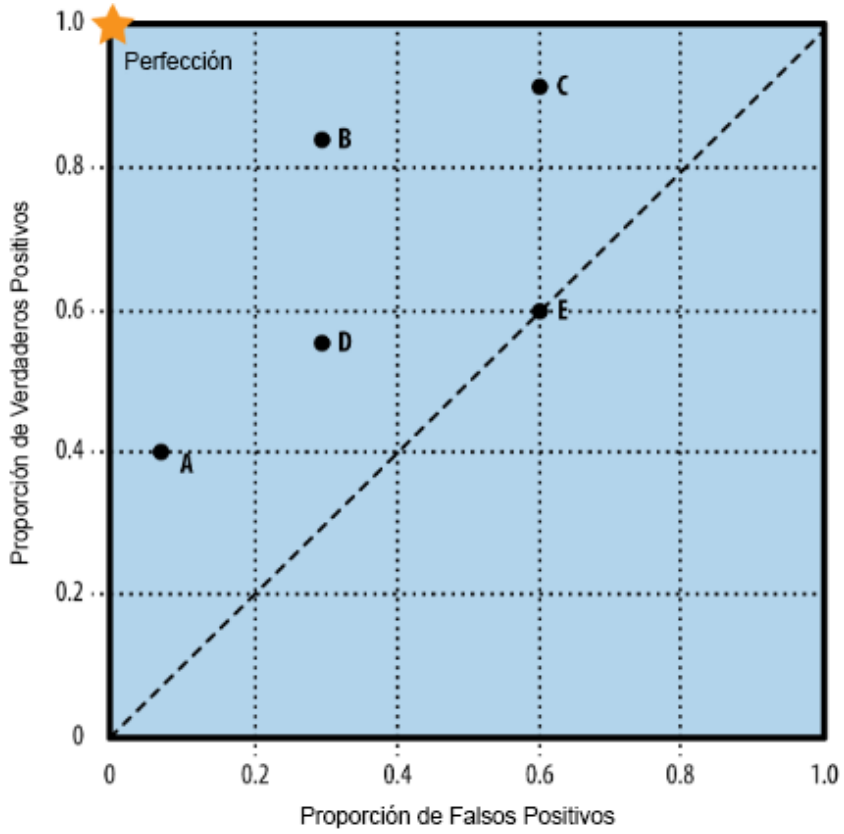


Figura 8. Representación de clasificadores y sus posiciones en el espacio ROC. (Foster Provost, 2013).

Otra estadística importante es el área bajo la curva ROC (AUC), que como su nombre lo indica cuantifica el área bajo la curva ROC, este valor está entre el rango de 0 y 1. Donde mientras más alto su valor mejor es el clasificador, ya que tiene una mayor capacidad de distinguir entre las clases.

CAPÍTULO

3 INFERENCIA AUTOMÁTICA DE JERARQUÌAS

3. Inferencia de Jerarquías

Este capítulo realiza una introducción a las funciones y procedimientos realizados para obtener la jerarquía de clases partiendo de los datos. Para ello se aplicarán distintas formas de transformar una matriz de confusión en una matriz de distancias o de similitudes entre clases y luego se aplicarán algoritmos de agrupamiento que nos permitan inducir las jerarquías.

3.1. Medidas de Distancias y Similitud

En el campo del reconocimiento de patrones un tema fundamental es la capacidad de reconocer similitudes entre las clases para inferir conocimiento. Esto se puede realizar a través de un análisis de clustering o agrupamiento de datos. El clustering nos permite tomar un conjunto de datos desorganizados y agruparlos en conjuntos con similares características para ello es necesario una adecuada función de distancia o similitud.

Por ejemplo, para calcular las distancias entre dos variables numéricas que se encuentran en las mismas unidades de medida se podría calcular simplemente la diferencia entre las variables o cualquier otra función de distancia definida entre valores reales. Cuando la variable es nominal, se deberá usar otro tipo de función que permita calcular la distancia entre los diferentes valores que pueda tomar la variable. En nuestros experimentos se busca medir la similitud que existe entre dos clases usando los datos de una matriz de confusión. Una vez obtenida la intensidad de similitud es posible obtener una jerarquía de clases que represente las relaciones entre las clases utilizadas.

A continuación se especifican las funciones de distancias utilizadas para medir la similitud entre dos clases.

3.1.1. Distancia Euclídea

Es la medida más antigua de distancia, se deriva del teorema de Pitágoras[3]. Se puede utilizar este concepto para medir las distancias entre dos puntos en un espacio de dos dimensiones como se muestra en la figura 9.

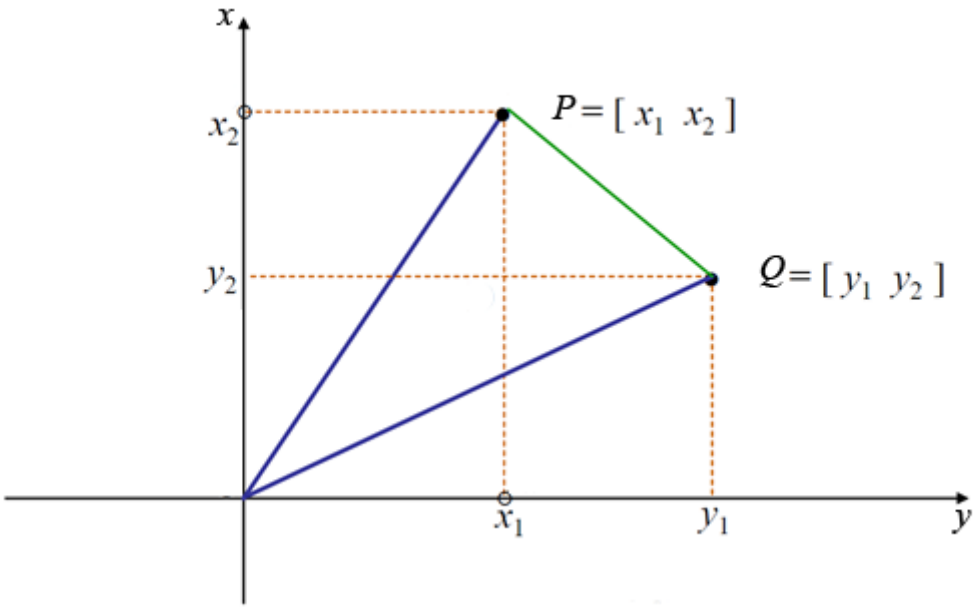


Figura 9. Distancia en entre dos puntos de en un plano.

La distancia entre dos puntos P y Q que son representados por sus coordenadas en un espacio bi-dimensional se calcula con la siguiente función:

$$d_{P,Q} = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$$

Es posible aplicar éste concepto a problemas que presentan un mayor número de dimensiones, por lo que se puede calcular la distancia entre objetos con n dimensiones, tal y como sigue:

$$d_{Euclidea}(\bar{p}, \bar{q}) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_n - q_n)^2} = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}$$

[3] Teorema de Pitágoras.- Establece que el cuadrado de la hipotenusa en un triángulo rectángulo es igual a la suma de los cuadrados de sus lados.

donde n representa el número de dimensiones, $\bar{p}$ y $\bar{q}$ son n-tuplas cuyas componentes son las coordenadas con respecto a cada uno de los n ejes. Este mismo concepto se puede utilizar para calcular la distancia entre dos clases, los valores de las filas de la matriz de confusión de las dos clases a comparar forman los vectores $\bar{p}$ y $\bar{q}$, y es posible utilizar la función presentada.

3.1.2. Distancia entre Conteos

Antes de calcular las distancias los valores deberán estar en las mismas unidades de medida. Por lo que es conveniente no trabajar con los valores directamente si no con sus valores estandarizados, tal y como se muestra a continuación:

$$valorEstandarizado = \frac{(ValorOriginal - Media)}{DesviacionEstandar}$$ [4]

Otra opción es normalizar todos los valores por ejemplo, dividiendo todos los valores de una componente por el valor máximo.

Si tenemos datos que representan conteos, por ejemplo la cantidad de manzanas en cada uno de cinco diferentes recipientes etiquetados a, b, c, d y e, y se encuentran en las mismas unidades de medida, no es necesaria una estandarización de los valores como se explicó anteriormente.

3.1.2.1. Distancia de Indice Bray-Curtis

Esta distancia ha sido utilizada por ecologistas, toma el nombre de J. Roger Bray y Jon T. Curtis, también conocida como distancia Manhattan. Cuantifica la disimilitud o distancia entre dos sitios, basado en el conteo de elementos en cada sitio. El término disimilitud es opuesto a la similitud, y se dice que una clase es similar a otra si la clase tiene una disimilitud de 0, por el contrario si tiene una disimilitud de 1 es porque las clases no tienen nada en común.

[4] Desviación Estándar.- Es una medida de dispersión para variables cuantitativas y de intervalo.

Si tenemos un vector de n elementos y queremos calcular la distancia con otro vector que posee la misma dimensión (Greenacre, 2008), se utiliza la siguiente función:

$$\boldsymbol{d_{Bray}}(\bar{p},\bar{q}) = \frac{\sum_{i=1}^{n}|p_i - q_i|}{\sum_{i=1}^{n} p_i + \sum_{i=1}^{n} q_i}$$

Donde $\bar{p}$ y $\bar{q}$ representan vectores con los valores que lo caracteriza, en nuestro caso los vectores son las filas de la matriz de confusión y lo que queremos comparar son las diferentes clases.

Para aplicar esta función primero se obtiene los dos vectores que representan las dos clases que se comparan. Siguiendo con la matriz de confusión mostrada en la Tabla 1, por ejemplo $\bar{p}$ puede ser la fila de la tabla que representa la clase Pato y el $\bar{q}$ es la fila correspondiente a la clase León:

$\bar{p} = [3,1,0]$

$\bar{q} = [1,5,4]$

Ahora se procede a aplicar la fórmula de la distancia Bray-Curtis:

$$\boldsymbol{d_{Bray}}(\bar{p},\bar{q}) = \frac{\sum_{i=1}^{n}|p_i - q_i|}{\sum_{i=1}^{n} p_i + \sum_{i=1}^{n} q_i}$$

$$\boldsymbol{d_{Bray}}(\bar{p},\bar{q}) = \frac{\sum_{n=1}^{n}|[3,1,0] - [1,5,4]|}{\sum_{n=1}^{n}[3,1,0] + \sum_{n=1}^{n}[1,5,4]}$$

$$\boldsymbol{d_{Bray}}(\bar{p},\bar{q}) = \frac{\sum_{n=1}^{n}[2,4,4]}{\sum_{n=1}^{n}[3,1,0] + \sum_{n=1}^{n}[1,5,4]}$$

$$\boldsymbol{d_{Bray}}(\bar{p},\bar{q}) = \frac{10}{4+10} = \frac{10}{14}$$

$$\boldsymbol{d_{Bray}}(\bar{p},\bar{q}) = \mathbf{7.714}$$

3.1.2.2. Distancia Chi-Cuadrado

La segunda distancia propuesta es la distancia Chi-Cuadrado la cual mide la disimilitud y es la base de lo que conoce como el análisis de correspondencia. Aunque tiene algunos años de antigüedad no es sino hasta que el francés Jean-Paul Bensecri realizó investigaciones sobre la implementación de esta técnica en casos reales (Greenacre, 2008).

Para el cálculo de la distancia Chi-Cuadrado primero se obtienen los vectores de las clases que se desean comparar, tal y como se realizó en el procedimiento anterior. Se deben normalizar los valores de las filas, lo cual se realiza sumando los valores del vector y dividiendo para los totales por fila. Los dos vectores obtenidos representan las clases que deseamos comparar, estos se representan con las variables $\bar{p}\ y\ \bar{q}$. La representación de los vectores normalizados es la siguiente $\bar{p}'\ y\ \bar{q}'$.

A continuación se presenta la función utilizada para el cálculo de esta distancia Chi:

$$\boldsymbol{d_{Chi}}(\bar{p}', \bar{q}') = \sqrt{\sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \frac{(\bar{p}'_i - \bar{q}'_i)^{\boldsymbol{2}}}{\overline{Promedio}}}$$

donde los vectores $\bar{p}'\ y\ \bar{q}'$ representan los vectores normalizados. El valor de la variable $\overline{Promedio}$ se calcula con la siguiente función:

$$\overline{Promedio} = \sum_{i=1}^{n} \frac{totAtr_i}{totalValores}$$

donde la variable $\overline{totAtr}$ representa el vector de los totales obtenidos por la suma de los valores de las columnas de la matriz de confusión. El valor de n es la cantidad total de filas de la matriz. La variable $totalValores$ es el valor total de la suma de todos los valores de la matriz de confusión para todas las clases.

Por ejemplo si aplicamos estos conceptos al ejemplo presentado en la Tabla 1, el resultado del cálculo obtenido del $\overline{Promedio}$ para este ejemplo es el siguiente:

$$\overline{Promedio} = \frac{[4,9,11]}{4+9+11} = \frac{[4,9,11]}{24} = \left[\frac{4}{24}, \frac{9}{24}, \frac{11}{24}\right] = [0.167,0.375,0.458]$$

Al usar este vector de promedios en la formula, incluimos en el cálculo los pesos que tiene cada valor sobre el total de valores. De la misma forma a continuación se procederá a calcular la distancia entre las clases.

Se obtiene los dos vectores que representan las dos clases que se compararan. En este ejemplo la variable $\bar{p}$ representa la fila de la clase Pato y el $\bar{q}$ representa la fila correspondiente a la clase Leon.

$\bar{p} = [3,1,0]$

$\bar{q} = [1,5,4]$

Luego obtenemos los vectores normalizados:

$$\bar{p}' = \frac{\bar{p}}{totalValores} = [3/4,1/4,0/4]$$

$$\bar{q}' = \frac{\bar{q}}{totalValores} = [1/10,5/10,4/10]$$

Ahora se procede a aplicar la fórmula de la distancia Chi-Cuadrada:

$$\boldsymbol{d_{Chi}}(\bar{p}', \bar{q}') = \sqrt{\sum_{i=1}^{n} \frac{(p'_i - q'_i)^2}{\overline{Promedio}}}$$

$$\boldsymbol{d_{Chi}}(\bar{p}', \bar{q}') = \sqrt{\sum_{i=1}^{n} \frac{([3/4,1/4,0/4] - [1/10,5/10,4/10])^2}{[0.167,0.375,0.458]}}$$

$$\boldsymbol{d_{Chi}}(\bar{p}', \bar{q}') = \sqrt{\sum_{i=1}^{n} \frac{([0.65, -0.25, 0])^2}{[0.167, 0.375, 0.458]}}$$

$$\boldsymbol{d_{Chi}}(\bar{p}', \bar{q}') = \sqrt{\sum_{i=1}^{n} \frac{[0.423, 0.063, 0.160]}{[0.167, 0.375, 0.458]}}$$

$$\boldsymbol{d_{Chi}}(\bar{p}', \bar{q}') = \sqrt{\sum_{i=1}^{n} [2.535, 0.167, 0.349]}$$

$$\boldsymbol{d_{Chi}}(\bar{p}', \bar{q}') = \sqrt{3.051}$$

$$\boldsymbol{d_{Chi}}(\bar{\boldsymbol{p}}', \bar{\boldsymbol{q}}') = \mathbf{1.747}$$

3.2. Clustering Jerárquico

El clustering jerárquico es un método de agrupamiento que busca construir jerarquías de los datos. Ya que en este proyecto de investigación buscamos construir jerarquías de los valores de las clases es necesario realizar una revisión del funcionamiento y las principales técnicas de clustering jerárquico.

3.2.1. Tipos de Clustering Jerárquico

Existen dos tipos de clustering jerárquico:

- **Aglomerativo:** Primeramente se asigna a cada observación un cluster único (singleton) y se empiezan a agrupar de dos en dos de acuerdo a una función de distancia. De esta forma dos clusters son fusionados en un cluster de nivel superior.
- **Divisible:** Al contrario que el anterior, esta metodología empieza con un único cluster que agrupa todas las observaciones y luego se divide en clusters más pequeños.

Típicamente el resultado del agrupamiento es un dendrograma como el que se muestra en la Figura 10. Para la inferencia automática de la jerarquía es necesario seleccionar una medida de distancia que nos ayude a agrupar o separar las clases.

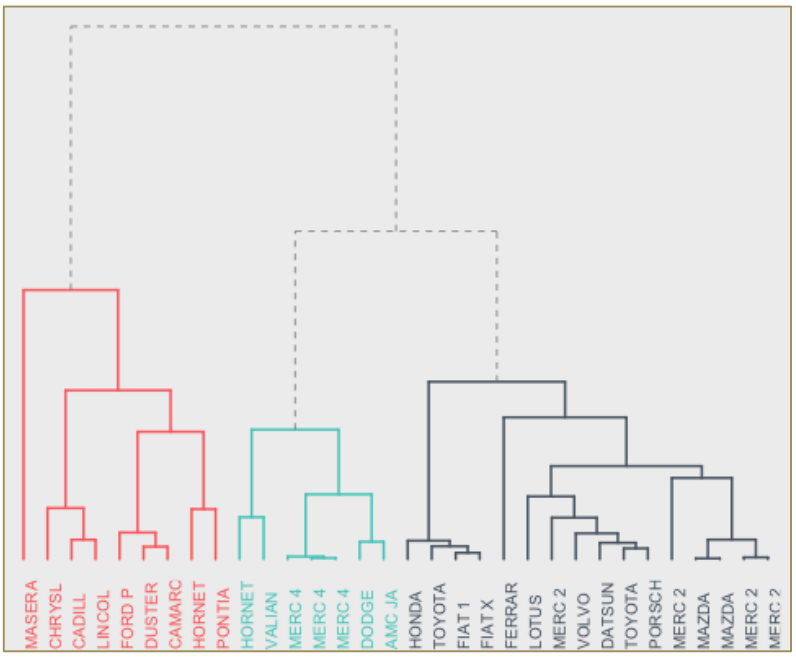


Figura 10. Visualización típica de un dendrograma.

Los dendrogramas nos ayudan a visualizar las relaciones que existen entre las clases, y son muy eficientes para representar gráficamente las relaciones entre clases que presentan una estructura jerárquica. A continuación se mencionan y describen cada una de las técnicas de clustering jerárquico conocidas.

Técnica de Enlace Simple

También conocida como la técnica del "vecino más próximo" (*Nearest Neighbor*). Define la distancia entre dos grupos como la distancia entre los 2 miembros, uno de cada grupo, más cercanos.

Técnica de Enlace Completo

Esta técnica define la distancia entre dos grupos como la distancia de sus 2 miembros más alejados.

Técnica de Promedio Simple

Esta técnica define la distancia entre dos grupos como el promedio entre las distancias de cada elemento en el grupo 1 con cada elemento del grupo 2, se dice que los dos grupos tienen una influencia igual sobre el resultado final.

Técnica de Centroide

Esta técnica define la distancia entre dos grupos como la distancia entre sus centroides[5].

Técnica de Mediana

Esta técnica define la distancia entre dos grupos como la distancia entre sus centroides. Los pesos asignados serán proporcionales al número de individuos en cada grupo.

Para este proyecto de investigación se realizarán experimentos con métodos tradicionales ya que serán suficientes para alcanzar los objetivos planteados.

3.3. Calculando Distancias y Jerarquías a Partir de la Matriz de Confusión.

Ya que en este proyecto de investigación nos interesa realizar la inferencia de jerarquías, hemos utilizado varias técnicas para calcular las distancias entre clases. Como ya hemos mencionado anteriormente nos interesa partiendo de la matriz de confusión obtener las distancias y así poder representar la jerarquía a través de un dendrograma. El procedimiento utilizado se puede describir en una serie de pasos que se muestran a continuación:

Se ha planteado un caso hipotético donde se obtiene una matriz de confusión como se muestra en la Tabla 4. Como se puede observar las posibles clases son a, b, c, d, e, f y g. Como se indicó en el Capítulo 2 cuando hablamos de inferencias automáticas de jerarquías a través de matrices de confusión, cada

[5] Centroide.- Centro de gravedad del vector promedio.

celda de la matriz c_{ij} representa cuantos ejemplos predichos como de clase c_j son de clase c_i. Por ejemplo en la Tabla 4, hay 23 instancias que son de clase c que se han predicho de clase d. Razonablemente se podría suponer que las instancias de clase c son muy parecidas a las instancias de clase d.

		Clase Predicha						
		a	**b**	**c**	**d**	**e**	**f**	**g**
Clase Real	**a**	10	2	7	1	0	0	0
	b	5	10	6	1	5	6	0
	c	5	4	1	23	3	3	0
	d	0	0	2	3	0	0	5
	e	0	0	4	6	1	10	2
	f	0	0	0	3	0	1	4
	g	0	0	0	0	0	1	2

Tabla 4. Ejemplo 2 de Tabla de confusión.

Por lo tanto vamos a generar una jerarquía de clases que represente las relaciones que se pueden observar en una matriz de confusión. A continuación aplicaremos a este ejemplo las diferentes técnicas para calcular las distancias.

3.3.1. Método Basado en Similitud

Se ha propuesto una técnica intuitiva inicial para el cálculo de las distancias entre clases. A continuación se describe el método de cálculo:

		Clase Predicha							
		a	**b**	**c**	**d**	**e**	**f**	**g**	**Total**
Clase Real	**a**	10	2	7	1	0	0	0	20
	b	5	10	6	1	5	6	0	33
	c	5	4	1	23	3	3	0	39
	d	0	0	2	3	0	0	5	10
	e	0	0	4	6	1	10	2	23
	f	0	0	0	3	0	1	4	8
	g	0	0	0	0	0	1	2	3

Tabla 5. Calculo de totales del método de distancia basado en similitud.

1. Se obtuvo el total de los valores de las filas de la matriz como se puede observar en la columna total de la Tabla 5, estos valores se utilizaron para realizar la normalización de los datos. Se dividió cada valor de la fila por el total de esa fila para obtener la matriz de valores normalizados. Cabe mencionar que si el caso se da que no existen valores en una fila se obtendría una división para cero. Esto se evita validando este comportamiento para que en estos casos se asigne un valor de similitud mínima entre las clases comparadas.

   La función utilizada para calcular la normalización de la tabla es:

$$M'_{xy} = \frac{M_{xy}}{\sum_{i=1}^{n} M_{xi}}$$

   Donde M representa la matriz de confusión y M' representa la matriz normalizada. En la tabla 6, se muestra la matriz normalizada calculada.

		Clase Predicha						
		a	**b**	**c**	**d**	**e**	**f**	**g**
Clase Real	**a**	0,50	0,10	0,35	0,05	0,00	0,00	0,00
	b	0,15	0,30	0,18	0,03	0,15	0,18	0,00
	c	0,13	0,10	0.03	0,59	0,08	0,08	0,00
	d	0,00	0,00	0,20	0,30	0,00	0,00	0,50
	e	0,00	0,00	0,17	0,26	0.04	0,43	0,09
	f	0,00	0,00	0,00	0,38	0,00	0.13	0,50
	g	0,00	0,00	0,00	0,00	0,00	0,33	0.67

Tabla 6. Normalización de la matriz mostrada en la Tabla 5.

2. A continuación se realiza la fusión de los valores que representan la relación entre dos clases. Esto se realiza sumando los valores M'_{xy} y M'_{yx} de las clases x e y. Por ejemplo el valor que representa la relación entre la clase "b" y la clase "a" se debe sumar al valor que representa la relación entre la clase "a" y la clase "b". El promedio de estos valores representan la similitud entre las clases. Donde 1 representa una similitud total y 0 que no existe confusión entre clases por lo tanto no hay ninguna similitud.

La función utilizada para calcular la similitud entre clases es:

$$similitudClase(x,y) = \frac{M'_{xy} + M'_{yx}}{2}$$

En la tabla 6, se muestra los valores de similitud calculados.

		Clase Predicha						
		a	**b**	**C**	**d**	**e**	**f**	**g**
Clase Real	**a**							
	b	0,252						
	c	0,478	0,284					
	d	0,050	0,030	0,790				
	e	0,000	0,152	0,251	0,261			
	f	0,000	0,182	0,077	0,375	0,435		
	g	0,000	0,000	0,000	0,500	0,087	0,833	

Tabla 7. Matriz de similitudes a partir de la matriz de la Tabla 6.

3. Finalmente se construye la matriz de distancias como se muestra en la tabla 8, restando 1 menos la matriz de similitud. Con estos valores es posible la creación de un dendrograma de distancias entre clases. Donde el valor máximo de distancia 1 significa que son totalmente diferentes, mientras del valor de 0 significa que son totalmente iguales.

   La función utilizada para calcular la distancia entre clases es:

$$ds(x,y) = 1 - similitudClase_{xy}$$

	a	**b**	**C**	**d**	**e**	**f**	**g**
a							
b	0,87						
c	0,76	0,86					
d	0,98	0,98	0,61				
e	1,00	0,92	0,87	0,87			
f	1,00	0,91	0,96	0,81	0,78		
g	1,00	1,00	1,00	0,75	0,96	0,58	

Tabla 8**.** Matriz de distancias obtenida al aplicar a la Tabla 4 el método basado en similitud.

A continuación aplicamos el algoritmo de agrupamiento jerárquico aglomerativo usando la técnica de enlace simple. Se obtiene el siguiente dendrograma:

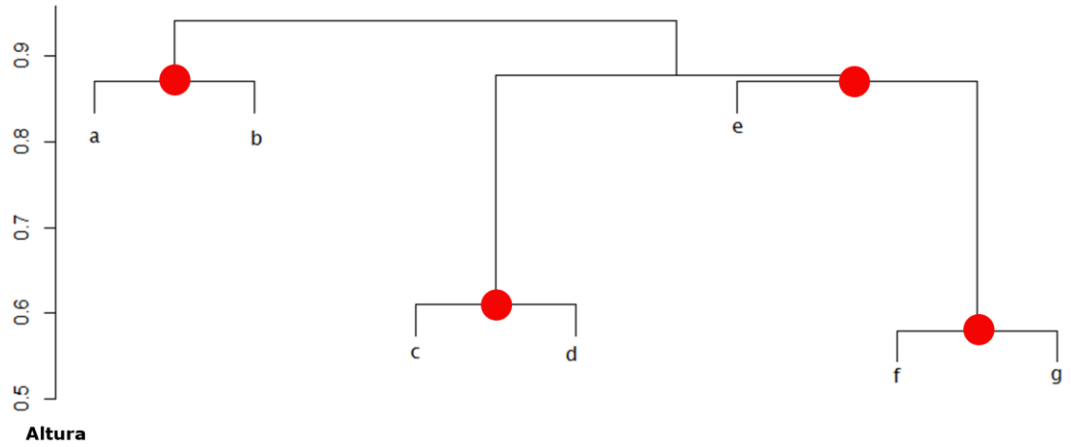


Figura 11. Dendrograma generado aplicando el método basado en similitud. (Usando la Matriz de Distancias Mostrada en la Tabla 8).

Este dendrograma puede verse como una jerarquía de clases en la que los clusters internos corresponden a los nodos internos de la jerarquía. Tal y como se muestra en la figura 11.

3.3.2. Método Aplicando Distancia Euclídea

El segundo método propuesto para la obtención de la matriz de distancias consiste en usar la distancia Euclídea desde la matriz de confusión. El procedimiento de aplicación se describe a continuación:

1. Se repite el paso 1 del método anterior para obtener la matriz con los valores normalizados.
2. A continuación se calcula la distancia entre dos clases tomando en cuenta que todas las predicciones de esa clase forman un vector de características. Para el cálculo de las distancias se utiliza la distancia Euclídea entre los vectores $\bar{p}\ y\ \bar{q}$. Tomando en cuenta que estos vectores representan los valores de las filas de la matriz de confusión para cada clase que se pretende comparar.

$$d_E(\bar{p},\bar{q}) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_n - q_n)^2} = \sqrt{\sum_{n=1}^{n}(p_i - q_i)^2}$$

Por ejemplo para encontrar la distancia Euclídea entre la clase a y la clase b que se muestra a continuación:

$\bar{p}' = [0.50,0.10,0.35,0.05,0.00,0.00,0.00]$

$\bar{q}' = [0.15,0.30,0.18,0.03,0.15,0.18,0.00]$

$$d_E(\bar{p}',\bar{q}') = \sqrt{\sum_{n=1}^{n}(p_i - q_i)^2}$$

$$d_E(\bar{p}',\bar{q}') = \sqrt{(0.50 - 0.15)^2 + (0.10 - 0.30)^2 + (0.35 - 0.18)^2}$$

$$\overline{+(0.05 - 0.03)^2 + (0.00 - 0.15)^2 + (0.00 - 0.18)^2 + (0.00 - 0.00)^2}$$

$$d_E(\bar{p}',\bar{q}') = \sqrt{0.12 + 0.04 + 0.03 + 0.00 + 0.02 + 0.03 + 0.00}$$

$$d_E(\bar{p}',\bar{q}') = 0.5$$

Si repetimos este procedimiento para obtener todos los valores de la matriz se obtiene al matriz de distancias como se muestra en la tabla 9.

	a	b	c	d	e	f	g
a							
b	0,50						
c	0,74	0,63					
d	0,77	0,70	0,63				
e	0,73	0,50	0,54	0,60			
f	0,87	0,74	0,58	0,25	0,56		
g	0,97	0,80	0,94	0,52	0,67	0,46	

Tabla 9. Matriz de distancias obtenida aplicando el método de distancia Euclídea a la matriz de la Tabla 4.

A continuación se aplicó el algoritmo de agrupamiento jerárquico aglomerativo usando la técnica de enlace simple. Se obtiene el siguiente dendrograma:

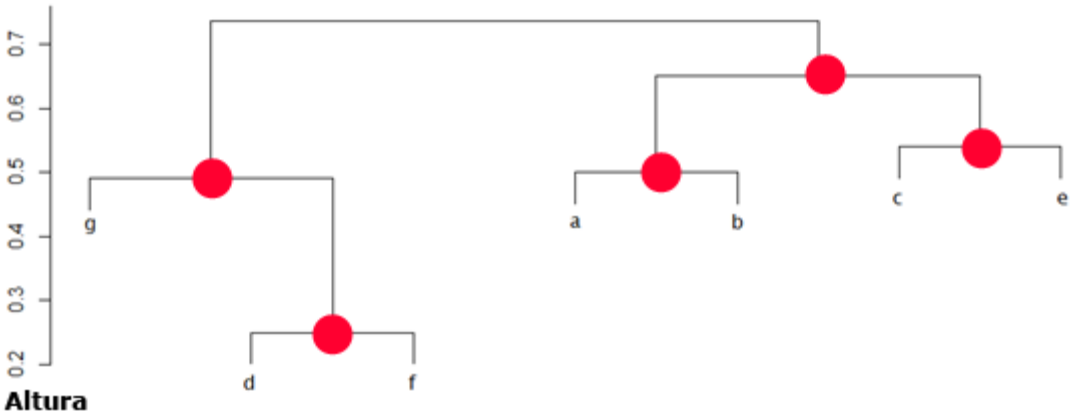


Figura 12. Dendrograma generado aplicando el método de Distancias Euclídea (usando la matriz de distancias mostrada en la Tabla 9).

Este dendrograma puede verse como una jerarquía de clases en la que los clusters internos corresponden a los nodos internos de la jerarquía. Tal y como se muestra en la Figura 12.

3.3.3. Método Aplicando Bryan-Curtis Index

El siguiente método consiste en aplicar Bryan-Curtis Index para el cálculo de distancias. El Algoritmo se describe a continuación:

1. En este apartado, el método parte de la matriz de confusión original, no se realiza una normalización de los valores previamente.
2. Se obtuvo el total de los valores de las filas de la matriz de la misma forma que se pudo observar en la tabla 4, estos valores se utilizaron para el cálculo del denominador de la formula.
3. Se calcula la distancia entre dos clases tomando en cuenta que todas las predicciones de esa clase forman un vector de características. Para el cálculo de las distancias se utiliza la siguiente función:

$$\boldsymbol{d_{Bray}}(\bar{p}, \bar{q}) = \frac{\sum_{i=1}^{n} |p_i - q_i|}{\sum_{i=1}^{n} p_i + \sum_{i=1}^{n} q_i}$$

Por ejemplo para calcular la distancia entra las clases a y b utilizando el mismo ejemplo que los métodos anteriores de cálculo, primero se toma cada fila de la matriz de confusión como un vector, tal y como se muestra a continuación:

$\bar{p} = [10,2,7,1,0,0,0]$

$\bar{q} = [5,10,6,1,5,6,0]$

A continuación se realiza la aplicación de la función para calcular distancias.

$\sum_{i=1}^{n}|p_i - q_i|$=[$|10-5|, |2-10|, |7-6|, |1-1|, |0-5|, |0-6|, |0-0|$]

$\sum_{i=1}^{n}|p_i - q_i|$=[|[5,8,1,0,5,6,0]=25

$$\sum_{i=1}^{n} p_i = [10+2+7+1+0+0+0] = 20$$

$$\sum_{i=1}^{n} q_i = [5+10+6+1+5+6+0] = 33$$

$$\boldsymbol{d_{Bray}(\bar{p},\bar{q}) = \frac{25.00}{20+33} = 0.47}$$

Si repetimos este procedimiento para obtener todos los valores de la matriz se obtiene al matriz de distancias como:

	a	b	c	d	e	f	g
a							
b	0,47						
c	0,69	0,53					
d	0,80	0,86	0,84				
e	0,77	0,57	0,65	0,58			
f	0,93	0,90	0,83	0,22	0,61		
g	1,00	0,94	0,95	0,69	0,77	0,45	

Tabla 10. Matriz de distancias obtenida aplicando el Método de Bryan-Curtis a la matriz de la Tabla 4.

A continuación se aplicó el algoritmo de agrupamiento jerárquico aglomerativo usando la técnica de enlace simple. Se obtiene el siguiente dendrograma:

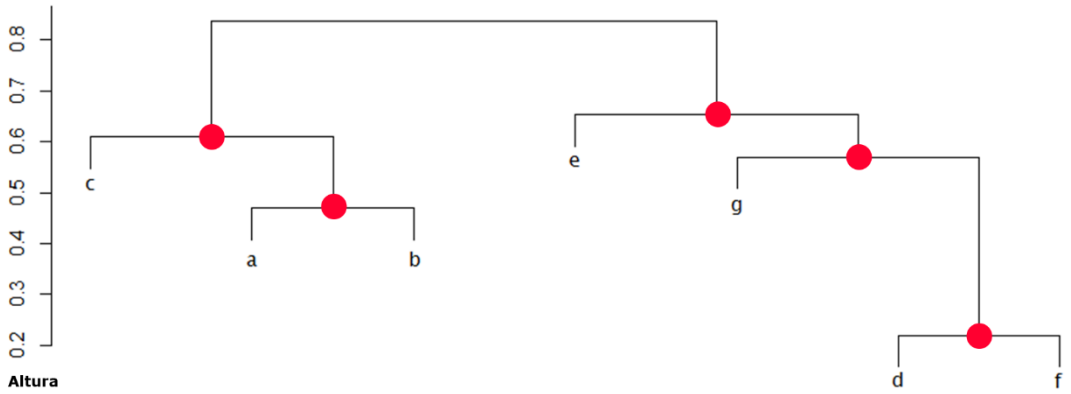


Figure 13. Dendrograma generado aplicando el método de Bryan-Curtis (usando la matriz de distancias mostrada en la Tabla 10)

Este dendrograma puede verse como una jerarquía de clases en la que los clusters internos corresponden a los nodos internos de la jerarquía. Tal y como se muestra en la figura 13.

3.3.4. Método Aplicando la Distancia Chi-Cuadrado

En este cuarto método se usa la función Chi-Cuadrado para el cálculo de las distancias. El Algoritmo se describe a continuación:

1. Se repite el paso 1 del método basado en Similitud para obtener la matriz normalizada. El valor de $\overline{totAtr}$ se obtuvo sumando los valores de las columnas de la matriz de confusión. El valor de $totalValores$ es la suma de todos los valores de la matriz de confusión. El cálculo del $\overline{Promedio}$ se muestra a continuación:

$$\overline{Promedio} = \sum_{i=1}^{n} \frac{totAtr_i}{totalValores}$$

Por ejemplo, para la matriz de la Tabla 4

$$\overline{Promedio} = \frac{[20,16,20,37,9,21,13]}{136} = [0.15,0.12,0.15,0.27,0.07,0.15,0.10]$$

2. A continuación se utiliza la fórmula para el cálculo de las distancias Chi-Cuadrado para encontrar la distancia entre las clases. Se parte de los valores normalizados tal y como se muestra:

$$\bar{p}' = [0.50,0.10,0.35,0.05,0.00,0.00,0.00]$$
$$\bar{q}' = [0.15,0.30,0.18,0.03,0.15,0.18,0.00]$$

$$\boldsymbol{d_{Chi}}(\bar{p}', \bar{q}') = \sqrt{\sum_{i=1}^{n} \frac{(\bar{p}'_i - \bar{q}'_i)^2}{Promedio}}$$

$$\boldsymbol{d_{Chi}}(\bar{p}', \bar{q}') = \sqrt{\frac{(0.50 - 0.15)}{0.15} + \frac{(0.10 - 0.30)}{0.12} + \frac{(0.35 - 0.18)}{0.15}}$$

$$\overline{\frac{(0.05 - 0.03)}{0.27} + \frac{(0.00 - 0.15)}{0.07} + \frac{(0.35 - 0.18)}{0.15} + \frac{(0.00 - 0.00)}{0.10}}$$

$$\boldsymbol{d_{Chi}}(\bar{p}', \bar{q}') = \sqrt{1.93} = \mathbf{1.389}$$

Si repetimos este procedimiento para obtener todos los valores de la matriz se obtiene al matriz de distancias como:

	a	b	c	d	e	f	g
a							
b	1,39						
c	1,69	1,35					
d	2,19	2,09	1,86				
e	1,87	1,34	1,29	1,75			
f	2,39	2,14	1,76	0,63	1,64		
g	2,83	2,51	2,58	1,27	2,02	1,04	

Tabla 11. Matriz de distancias obtenida aplicando el Método Chi-Cuadrado a la matriz de la Tabla 4.

A continuación se aplicó el algoritmo de agrupamiento jerárquico aglomerativo usando la técnica de enlace simple. Se obtiene el siguiente dendrograma:

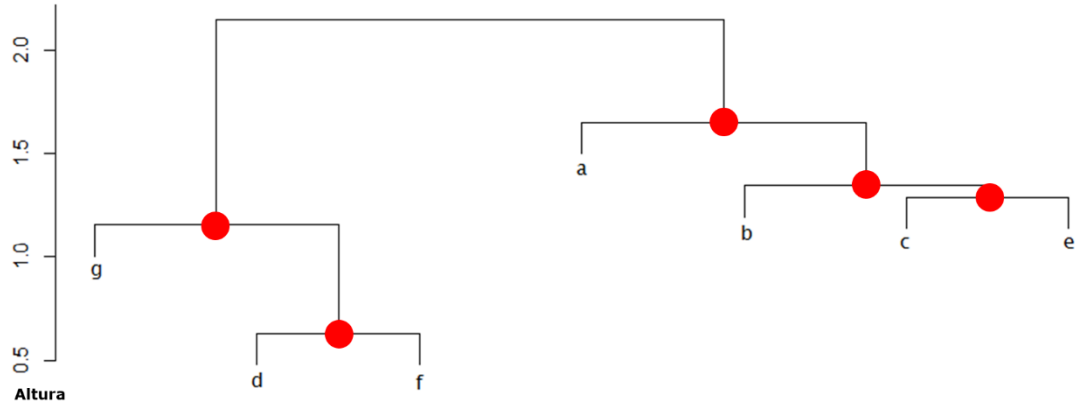
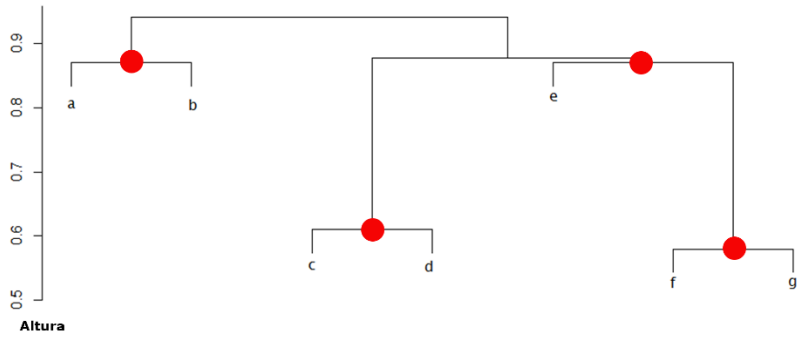


Figure 14. Dendrograma generado aplicando el método de Chi-Cuadrado (usando la matriz de distancias mostrada en la Tabla 11)

Este dendrograma puede verse como una jerarquía de clases en la que los clusters internos corresponden a los nodos internos de la jerarquía. Tal y como se muestra en la Figura 14.

3.4. Comparación de la Jerarquía Generadas

En este último apartado del capítulo, analizamos las diferencias entre las jerarquías generadas en función del método utilizado

Método	Jerarquía de Clases
Derivado de Similitud	

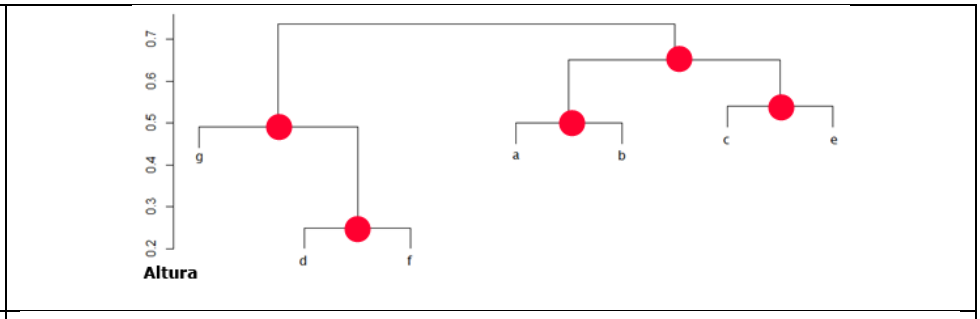
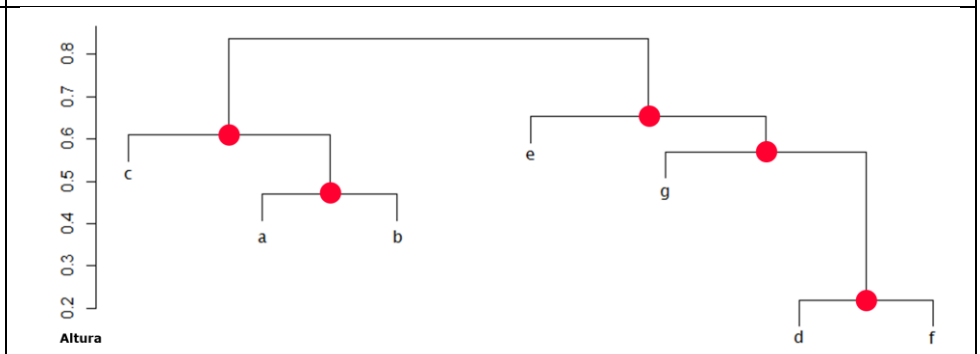
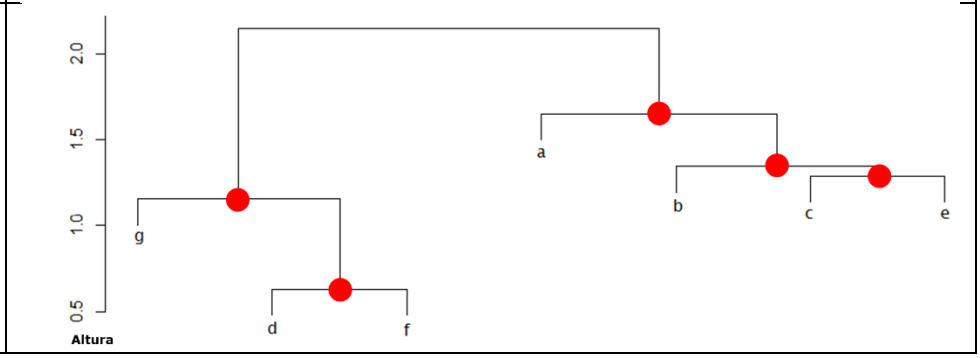
Euclidea	
Bray-Curtis	
Chi-Cuadrado	

Tabla 12. Jerarquías obtenidas por métodos de distancia.

Como se puede observar las jerarquías difieren no solo en el número de nodos internos (4 ó 5) sino también en su estructura. Así, por ejemplo, la clase c es próxima a la clase d en la primera jerarquía generada, mientras que es próxima a la e en la segunda y cuarta jerarquías, y próxima a las clases a y b en la tercera jerarquía. También se observa que aunque las jerarquía 3 y 4 se parecen en cuanto a su estructura, difieren en las clases con las que se etiquetan las hojas. En resumen, tenemos diferentes jerarquías con las que evaluar cómo se comportan los métodos de clasificación jerárquicos que presentamos en el siguiente capítulo.

CAPÍTULO

4 EXPERIMENTOS

4. Experimentos

Este capítulo utilizamos las jerarquías inferidas de acuerdo a los métodos descritos en el capítulo anterior, para aprender clasificadores jerárquicos. En concreto usamos el método de clasificación jerárquica local por nodo padre y compararemos los resultados con un clasificador plano que usaremos como referencia.

4.1. Especificaciones del Lenguaje Utilizado

Para el desarrollo de los métodos, clasificadores y demás scripts utilizados en la carga de datos, creación de métodos para el cálculo de distancias y en la creación de los clasificadores jerárquicos se utilizó R como lenguaje de programación. El usar R en este proyecto tiene las siguientes ventajas:

- Tiene un soporte continuo de toda una comunidad de desarrolladores.
- Código abierto y sin ningún precio.
- Variedad de librerías utilizadas en cálculos estadísticos, minería de datos, generación de gráficas, etc.
- Presenta un entorno de desarrollo completo.

La versión de R utilizada en este proyecto es la versión 3.2.0.

4.2. Especificaciones del Computador de Desarrollo

Se realizó las pruebas sobre una computadora personal con las siguientes características:

- Windows 8.1 Pro
- Procesador Intel Core i7-3770
- Memoria Ram: 8 GB
- Sistema Operativo de Windows x64

4.3. Librerías Utilizadas

Se utilizaron las siguientes librerías externas:

Caret versión 6.0-52.- Esta librería es un paquete dedicado a la creación de modelos predictivos. Caret tiene funcionalidades que permiten la división de datos, pre-procesamiento, visualización, optimización de modelos, etc. Esta librería tiene extensa gama de modelos integrados como son: J48, Random Forest, k-Nearest Neighbors, Regresion Lineal, Neural Network, etc.

Dendextend versión 1.1.0.- Ofrece un conjunto de funcionalidades extendidas al momento de crear dendrogramas, permite entre otras cosas visualizar y comparar árboles de clustering jerárquico. En particular este paquete permite realizar recorridos a través de un árbol jerárquico lo que fue necesario para la creación de los clasificadores utilizados en los experimentos.

Plyr versión 1.8.3.- Es un conjunto de herramientas utilizado para resolver problemas comunes, que entre otras cosas permite crear vectores de altas dimensiones, resumir datos, etc.

AUC versión 0.3.0.- Esta librería incluye funciones para el cálculo del área bajo la curva de sensibilidad, especificidad, precisión y bajo *receiver operating characteristic curve* (ROC). También incluye funciones para visualización de las curvas.

Ggplot2 versión 1.0.1.- Paquete para generar gráficas y visualizaciones a partir de datos.

4.4. Esquema del Experimento

El proceso de ejecución de los experimentos se desarrolla en tres etapas claramente definidas. Primeramente se debe definir las fuentes de datos que se utilizará en los experimentos. En segundo lugar es necesario realizar la creación del clasificador plano a partir del cuya matriz de confusión se va a inducir la jerarquía de clases.

En tercer lugar se creará un clasificador jerárquico para cada una de las jerarquías inducidas. Finalmente se realizará una evaluación y comparación de los clasificadores planteados. Los diferentes pasos del proceso se los puede visualizar en el diagrama de la Figura 15, donde se muestra el proceso completo de experimentación. A continuación se describen cada una de estas etapas en más detalle.

4.4.1. Selección de la Fuente de Datos

Como se puede apreciar en la Figura 15, primero es necesaria la selección de fuentes de datos para la realizar los experimentos. Las fuentes de datos seleccionadas para los experimentos deben cumplir con los siguientes requerimientos mínimos para garantizar los resultados deseados.

- Tener una cantidad razonable de ejemplos.
- Ser una fuente de datos confiable.

Las bases de datos seleccionadas para realizar los experimentos se muestran en la Tabla 13.

Nombre del Conjunto de Datos	Nro. Instancias	Nro. Atributos	Nro. Clases	Instancias Entrenamiento	Instancias Prueba
Dermatology	358	34	7	179	179
Forest Type	326	27	4	163	163
Letters Recognition	20000	16	6	10000	10000

Tabla 13. Bases de datos seleccionadas para el experimento.

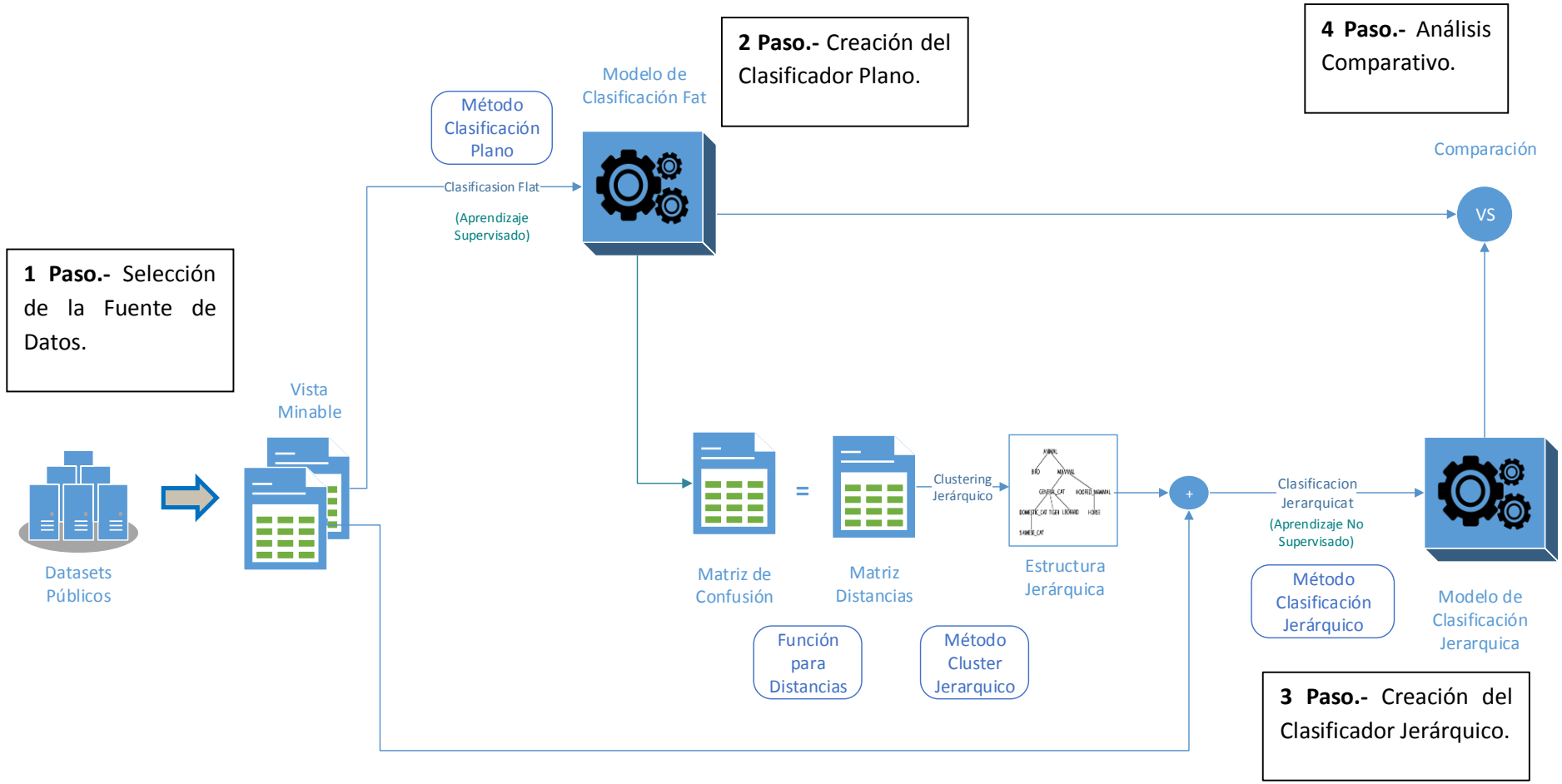


Figure 15. Proceso de Desarrollo de Experimentos

Se ha seleccionado estos conjuntos de datos ya que son multivariables, presentan una cantidad razonable de ejemplos de entrenamiento y test, además provienen de repositorios de investigación confiables, como se indica en la Sección 4.4.1.

4.4.2. Creación de un Clasificador Plano

La construcción de un clasificador plano se realiza partiendo de los datos de entrenamiento. Se debe seleccionar un algoritmo de clasificación para la creación del modelo, este procedimiento se puede visualizar en la Figura 16.

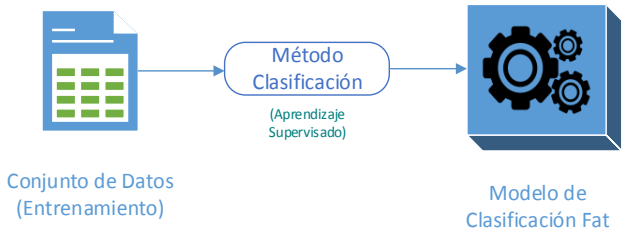


Figure 16. Proceso de Creación de Clasificador Plano

El algoritmo de clasificación seleccionado para ser utilizado para crear el clasificador plano fue el modelo J48, que es un modelo de tipo árbol de decisión.

4.4.3. Creación de los Clasificadores Jerárquicos

La construcción de un clasificador jerárquico se realiza normalmente partiendo del conjunto de datos y una estructura jerárquica. La taxonomía por lo general es conocida por el usuario o es obtenida de bases de datos ontológicas, como por ejemplo lo es "WordNet[6]". En esta investigación ya se ha realizado el proceso de inferir las jerarquías de clase por lo tanto estamos listos para construir el modelo de clasificación jerárquico.

[6] WordNet.- Es una base de datos léxica del inglés, que permite representarla a través de conceptos y sus relaciones.

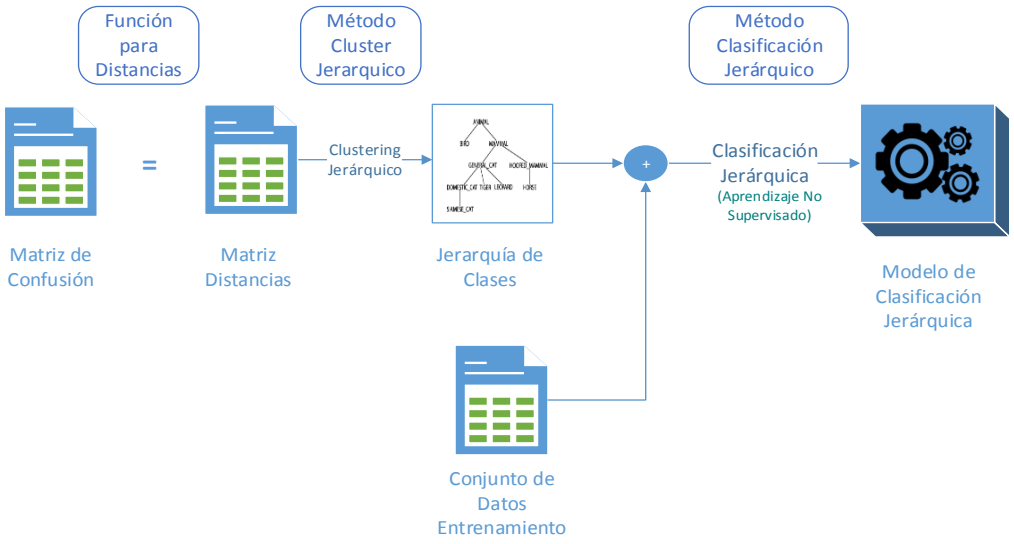


Figure 17. Proceso de Creación de Clasificador Jerárquico

Como se puede ver en la Figura 17, una vez creado el clasificador plano en el paso anterior, podemos obtener la matriz de confusión de los resultados obtenidos. Con esta información es posible a través de los métodos de distancias estudiados en el Capítulo 3, obtener la matriz de distancias entre clases. Al aplicar un agrupamiento jerárquico a esta matriz de distancias es posible obtener la jerarquía de clases, tal y como se explicó en el Capítulo 3. Para la generación del clasificador jerárquico se utiliza la información de la jerarquía y el conjunto de datos de entrenamiento para crear el clasificador. En este paso es necesario decidir el método de agrupamiento y el tipo de modelo que se utilizará en la construcción del clasificador jerárquico.

4.4.4. Comparación de Clasificadores

Se realizará la evaluación de los clasificadores creados a través de los siguientes indicadores.

- Acurracy e Intervalos de Confianza.
- Estadístico Kappa.
- Área Bajo la Curva ROC.
- Tiempo de predicción de los datos de prueba.

Como ya se mencionó en los capítulos anteriores estas medidas permiten evaluar el funcionamiento de los clasificadores generados. Con lo cual es posible analizar si el aplicar clasificación jerárquica (usando distintas jerarquías obtenidas a partir de los datos) mejora los resultados en relación a los obtenidos si se usan clasificadores planos.

4.5. Desarrollo de los Experimentos y Resultados Obtenidos

La ejecución de los experimentos se realizó primero fijando los parámetros de ejecución y luego realizó la ejecución para las 3 bases de datos seleccionadas.

4.5.1. Parámetros del Experimento

Los experimentos realizados a las tres bases de datos fueron realizados según los parámetros mostrados en la Tabla 14.

Parámetros del Experimento	
Atributos	**Valores**
Método de Clasificación Plano	J48 Árbol de Decisión
Función para Distancias	Propio, Euclídea, Bryan-Curtis, Chi-Cuadrado
Método de Agrupamiento Jerárquico	Técnica de Promedio Simple (Librería HCLUST)
Método de Clasificación Jerárquico	Clasificador Local por Nodo Padre

Tabla 14. Parámetros del experimento.

4.5.1.1.1. Experimento 1 "Forest Type"

Este experimento se realizó sobre la base de datos "Forest Type" (Tipo de Bosque), la cual se obtuvo del repositorio de aprendizaje automático de la Universidad de California (UCI). El objetivo de este conjunto de datos es el mapeo de diferentes tipos de bosques basados en sus características espectrales en longitudes de onda infrarroja. Esta salida puede ser utilizar para identificar o cuantificar los ecosistemas dados por los bosques.

Las clases que serán predichas son:

- S=Sugi (monte)
- H=Hinoki (monte)
- D=Mixed Deciduous (monte)
- O=Otros (no monte)

A continuación se puede observar los resultados obtenidos del experimento:

4.5.1.1.2. Resultados del Experimento 1

Siguiendo el paso 1 de nuestro esquema de experimento, al construir el modelo de clasificación plano se obtuvieron los resultados que se muestran en la Figura 18.

	D	H	O	S
D	70	0	6	5
H	0	29	1	12
O	15	0	38	0
S	20	9	1	119

Figura 18. Matriz de confusión con clasificador plano J48 para "Forest Type".

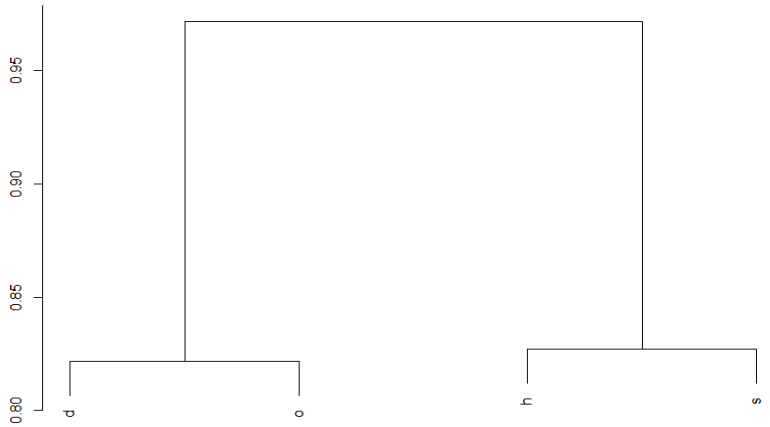
Como puede observarse, 256 ejemplos fueron clasificados correctamente por el modelo. Además se encontró que 69 fueron errores en los que el clasificador seleccionó una clase por otra. De la visualización y análisis de esta matriz se puedo reconocer a simple vista varios clústers o grupos, tal y como se muestra en la figura 19.

	D	H	O	S
D		0	6	5
H	0		1	12
O	15	0		0
S	20	9	1	

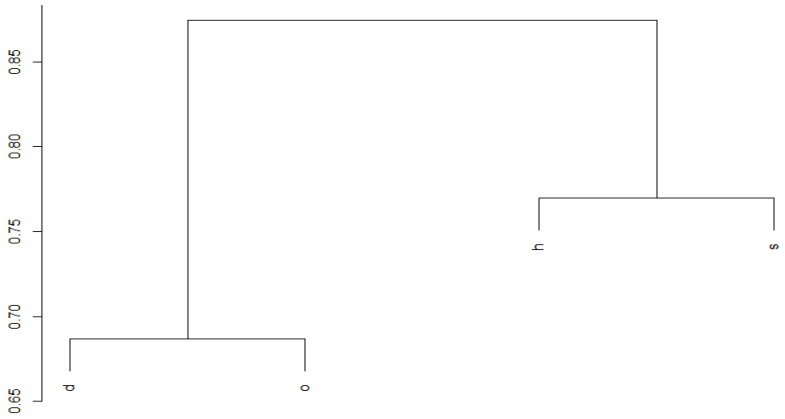
Figure 19. Clúster en Matriz de Confusión para "Forest Type" Dataset.

Luego se procedió a inferir la jerarquía de clases partiendo de los valores de la matriz de confusión. A continuación se construyeron las jerarquías de clases, aplicando los métodos de distancias ya estudiados y clustering jerárquico con el método de promedio simple.

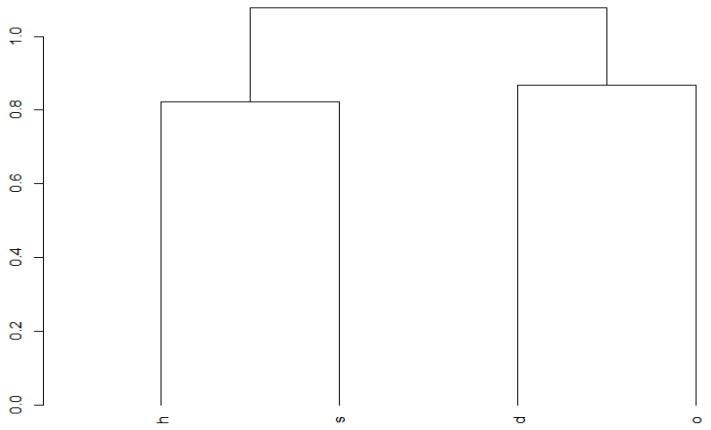
Las jerarquías generadas se presentan en la Figura 20, en ellas se puede apreciar las diferencias entre los distintos métodos de distancias.



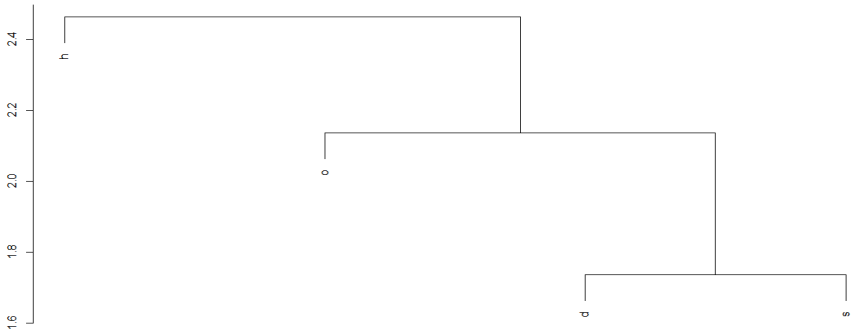
Jerarquía de clases por el método de similitud.



Jerarquía de clases por el método Bryan-Curtis.



Jerarquía de clases por el método Euclídea



Jerarquía de clases por el método Chi-Cuadrado

Figure 20. Gráficas de Jerarquías para el Conjunto de Datos "Forest Type".

Al usar los métodos de derivación de similitud, Bryan-Curtis y Euclídea se obtuvieron los mismos resultados de jerarquía. Por un lado se agrupó el tipo de monte "H" y "S" , y por otro se agrupó los tipos "D" y "O". Como se observó al dibujar los clusters sobre la tabla las relaciones entre las clases "D" y "H", así como las clases "O" y "H" son muy poco probables por lo que no se esperaría verlas agrupadas dentro de la misma rama. Además se puede observar que la distribución de la jerarquía Chi-Cuadrado es diferente a las demás, presenta una sola rama que se abre con una distribución uniforme hacia abajo.

Se procedió a la construcción del modelo de clasificación jerárquico con los datos de las jerarquías. A continuación se aplicó éste clasificador al conjunto de datos de prueba y se obtuvo los siguientes indicadores:

	Plano	Jerárquico			
		Similitud	Euclídea	Bryan Curtis	Chi - Cuadrado
Acuracy	0,788	0,803	0,803	0,803	0,791
95% CI	0,7392 - 0,8309	0,7556 - 0,8449	0,7556 - 0,8449	0,7556 - 0,8449	0,7424 - 0,8337
KAPPA	0,692	0,716	0,716	0,716	0,693
AUC	0,723	0,712	0,712	0,712	0,753
Tiempo Predicción (segundos)	0,5	17,0	17,0	17,0	28,0

Table 15. Resultados del rendimiento obtenido por los clasificadores para "Forest Type" DataSet.

Del rendimiento obtenido por los clasificadores utilizados se puede ver que sus indicadores mejoraron los resultados del clasificador plano, ya que los clasificadores que usaron distancia basada en Similitud, Euclídea y Bryan-Curtis han mostrado mejoría en sus predicciones. El accuracy es mayor que el clasificador usado de referencia. Los valores de los mencionados clasificadores son iguales por que al inducir las jerarquías de clases los tres presentaron la misma estructura.

Con relación al indicador KAPPA los clasificadores por Similitud, Euclidea y Bryan-Curtis obtuvieron los mejores resultados, mientras que el clasificador plano tiene el menor grado de concordancia entre las clases predichas y las reales.

Además se puede observar de la Figura 21, el área bajo la curva del clasificador Chi-Cuadrado es mayor que los demás clasificadores, por lo tanto este clasificador tiene una mayor capacidad de distinguir entre clases en esta base de datos.

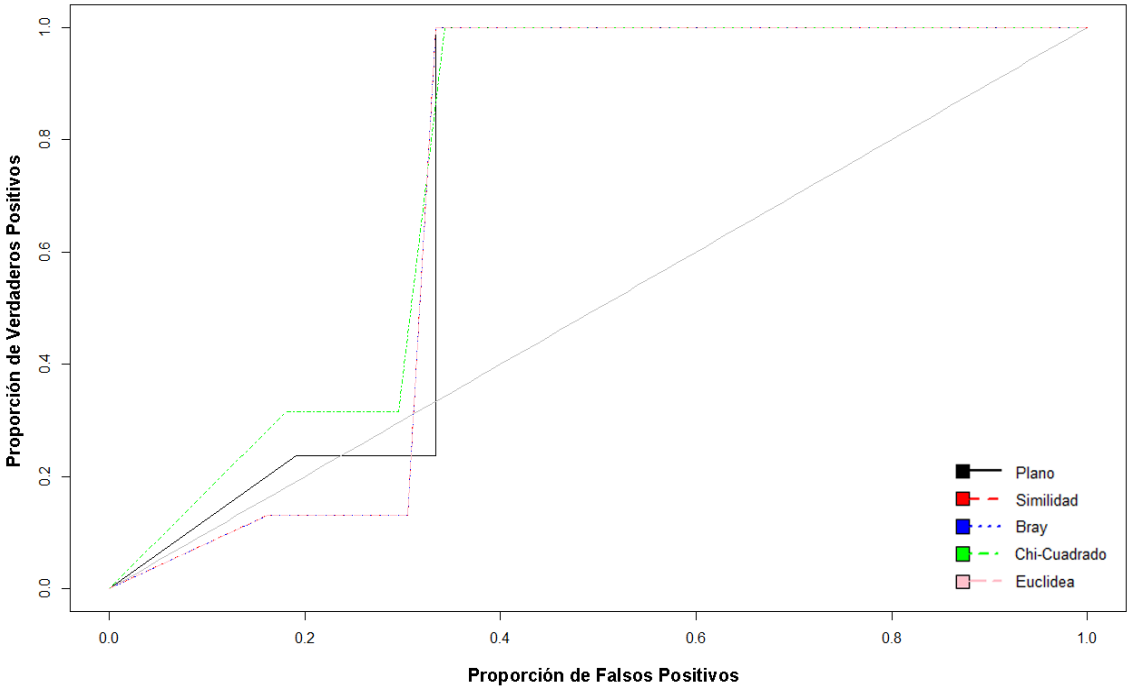


Figure 21. Análisis ROC de clasificadores para "Forest Type" DataSet.

4.5.1.2. Experimentos 2 "Dermatology"

Este experimento se realizó sobre la base de datos "Dermatology". La cual se obtuvo del repositorio de Knowledge Extraction Evolutionary Learning (KEEL).

Este conjunto de datos contiene un conjunto de 34 atributos enteros que representa características histopatológicas de un paciente. El objetivo es que se pueda realizar un diagnóstico diferencial de las enfermedades eritemato escamosas ya que es muy difícil realizarlo manualmente. Las enfermedades de este tipo pueden ser agrupadas por las siguientes clases:

- 1=Psoriasis
- 2=Seboreic dermatitis
- 3=Lichen planus
- 4=Pityriasis rosea
- 5=Cronic dermatitis
- 6=Pityriasis rubra pilaris

A continuación se pueden observar los resultados obtenidos del experimento realizado.

4.5.1.2.1. Resultados del Experimento 2

Efectuamos el paso 1 de nuestro esquema de experimento para construir el modelo de clasificación plano y obtuvimos los siguientes resultados:

	1	2	3	4	5	6
1	52	0	0	0	0	0
2	3	29	1	1	0	0
3	0	0	32	0	0	0
4	0	0	2	23	0	0
5	0	0	0	0	24	0
6	0	1	0	0	0	10

Figura 22. Matriz de confusión con clasificador plano J48 para "Dermatology".

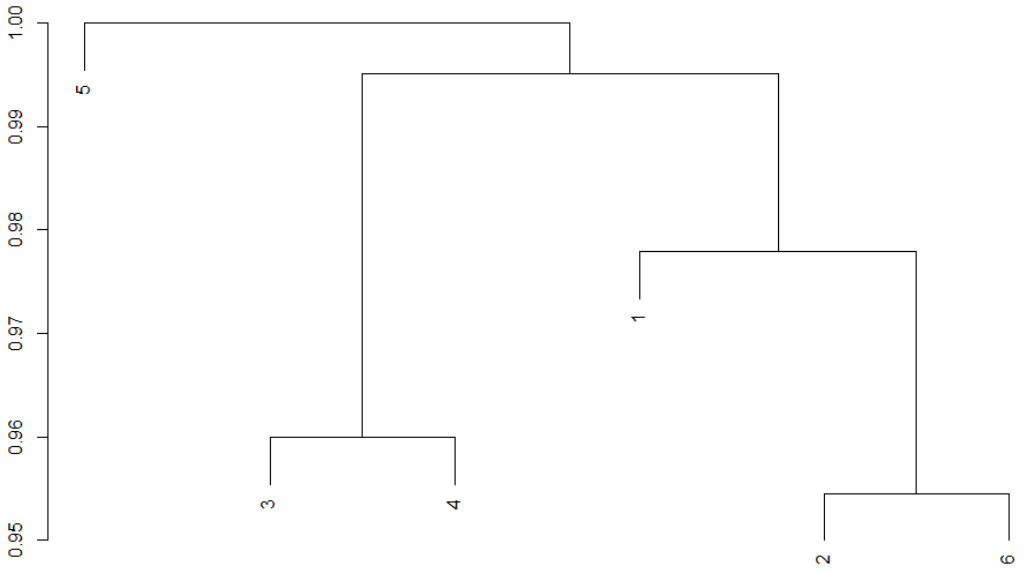
Al analizar los resultados obtenidos que se muestran en la Figura 22, se pudo observar que 170 ejemplos fueron clasificados correctamente por el modelo. Se encontró que 8 fueron errores en los que el clasificador seleccionó una clase por otra. De la visualización y análisis de esta matriz se puedo reconocer a simple vista varios clústers en los datos como se muestra en la Figura 23.

	1	2	3	4	5	6
1		0	0	0	0	0
2	3		1	1	0	0
3	0	0		0	0	0
4	0	0	2		0	0
5	0	0	0	0		0
6	0	1	0	0	0	

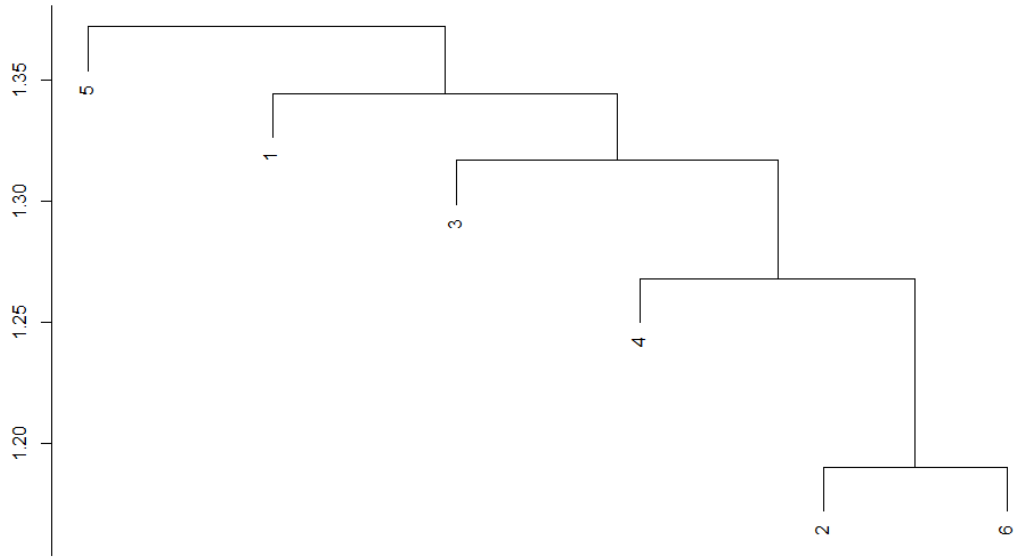
Figure 23. Clúster en Matriz de Confusión para "Dermatology".

Siguiendo el proceso planteado se procedió a inferir la jerarquía de clases partiendo de los valores de la matriz de confusión. A continuación se construyó la jerarquía de clases, aplicando los métodos de distancias ya estudiados y clustering jerárquico con el método de promedio simple.

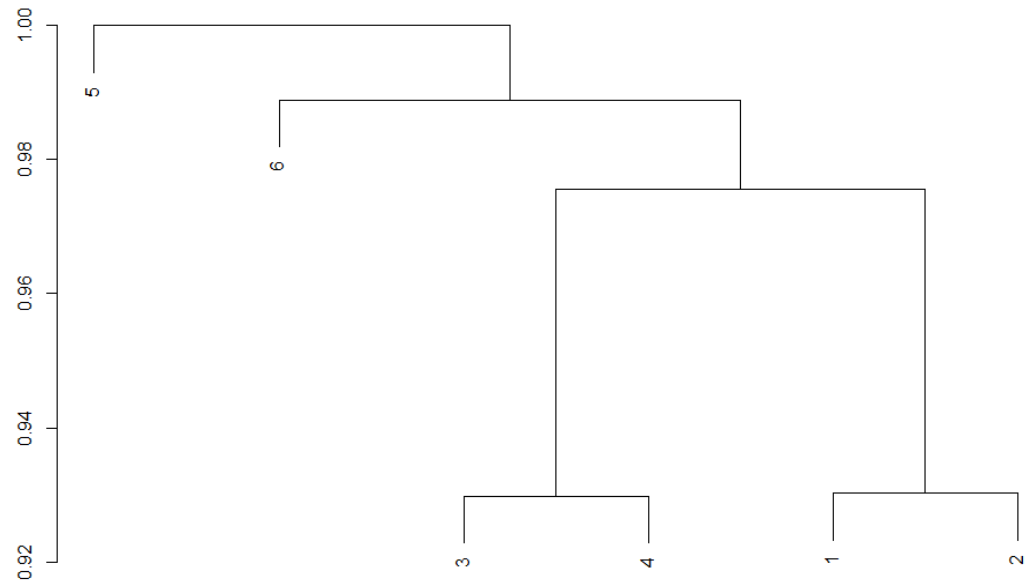
Las jerarquías generadas se presentan en la Figura 24, en ellas se puede apreciar las diferencias cuando se usan distintos métodos de distancias.



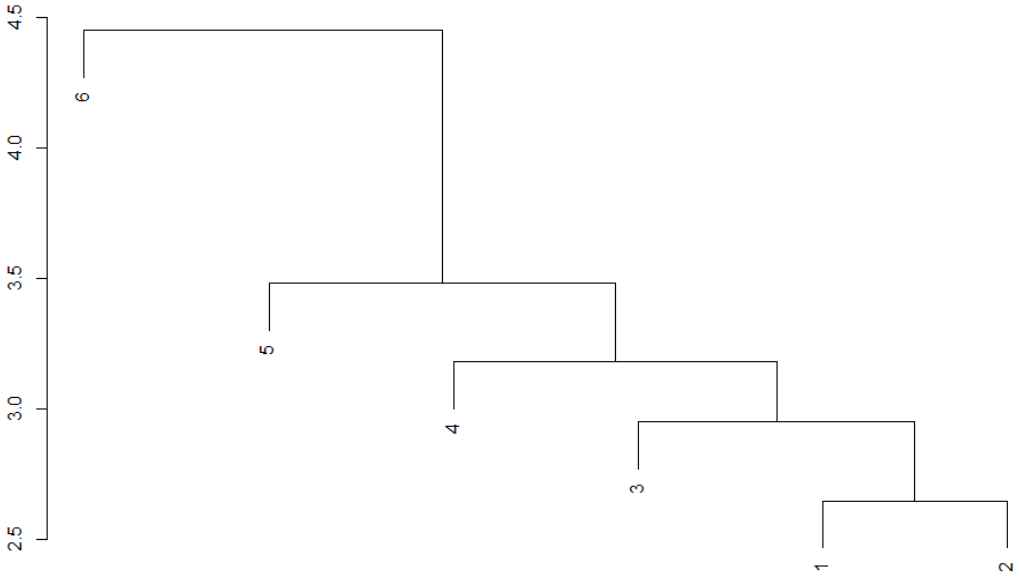
Jerarquía de clases por el método de similitud.



Jerarquía de clases por el método Euclidea.



Jerarquía de clases por el método Bray-Curtis.



Jerarquía de clases por el método Chi-Cuadrado

Figure 24. Gráficas de Jerarquías para el Conjunto de Datos "Dermatology"

Como se puede observar en los errores que aparecen en la matriz de confusión se podría decir que parecen a simple vista fallos aislados en puntos específicos de la matriz. De las gráficas obtenidas se puede observar que al usar el método de distancias por similitud y el método Bray-Curtis, tienen una dispersión horizontal de las ramas, mientras que por Chi-Cuadrado y Euclidea las jerarquías tienen una mayor profundidad.

Se procedió a la construcción del modelo de clasificación jerárquico con los datos de las jerarquías. Se aplicó éste clasificador al conjunto de datos de prueba y se obtuvo los siguientes indicadores:

	Plano	Jerárquico			
		Similitud	Euclídea	Bryan Curtis	Chi - Cuadrado
Acuracy	0,955	0,910	0,949	0,955	0,893
95% CI	0,9134 - 0,9804	0,8581 - 0,9477	0,9062 - 0,9766	0,9134 - 0,9804	0,8383 - 0,9345
KAPPA	0,944	0,888	0,937	0,944	0,867
AUC	0,974	0,958	0,974	0,974	0,980
Tiempo Predicción (segundos)	1,3	20,7	21,4	25,7	30,0

Tabla 16. Resultados del Rendimiento Obtenido por los Clasificadores para "Dermatology" DataSet

Del rendimiento obtenido por los clasificadores utilizados se puede observar que los clasificadores que dieron las mejores predicciones fueron los que usaron las distancias Bryan Curtis y el clasificador plano con un igual y más alto indicador de precisión. Con respecto al estadístico KAPPA se puede observar que los que mejores resultados nuevamente los obtuvieron el clasificador plano y el clasificador que uso Bryan-Curtis que presenta una muy buena concordancia entre el valor predicho y el real. Sin embargo los valores de KAPPA para los clasificadores que usaron Chi-Cuadrado y Derivación de Similitud también son buenos. El indicador con mayor área bajo la curva es el clasificador Chi-Cuadrado lo que quiere decir que este clasificador es el que tiene la mayor capacidad de distinguir entre clases.

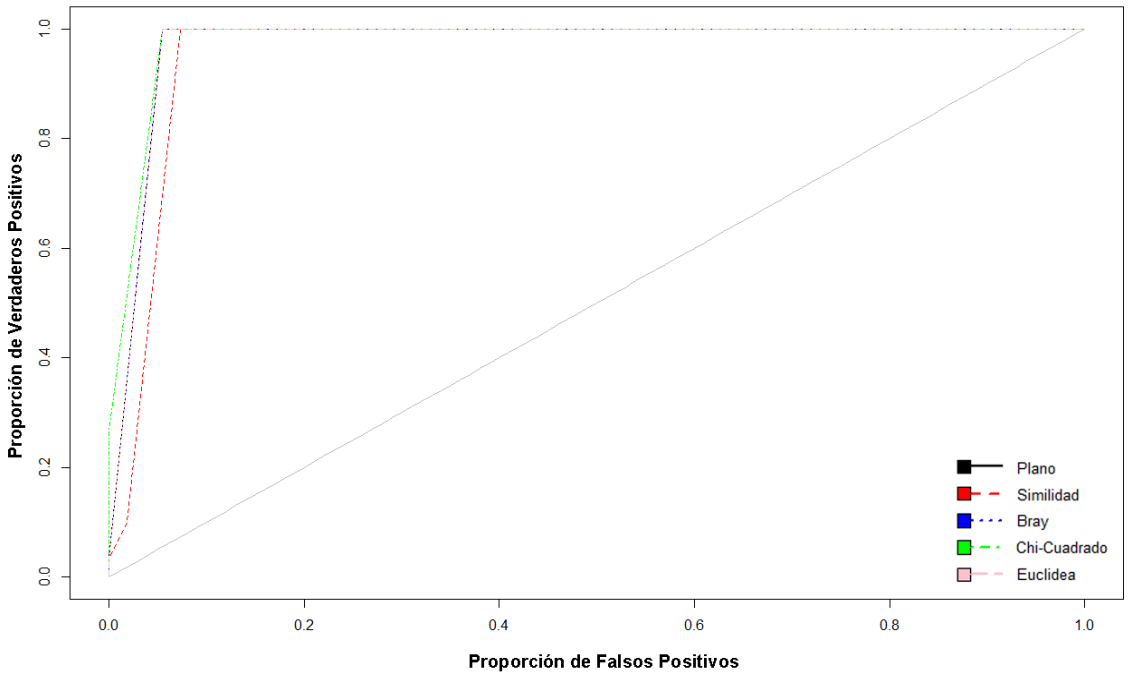


Figure 25. Análisis ROC de clasificadores para "Forest Type" DataSet.

4.5.1.3. Experimentos 3 "Letters Recognition"

Este experimento se realizó sobre la base de datos "Letters Recognition" (Reconocimiento de Letras). La cual se obtuvo del repositorio de aprendizaje automático de la Universidad de California (UCI). El objetivo de este conjunto de datos es el de identificar cada una de las letras del alfabeto ingles a través de un conjunto de características. Estas características representan los pixeles en blanco y negro rectangulares que forman la letra. Las imágenes fueron basadas en 20 fuentes diferentes y cada letra dentro de estas fuentes fue desordenada aleatoriamente para producir 20000 estímulos únicos. Cada estimulo fue convertido en 16 atributos numéricos que representan una letra.

A continuación se puede observar los resultados obtenidos de los experimentos realizados:

4.5.1.3.1. Resultados del Experimento 3

Siguiendo el paso 1 de nuestro esquema de experimento se realizó la construcción del modelo plano y se obtuvo los siguientes resultados:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	379	0	0	0	0	0	1	1	1	4	1	4	6	0	2	1	1	1	3	2	3	1	1	6	0	0
B	0	303	0	19	2	10	10	2	2	2	4	1	2	0	5	5	2	12	19	0	1	1	3	14	1	4
C	3	1	315	0	13	1	21	3	3	0	1	0	0	0	6	1	5	0	1	0	3	0	0	2	0	1
D	1	14	0	343	0	2	3	10	2	1	1	3	1	8	22	1	1	4	4	2	0	0	0	1	3	6
E	0	5	9	0	326	2	11	7	1	5	11	7	0	0	0	0	11	4	4	3	0	0	0	8	2	4
F	0	1	0	0	0	323	0	5	6	2	1	0	0	2	0	22	0	0	2	4	0	4	4	0	3	5
G	2	0	9	2	5	1	288	8	1	1	3	4	0	0	6	2	7	6	10	2	2	1	3	4	0	3
H	1	12	1	9	0	2	3	285	0	7	19	0	3	12	3	5	8	5	3	2	7	1	5	5	2	0
I	0	0	0	1	1	0	0	1	332	17	1	3	2	1	0	2	4	1	2	0	0	0	1	0	5	1
J	0	4	1	0	0	2	1	0	8	307	0	0	1	1	0	3	1	0	0	3	1	0	1	8	1	5
K	2	3	0	2	0	1	1	12	1	1	294	4	5	1	2	0	0	22	0	2	2	0	2	16	0	1
L	2	1	0	0	2	0	8	3	1	2	0	336	0	0	1	2	1	1	1	0	1	0	0	1	1	5
M	1	0	0	0	1	0	5	2	0	0	3	2	350	11	0	1	0	2	0	0	2	2	5	0	2	0
N	0	3	0	5	0	0	1	1	0	1	3	0	6	337	7	2	0	11	0	0	11	4	1	1	0	0
O	0	4	6	6	0	0	10	5	1	2	0	0	0	2	298	0	20	2	4	1	7	2	8	1	0	3
P	1	6	2	0	0	14	2	0	4	3	0	2	0	0	0	346	0	0	0	1	1	1	0	1	2	0
Q	0	1	3	1	2	0	13	2	3	3	0	3	1	1	7	0	315	4	2	2	2	0	0	0	4	9
R	0	6	3	1	3	3	2	10	0	2	13	3	2	5	2	1	5	295	4	3	3	4	0	6	1	0
S	2	8	1	2	8	4	0	5	4	4	4	4	0	0	2	2	1	3	295	1	1	1	0	5	2	14
T	0	0	2	1	6	12	2	0	2	0	2	1	0	0	0	1	0	0	7	356	3	3	0	0	6	3
U	0	0	14	1	0	0	1	2	0	1	1	0	11	1	5	0	0	0	2	1	353	2	2	0	1	0
V	0	7	0	0	0	3	0	1	1	1	0	0	1	4	0	1	0	1	2	2	1	344	6	3	2	0
W	0	0	0	0	0	0	1	1	0	0	0	0	4	0	7	1	0	0	0	1	0	5	334	0	0	0
X	0	2	0	5	7	0	1	1	2	4	6	2	0	0	0	0	3	4	4	5	1	0	0	310	0	1
Y	0	0	1	1	1	4	1	0	1	0	0	0	1	5	1	2	0	0	0	5	1	6	0	0	354	0
Z	0	2	1	3	7	3	0	0	1	3	1	1	0	0	0	0	6	2	5	0	0	0	0	4	1	302

Figure 26. Matriz de confusión con clasificador plano J48 para "Letters".

Al analizar los resultados obtenidos al entrenar un clasificador plano se pudo observar que 8420 ejemplos fueron clasificados correctamente por el modelo. Además se encontró que 1577 fueron errores en los que el clasificador seleccionó una clase por otra. De la visualización y análisis de esta matriz se puedo reconocer a simple vista varios clústers en los datos como se muestra en la Figura 27.

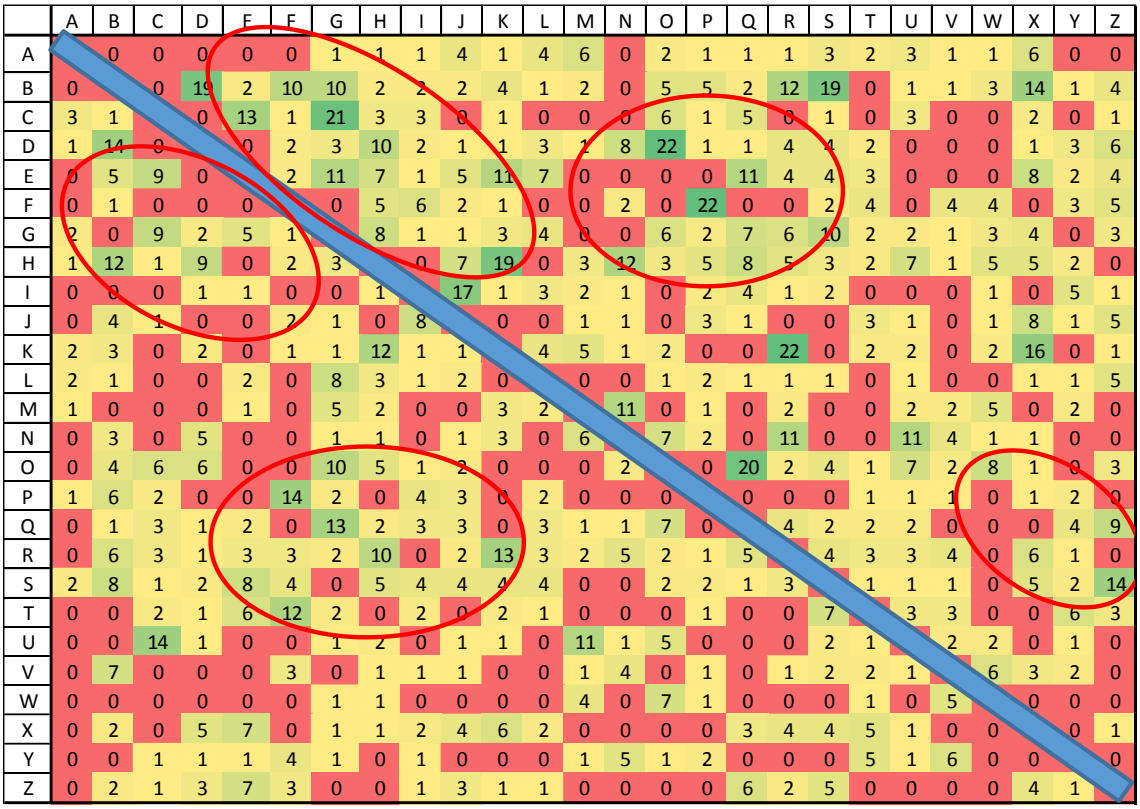
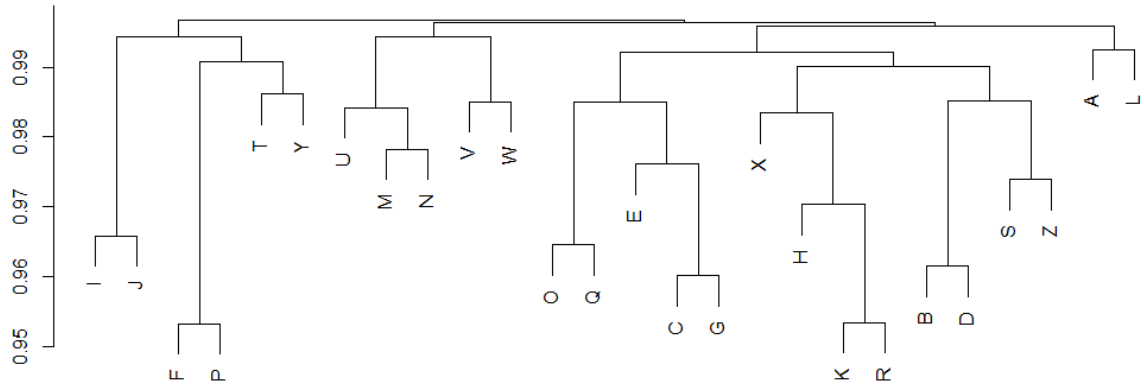


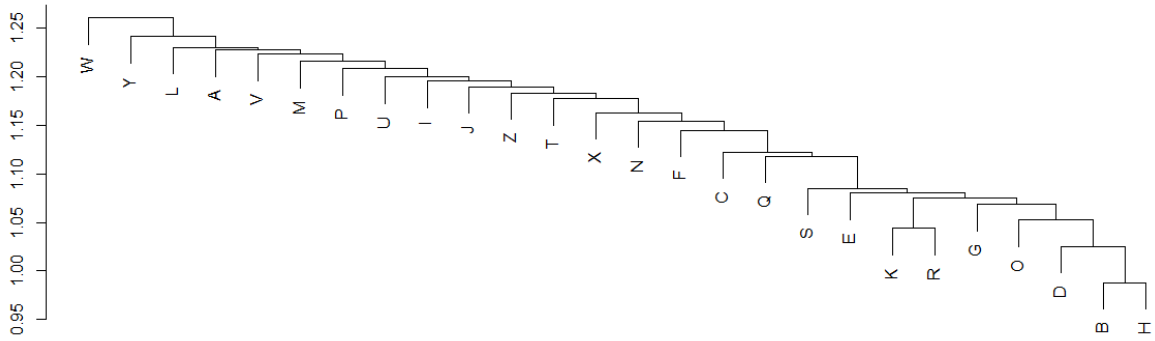
Figure 27. Clúster en Matriz de Confusión para "Letters".

Se procedió a inferir la jerarquía de clases partiendo de los valores de la matriz de confusión. A continuación se construyó la jerarquía de clases, aplicando los métodos de distancias ya estudiados y clustering jerárquico con el método de promedio simple. Las jerarquías generadas se presentan en la Figura 28, se puede apreciar las diferencias entre los distintos métodos de distancias.

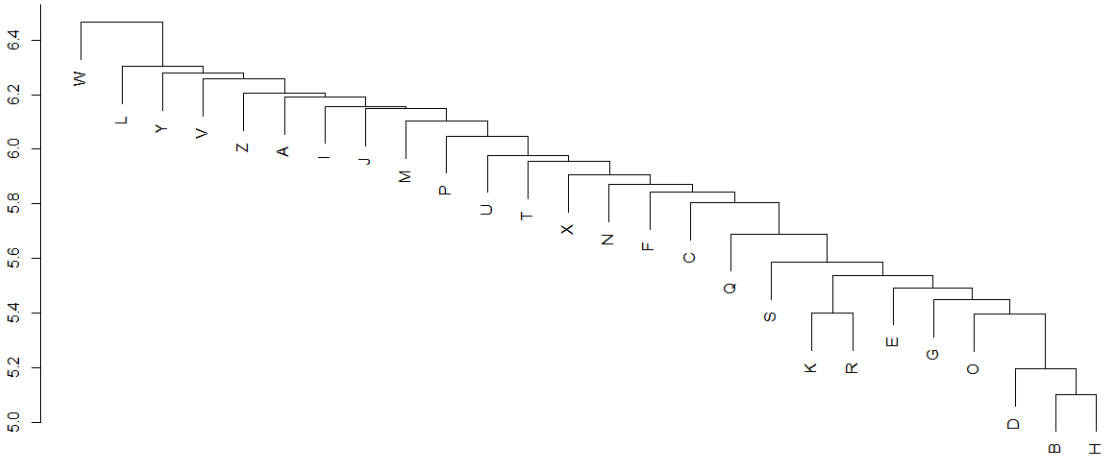


Jerarquía de clases por el método Similitud

Jerarquía de clases por el método Bryan Curtis



Jerarquía de clases por el método Euclídea



Jerarquía de clases por el método Chi-Cuadrado

Figure 28. Gráficas de Jerarquías para el Conjunto de Datos "Letters"

Al usar los métodos de derivación de similitud y Bray-Curtis se puede ver que las estructuras tienen sus ramificaciones agrupadas en ramas que contienen nodos similares. Por ejemplo, la letra "O","Q","C","G" se encuentran en una misma rama ya que tiene características semejantes en su forma. Mientras que al utilizar el método de distancias Euclídea y Chi-Cuadrado solamente se han agrupado las rama las letras "K","R" y la "B","H". La eficiencia de la inferencia se verá en los modelos de clasificación generados a partir de las jerarquías.

Se procedió a la construcción del modelo de clasificación jerárquico con los datos de las jerarquías. A continuación se aplicó este clasificador al conjunto de datos de prueba y se obtuvo los siguientes indicadores:

	Plano	Jerárquico			
		Similidad	Euclídea	Bryan Curtis	Chi - Cuadrado
Acuracy	0,843	0,750	0,841	0,838	0,831
95% CI	0,8353 - 0,8497	0,7416 - 0,7587	0,8337 - 0,8481	0,831 - 0,8456	0,8306 - 0,8452
KAPPA	0,836	0,740	0,835	0,832	0,832
AUC	0,967	0,926	0,916	0,930	0,934
Tiempo Predicción (segundos)	1,86	1930	5194	3240	5028

Tabla 17. Resultados del Rendimiento Obtenido por los Clasificadores para "Letters" DataSet

Del rendimiento obtenido por los clasificadores utilizados se puede ver que sus indicadores no mejoran los resultados de un clasificador plano, pero se acercan a sus valores. El clasificador que usó el método de la distancia Euclídea al parecer realizó una clasificación similar al clasificador plano.

Con relación al indicador KAPPA el clasificador que usó el método de distancia por similitud tiene la menor concordancia entre las predicciones y los datos reales, la cual es buena mientras que para los demás clasificadores la concordancia es muy buena. Además se puede observar en la Figura 29, que el área bajo la curva del clasificador Chi-Cuadrado es el que más se aproxima a un clasificador plano que aún sigue siendo la mayor del grupo.

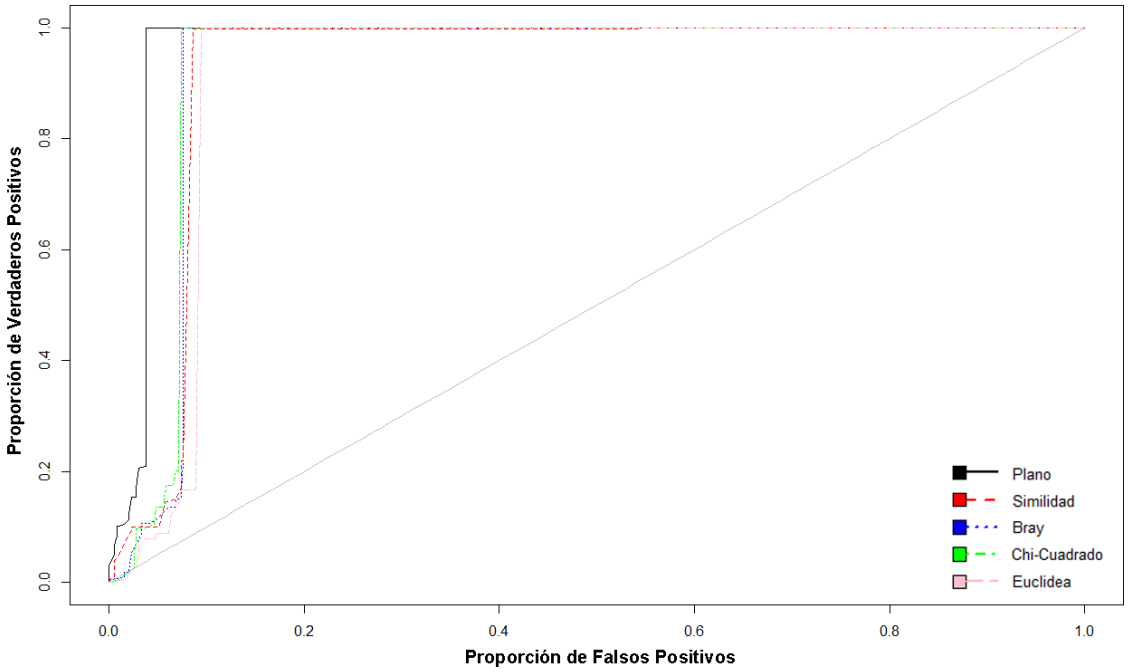


Figure 29. Análisis ROC de clasificadores para "Letters" DataSet.

CAPÍTULO

5 CONCLUSIONES

5. Conclusiones

Esta capitulo se presenta las conclusiones obtenidas al desarrollar este trabajo de fin de master.

5.1. De la Documentación.

Al realizar este trabajo de investigación se encontró que ya existen estudios realizados en el área de clasificadores y clustering jerárquicos, se encontró documentación histórica relevante y artículos relacionados a estos temas. Sin embargo, la construcción de jerarquías a través de datos es un tema que aún no ha sido investigado en profundidad por lo cual el trabajo realizado en este TFM es relevante y útil para resolver este problema.

5.2. De los Experimentos

De los resultados obtenidos en los experimentos se puede observar que no existe una única función para la inferencia automática de jerarquías entre clases, si no que puede aplicarse varios métodos y esto varía según el problema de predicción que se quiere resolver y de las características individuales de la base de datos. Por ejemplo para la base de datos "Forest Type" se tuvo una mejora con el método de similitud, Euclídea y Bray-Curtis, para la base de datos Dermatology el mejor fue el método de Bray-Curtis y para la base de datos de "Letters" fue la Euclidea.

De los resultados obtenidos en el Experimento se puede observar que el efecto sobre el rendimiento después de aprender la jerarquía de clases es diferente para cada problema. En general el realizar la inferencia de clases y construir un clasificador partiendo de la estructura de las jerarquías parece lograr que se mejore la capacidad que tiene el clasificador de distinguir entre clases y esto es descrito por la mejora que hay en el área bajo la curva ROC. Con respecto al accuracy no se logrado comprobar una mejora sustancial sobre este indicador, ya que únicamente al usarlo con la base de datos "Forest Type" se logró ver

mejoras en este indicador, pero aun así los resultado obtenidos por el clasificador jerárquico son tan buenos como el clasificador plano de referencia.

De la misma forma sucede con el estadístico Kappa que muestra la concordancia entre las clases reales y las predichas se ha visto que únicamente se mejoró con la base de datos "Forest Type", pero en general con los otros métodos de distancia y bases de datos se obtiene un valor igual o levemente menor. Lo cual nos indica que la influencia que tiene el azar no es relevante en ninguno de los experimentos.

5.3. De las Métodos de Distancia

De los métodos utilizados para la generación de jerarquías se puede observar que si se calcula las distancias a través de métodos de distancias ya conocidos como son las distancias de similidad, Euclidea, Bryan-Curtis o Chi-Cuadrado es posible encontrar distancias que representen verdaderamente la relación entre clases en una medida diferente. Por lo tanto es necesario realizar futuras validaciones en el algoritmo de forma que se realice una mejor representación de las distancias entre clases. Además se podría desarrollar un algoritmo que en función de los indicadores de rendimiento pueda seleccionar la función óptima que permita distinguir la estructura que mejor represente la jerarquía de clases real.

También se encontró que mientras más profunda es la estructura jerárquica mayor el tiempo que se gasta en generar un clasificador jerárquico y también realizar predicciones, esto es razonable ya que por su tipo es necesario que recorra la jerarquía de clases en busca de una clase que se ajuste al ejemplo.

5.4. De los Procesos

Se estableció el proceso necesario para la creación de un clasificador jerárquico partiendo de la información obtenida en la jerarquía de clases. Esto es importante ya que servirá como guía para futuras investigaciones sobre la inferencia de jerarquías partiendo de las distancia entre clases.

5.5. De las Herramientas Desarrolladas

Con relación a las herramientas utilizadas, se puede concluir que R es una de las herramientas más potentes para el desarrollo de análisis de datos. La variedad de librerías disponibles permiten crear scripts que se ocupen automáticamente realizar rutinas conocidas y la ejecución inmediata de las técnicas propuestas. Sin embargo se encontró que actualmente no existen librerías que permita la creación de clasificadores jerárquicos lo cual es muy necesario y podría ser objeto de futuros desarrollos.

Se ha elaborado una serie de scripts parametrizables que permiten realizar de forma automática inferencias de jerarquías partiendo de las distancias obtenidas a través de los métodos de distancias: Similitud, Euclidea, Bryan-Curtis y Chi-Cuadrado. Esos scripts podrían ser utilizados para futuras investigaciones y experimentación sobre otras bases de datos y otros dominios de aplicación.

5.6. Nuevas Campos por Investigación

Este proyecto de tesis ha tratado de abordar varios temas relacionados con inferencia de jerarquías, clasificación jerárquica y clustering pero existen temas en los que aún se necesita futuras investigaciones. Tal y como se menciona a continuación:

- Es necesario la creación de un paquete de librerías reutilizables que permita la creación de los diferentes tipos de clasificadores jerárquicas. De forma que se tenga herramientas para realizar pruebas y experimentos relacionados con clasificadores jerárquicos.
- Más experimentos con otros datasets, con un amplio rango de clases e incluso en dominios que sepamos son jerárquicos aunque desconozcamos la jerarquía (como por ejemplo, en el ámbito de la

clasificación de documentos), son necesarios para poder extraer conclusiones más generales.

- Asimismo, se podrían realizar otros experimentos modificando las variables que intervienen en el experimento. Por ejemplo, en los experimentos realizados se mantuvo fijo el algoritmo de clasificación (un árbol de decisión), pero es posible realizar pruebas con otros algoritmos como son las redes neuronales, máquinas de soporte vectorial, etc. Los parámetros que se podrían modificar para futuros experimentos son:
    - Método del Clasificador Plano
    - Función para el Cálculo de Distancia
    - Método de Cluster Jerárquico
    - Método de Clasificador Jerárquico

Referencias

Referencias

Alonso, J. B. (2014). *Aprendizaje Automático.* Barcelona: Universidad Politecnica de Catalunya.

Berkhin, P. (2002). *Survey of Clustering Data Mining Techniques.* San Jose CA: Accrue Software.

Breiman, L., Friedman, J. H., & Olshen, R. A. (1984). *Classification and Regression.* Belmont: Wadsworth.

Bronoski, H., Silla, C., & Nievola, J. C. (2013). An evaluation of global-model hierarchical classification. *Computers & Mathematics with Applications*, 1991-2002.

Cerda, L. J., & Villarroel, L. (2008). Evaluación de la concordancia inter-observador en investigación pediátrica: Coeficiente de Kappa. *Rev Chil Pediatr*, 54-58.

Cesar Ferri; Maria José Ramirez-Quintana. (2015). An adaptive probabilistic classification method for dynamic class hierarchies. *LMCE*.

Cha, S.-H. (2007). Comprehensive Survey on Distance/Similarity Measures between Probability Density Functions. *International Jornal of Mathematical Models and Methods in Applied Sciences*, 300-307.

Ferri, C., Orallo, J., & Salido, M. A. (2003). Volume under the ROC Surface for Multi-class Problems. *Machine Learning*, 108-120.

Foster Provost, T. F. (2013). *Data Science for Business* . O'Reilly Media.

Greenacre, M. (2008). *Correspondence Analysis and Related Methods.* Stanford University: Department of Statistics.

Hall, P., Dean, J., & Kabul , I. (2014). An Overview of Machine Learning with SAS Enterprise Miner. 1-24.

Hernández Orallo, J., Ramirez Quintana, M. J., & Ferri Ramirez, C. (2004). *Introducción a la Mineria de Datos.* Pearson Prentice Hall.

Markham, K. (2015). *Optimizing your model with cross-validation.* http://blog.kaggle.com/2015/06/29/scikit-learn-video-7-optimizing-your-model-with-cross-validation/: Kaggle.

Microsoft SQL Server. (2014). *Conceptos de mineria de datos.* https://msdn.microsoft.com/es-es/library/ms174949%28v=sql.120%29.aspx.

Silla, C. N., & Freitas, A. A. (2011). A survey of hierarchical classification across different application domains. *Springer Online*.

Usama Fayyad, G. P.-S. (1996). The KDD Process for Extracting Useful Knowledge from Volumes of Data. *Comunication of the ACM*, 27-34.

Visa, S., Ramsay, B., Ralescu, A., & Van der Knaap, E. (2011). Confusion Matrix-based Feature Selection. *The 22nd Midwest Artificial Intelligence and Cognitive Science Conference 2011*, (págs. 120-127).

Waterman, D. A., & Hayes-Roth, F. (1978). *Pattern-Directed Inference Systems.* Orlando: Academic Press Inc.

Witten, I., Eibe, F., & Hall, H. A. (2011). *Practical Machine Learning Tools and Techniques.* Burlington: Elsevier Inc.

ANEXOS

Anexo 1

A continuación se muestran los algoritmos de clasificación jerárquica según su definición formal (Silla & Freitas, 2011).

Approach (Θ)	Class structure (Ω)	List of works
Flat classifier	Tree	Barbedo and Lopes (2007)
	DAG	Hayete and Bienkowska (2005)
LCN	Tree	D'Alessio et al. (2000), Dumais and Chen (2000), Sun and Lim (2001), Mladenic and Grobelnik (2003), Sun et al. (2003, 2004), Liu et al. (2005), Wu et al. (2005), Cesa-Bianchi et al. (2006a,b), Cesa-Bianchi and Valentini (2009), Esuli et al. (2008), Punera and Ghosh (2008), Xue et al. (2008), Bennett and Nguyen (2009), Binder et al. (2009), Valentini (2009) and Valentini and Re (2009)
	DAG	Barutcuoglu and DeCoro (2006), Barutcuoglu et al. (2006), DeCoro et al. (2007), Guan et al. (2008) and Jin et al. (2008)
LCPN	Tree	Koller and Sahami (1997), Chakrabarti et al. (1998), McCallum et al. (1998), Weigend et al. (1999), D'Alessio et al. (2000), Ruiz and Srinivasan (2002), Burred and Lerch (2003), Tikk and Biró (2003), Tikk et al. (2003), McKay and Fujinaga (2004), Li and Ogihara (2005), Brecheisen et al. (2006a), Tikk et al. (2007), Holden and Freitas (2005, 2006, 2008, 2009), Xiao et al. (2007), Secker et al. (2007, 2010), Costa et al. (2008), Silla Jr and Freitas (2009b) and Gauch et al. (2009)
	DAG	Kriegel et al. (2004)
LCL	Tree	Clare and King (2003)
	DAG	
Global classifier	Tree	Labrou and Finin (1999), Wang et al. (1999, 2001), Clare and King (2003), Blockeel et al. (2006), Cai and Hofmann (2004, 2007), Dekel et al. (2004a,b), Peng and Choi (2005), Rousu et al. (2005, 2006), Astikainen et al. (2008), Seeger (2008), Silla Jr and Freitas (2009a) and Qiu et al. (2009)
	DAG	Kiritchenko et al. (2005, 2006), Alves et al. (2008), Dimitrovski et al. (2008), Vens et al. (2008), Aleksovski et al. (2009), Otero et al. (2009), Wang et al. (2009)

Anexo 2

Tabla de Incrementos en Indicadores de Rendimiento por Clase de "Letter" Dataset.

CLASIFICADO CON APRENDIZAJE JERARQUICO METODO PROPIO																											
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Sensitivity	-2,57%	-0,83%	-100,00%	-0,15%	-100,00%	0,62%	-100,00%	0,17%	-1,68%	-0,49%	1,67%	-2,60%	2,64%	0,15%	-1,71%	-3,59%	2,93%	0,00%	3,59%	-2,89%	1,81%	-1,47%	1,76%	4,76%	-3,36%	2,74%
Specificity	0,04%	0,03%	0,34%	0,01%	0,49%	-0,14%	0,41%	0,12%	0,05%	-0,01%	-0,12%	0,04%	0,05%	0,13%	0,06%	-0,05%	-0,09%	-0,09%	-4,35%	0,11%	0,03%	0,04%	-0,16%	0,04%	-0,13%	-1,87%
Pos Pred Value	0,64%	0,36%	#VALUE!	0,09%	#VALUE!	-3,27%	#VALUE!	3,00%	1,06%	-0,35%	-2,43%	0,77%	1,37%	3,29%	1,12%	-1,75%	-1,74%	-2,33%	-49,64%	2,42%	0,81%	0,81%	-3,84%	1,64%	-3,49%	-32,62%
Neg Pred Value	-0,10%	-0,03%	-1,60%	-0,01%	-1,66%	0,02%	-1,46%	0,01%	-0,06%	-0,02%	0,05%	-0,09%	0,10%	0,01%	-0,05%	-0,13%	0,10%	0,00%	0,09%	-0,10%	0,07%	-0,05%	0,06%	0,16%	-0,12%	0,08%
Prevalence	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
Detection Rate	-2,57%	-0,83%	-100,00%	-0,15%	-100,00%	0,62%	-100,00%	0,18%	-1,68%	-0,49%	1,67%	-2,59%	2,65%	0,15%	-1,71%	-3,59%	2,93%	0,00%	3,59%	-2,89%	1,82%	-1,47%	1,76%	4,76%	-3,36%	2,74%
Detection Prevalence	-3,21%	-1,19%	-100,00%	-0,23%	-100,00%	3,88%	-100,00%	-2,83%	-2,75%	-0,14%	4,10%	-3,37%	1,27%	-3,14%	-2,82%	-1,85%	4,66%	2,33%	52,30%	-5,30%	0,99%	-2,29%	5,61%	3,12%	0,13%	35,05%
Balanced Accuracy	-1,23%	-0,35%	-29,81%	-0,07%	-29,56%	0,21%	-26,95%	0,14%	-0,76%	-0,23%	0,69%	-1,18%	1,28%	0,14%	-0,72%	-1,67%	1,28%	-0,05%	-0,68%	-1,29%	0,86%	-0,68%	0,76%	2,19%	-1,63%	0,27%
CLASIFICADO CON APRENDIZAJE JERARQUICO METODO DISTANCIA EUCLIDEA																										
Sensitivity	-2,99%	2,57%	-0,96%	-2,85%	-1,87%	0,00%	3,68%	2,56%	-0,45%	-1,82%	-1,03%	-1,05%	-0,29%	-1,81%	3,56%	-2,37%	3,08%	2,80%	1,50%	-3,19%	-1,00%	-4,72%	0,45%	6,49%	-1,87%	0,98%
Specificity	0,07%	-0,13%	0,07%	0,10%	-0,01%	-0,12%	0,02%	-0,19%	0,09%	0,05%	0,01%	0,07%	0,10%	0,09%	-0,04%	-0,03%	-0,06%	-0,26%	0,01%	0,10%	0,02%	0,11%	-0,05%	-0,03%	-0,09%	0,00%
Pos Pred Value	1,38%	-1,93%	1,60%	1,74%	-0,55%	-2,91%	1,16%	-3,43%	2,43%	1,29%	0,05%	1,88%	2,49%	2,15%	-0,10%	-1,05%	-0,99%	-5,26%	0,44%	2,26%	0,40%	2,64%	-1,36%	0,19%	-2,52%	0,11%
Neg Pred Value	-0,11%	0,08%	-0,03%	-0,10%	-0,06%	0,00%	0,11%	0,08%	-0,02%	-0,06%	-0,03%	-0,04%	-0,01%	-0,06%	0,11%	-0,08%	0,10%	0,09%	0,05%	-0,11%	-0,04%	-0,16%	0,02%	0,22%	-0,07%	0,03%
Prevalence	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
Detection Rate	-2,99%	2,57%	-0,96%	-2,85%	-1,87%	0,00%	3,68%	2,56%	-0,45%	-1,82%	-1,03%	-1,05%	-0,29%	-1,81%	3,56%	-2,37%	3,08%	2,80%	1,50%	-3,19%	-1,00%	-4,72%	0,45%	6,48%	-1,87%	0,98%
Detection Prevalence	-4,37%	4,50%	-2,56%	-4,59%	-1,33%	2,91%	2,52%	5,99%	-2,89%	-3,11%	-1,08%	-2,94%	-2,77%	-3,95%	3,65%	-1,31%	4,06%	8,04%	1,06%	-5,44%	-1,40%	-7,35%	1,82%	6,29%	0,65%	0,87%
Balanced Accuracy	-1,41%	1,09%	-0,41%	-1,24%	-0,86%	-0,07%	1,62%	1,04%	-0,16%	-0,79%	-0,45%	-0,45%	-0,08%	-0,78%	1,59%	-1,10%	1,37%	1,11%	0,67%	-1,43%	-0,45%	-2,13%	0,18%	2,96%	-0,93%	0,45%
CLASIFICADO CON APRENDIZAJE JERARQUICO METODO DISTANCIA BRYAN CURTIS																										
Sensitivity	-2,02%	-0,50%	-0,32%	-1,93%	-1,56%	-0,62%	1,37%	-2,33%	-2,00%	1,92%	-0,68%	1,03%	0,28%	1,89%	-2,23%	-2,52%	2,17%	-0,17%	1,67%	-3,64%	-0,57%	-2,69%	0,45%	6,06%	-2,02%	2,89%
Specificity	0,14%	0,06%	0,01%	0,02%	0,11%	-0,10%	-0,16%	0,08%	0,06%	0,05%	-0,08%	0,02%	0,07%	-0,08%	0,04%	0,00%	-0,14%	-0,17%	-0,15%	0,08%	0,07%	0,01%	-0,05%	0,00%	0,02%	-0,11%
Pos Pred Value	3,14%	1,18%	0,21%	0,07%	2,28%	-2,54%	-3,56%	1,29%	1,31%	1,63%	-2,26%	0,49%	1,72%	-1,66%	0,60%	-0,27%	-2,97%	-4,24%	-3,18%	1,67%	1,61%	-0,12%	-1,36%	0,76%	0,38%	-2,50%
Neg Pred Value	-0,08%	-0,02%	-0,01%	-0,07%	-0,05%	-0,02%	0,04%	-0,07%	-0,07%	0,06%	-0,02%	0,04%	0,01%	0,07%	-0,07%	-0,09%	0,07%	-0,01%	0,05%	-0,13%	-0,02%	-0,09%	0,02%	0,21%	-0,07%	0,09%
Prevalence	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
Detection Rate	-2,02%	-0,50%	-0,32%	-1,93%	-1,56%	-0,62%	1,37%	-2,33%	-2,00%	1,92%	-0,68%	1,03%	0,28%	1,89%	-2,23%	-2,52%	2,17%	-0,17%	1,67%	-3,64%	-0,57%	-2,68%	0,45%	6,06%	-2,02%	2,89%
Detection Prevalence	-5,15%	-1,68%	-0,53%	-2,00%	-3,83%	1,91%	4,92%	-3,62%	-3,32%	0,29%	1,58%	0,54%	-1,43%	3,55%	-2,82%	-2,25%	5,14%	4,07%	4,84%	-5,30%	-2,18%	-2,56%	1,82%	5,30%	-2,40%	5,40%
Balanced Accuracy	-0,91%	-0,19%	-0,14%	-0,87%	-0,65%	-0,34%	0,50%	-0,97%	-0,90%	0,90%	-0,35%	0,50%	0,17%	0,84%	-0,95%	-1,15%	0,91%	-0,17%	0,66%	-1,64%	-0,23%	-1,26%	0,18%	2,77%	-0,94%	1,27%
CLASIFICADO CON APRENDIZAJE JERARQUICO METODO DISTANCIA CHI-CUADRADO																										
Sensitivity	-1,88%	0,49%	-1,45%	-1,63%	-3,66%	0,31%	1,03%	0,00%	-2,63%	-0,82%	-2,44%	0,44%	0,00%	-1,81%	3,25%	0,72%	2,63%	-1,03%	1,50%	-2,59%	-0,43%	-4,56%	0,45%	6,49%	-1,14%	2,11%
Specificity	0,03%	-0,15%	0,09%	0,05%	-0,21%	-0,14%	0,04%	-0,15%	0,14%	0,08%	-0,04%	0,01%	0,09%	0,17%	-0,06%	-0,14%	-0,08%	-0,08%	0,00%	0,10%	-0,02%	0,10%	-0,05%	-0,01%	-0,09%	0,07%
Pos Pred Value	0,44%	-3,03%	2,24%	0,86%	-5,52%	-3,33%	1,30%	-3,38%	3,61%	2,14%	-1,50%	0,31%	2,37%	4,11%	-0,77%	-3,26%	-1,56%	-2,34%	0,31%	2,17%	-0,43%	2,49%	-1,36%	0,56%	-2,43%	2,11%
Neg Pred Value	-0,07%	0,01%	-0,05%	-0,06%	-0,12%	0,01%	0,03%	0,00%	-0,09%	-0,03%	-0,07%	0,02%	0,00%	-0,06%	0,10%	0,03%	0,09%	-0,03%	0,05%	-0,09%	-0,02%	-0,16%	0,02%	0,22%	-0,04%	0,07%
Prevalence	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
Detection Rate	-1,88%	0,49%	-1,45%	-1,63%	-3,65%	0,31%	1,03%	0,00%	-2,63%	-0,82%	-2,44%	0,44%	0,00%	-1,81%	3,24%	0,72%	2,63%	-1,03%	1,50%	-2,59%	-0,43%	-4,56%	0,45%	6,48%	-1,14%	2,11%
Detection Prevalence	-2,32%	3,52%	-3,69%	-2,48%	1,87%	3,64%	-0,27%	3,38%	-6,24%	-2,96%	-0,94%	0,14%	-2,37%	-5,92%	4,02%	3,98%	4,18%	1,31%	1,19%	-4,76%	0,00%	-7,05%	1,82%	5,92%	1,28%	0,00%
Balanced Accuracy	-0,90%	0,14%	-0,61%	-0,72%	-1,77%	0,06%	0,47%	-0,08%	-1,14%	-0,33%	-1,09%	0,21%	0,05%	-0,74%	1,43%	0,26%	1,15%	-0,50%	0,67%	-1,16%	-0,21%	-2,06%	0,18%	2,97%	-0,59%	1,00%

Anexo 3

Tabla de Incrementos en Indicadores de Rendimiento por Clase de "Forest Type" Dataset.

Metodo Propio				
	Class: d	Class: h	Class: o	Class: s
Sensitivity	0,00%	6,45%	0,00%	0,42%
Specificity	0,00%	-0,18%	0,00%	1,85%
Pos Pred Value	0,00%	0,83%	0,00%	2,12%
Neg Pred Value	0,00%	0,71%	0,00%	0,45%
Prevalence	0,00%	-0,01%	0,00%	0,00%
Detection Rate	0,00%	6,43%	0,00%	0,41%
Detection Prevalence	0,00%	5,61%	0,00%	-1,71%
Balanced Accuracy	0,00%	2,88%	0,00%	1,13%
Metodo Euclidea				
	Class: d	Class: h	Class: o	Class: s
Sensitivity	0,00%	6,45%	0,00%	0,42%
Specificity	0,00%	-0,18%	0,00%	1,85%
Pos Pred Value	0,00%	0,83%	0,00%	2,12%
Neg Pred Value	0,00%	0,71%	0,00%	0,45%
Prevalence	0,00%	-0,01%	0,00%	0,00%
Detection Rate	0,00%	6,43%	0,00%	0,41%
Detection Prevalence	0,00%	5,61%	0,00%	-1,71%
Balanced Accuracy	0,00%	2,88%	0,00%	1,13%
Metodo Bryan Curtis				
	Class: d	Class: h	Class: o	Class: s
Sensitivity	0,00%	6,45%	0,00%	0,42%
Specificity	0,00%	-0,18%	0,00%	1,85%
Pos Pred Value	0,00%	0,83%	0,00%	2,12%
Neg Pred Value	0,00%	0,71%	0,00%	0,45%
Prevalence	0,00%	-0,01%	0,00%	0,00%
Detection Rate	0,00%	6,43%	0,00%	0,41%
Detection Prevalence	0,00%	5,61%	0,00%	-1,71%
Balanced Accuracy	0,00%	2,88%	0,00%	1,13%
Incremento Metodo Chi Cuadrado				
	Class: d	Class: h	Class: o	Class: s
Sensitivity	-0,73%	-5,46%	-2,70%	2,86%
Specificity	-0,24%	0,18%	0,57%	-0,63%
Pos Pred Value	-0,72%	-0,46%	2,25%	-0,08%
Neg Pred Value	-0,24%	-0,52%	-0,35%	1,99%
Prevalence	0,00%	-0,01%	0,00%	0,00%
Detection Rate	-0,72%	-5,45%	-2,68%	2,85%
Detection Prevalence	0,00%	-5,01%	-4,95%	2,93%
Balanced Accuracy	-0,43%	-2,25%	-0,93%	1,18%

Anexo 4

Código Fuente Utilizado para los Experimentos.

```
#1) Especificacion de Fuente de Datos
# Seleccion una Opcion de Fuente de Datos
opcionDataSet=7

#metodoDistancias="propio"
metodoDistancias="euclidea"
#metodoDistancias="bray"
#metodoDistancias="chi"

#modeloTipo="J48"

mostrarRoc=0;


#Fuente de Datos Forest Type
if(opcionDataSet==4){

  foresttypeTrain <- read.csv("C:/Users/DanielAndres/Desktop/Tesis2/version2Desarrollo/DataSetsPruebas/training.csv")
  foresttypeTest <- read.csv("C:/Users/DanielAndres/Desktop/Tesis2/version2Desarrollo/DataSetsPruebas/testing.csv")

  colnames(foresttypeTrain)[colnames(foresttypeTrain)=="tipo"] <- "tipo"
  foresttypeTrain <- foresttypeTrain[complete.cases(foresttypeTrain),]
  colnames(foresttypeTest)[colnames(foresttypeTest)=="tipo"] <- "tipo"
  foresttypeTest <- foresttypeTest[complete.cases(foresttypeTest),]


  #Muestro la relacion entre variables a traves de un dendograma:
  cc <- cor(foresttypeTrain[2:28],use="pairwise",method="pearson")
  thc <- hclust(dist(cc), method="average")
  dn <- as.dendrogram(thc)
  plot(dn, horiz = TRUE)


  dataSetUse<-foresttypeTrain
}


#Fuente de Datos Dermatology
if(opcionDataSet==8){

        dermatology <- read.csv("C:/Users/DanielAndres/Desktop/Tesis2/basesDatos/dermatology.txt", header=FALSE, stringsAsFactors=FALSE)


        colnames(dermatology)[colnames(dermatology)=="V35"] <- "tipo"
        dermatology <- dermatology[complete.cases(dermatology),]


        #Reemplazando por el nombre de las clases
        x<-dermatology$tipo
        x[x==1]<-"1"
        x[x==2]<-"2"
        x[x==3]<-"3"
        x[x==4]<-"4"
        x[x==5]<-"5"
```

```
55.         x[x==6]<-"6"
56.         dermatology$tipo<-x
57.         dermatology$tipo<-factor(dermatology$tipo)
58.
59.
60.         #Muestro la relacion entre variables a traves de un dendograma:
61.         cc <- cor(dermatology[1:34],use="pairwise",method="pearson")
62.         thc <- hclust(dist(cc), method="average")
63.         dn <- as.dendrogram(thc)
64.         plot(dn, horiz = TRUE)
65.
66.
67.         dataSetUse<-dermatology
68. }
69.
70.
71.
72. #Fuente de Datos Shuttle
73. if(opcionDataSet==5){
74.
75.         # Obtencion de los Datos
76.         shuttle <-
     read.table("C:/Users/DanielAndres/Downloads/datasetML/shuttle(1).trn", quote=
    "\"", stringsAsFactors=FALSE)
77.         # Asigno los nombres de las columnas
78.         colnames(shuttle)[colnames(shuttle)=="V10"] <- "tipo"
79.         shuttle$tipo<-as.character(shuttle$tipo)
80.
81.         #Reemplazando por el nombre de las clases
82.         x<-shuttle$tipo
83.         x[x==1]<-"radflow"
84.         x[x==2]<-"fpvClose"
85.         x[x==3]<-"fpvOpen"
86.         x[x==4]<-"high"
87.         x[x==5]<-"bypass"
88.         x[x==6]<-"bpvClose"
89.         x[x==7]<-"bpvOpen"
90.         shuttle$tipo<-x
91.         shuttle$tipo<-factor(shuttle$tipo)
92.
93.         # Muestro la composicion de los datos
94.         summary(shuttle)
95.         str(shuttle)
96.
97.         #Operaciones de Descripcion de datos
98.         shuttle <- shuttle[complete.cases(shuttle),]
99.
100.                #Muestro la relacion entre variables a traves de un dendograma:

101.                cc <- cor(shuttle[1:9],use="pairwise",method="pearson")
102.                thc <- hclust(dist(cc), method="average")
103.                dn <- as.dendrogram(thc)
104.                plot(dn, horiz = TRUE)
105.
106.                dataSetUse<-shuttle
107.
108.        }
109.
110.
111.
112.        if(opcionDataSet==7){
113.
114.                # Obtencion de los Datos
```

```
115.              letter <-
    read.csv("C:/Users/DanielAndres/Desktop/Tesis2/basesDatos/letter-
   recognition.data", header=FALSE)
116.              # Asigno los nombres de las columnas
117.              colnames(letter)[colnames(letter)=="V1"] <- "tipo"
118.
119.              # Muestro la composicion de los datos
120.              summary(letter)
121.              str(letter)
122.
123.              #Operaciones de Descripcion de datos
124.              letter <- letter[complete.cases(letter),]
125.
126.              #Muestro la relacion entre variables a traves de un dendograma:

127.              cc <- cor(letter[2:17],use="pairwise",method="pearson")
128.              thc <- hclust(dist(cc), method="average")
129.              dn <- as.dendrogram(thc)
130.              plot(dn, horiz = TRUE)
131.
132.              dataSetUse<-letter
133.
134.      }
135.
136.
137.      # 2) Particiono el DataSet en grupos para Test y Entrenamiento
138.      library(caret)
139.      set.seed(998)
140.      inTraining <-
    createDataPartition(dataSetUse$tipo, p = 0.50, list = FALSE)
141.      training <- dataSetUse[ inTraining,]
142.      testing  <- dataSetUse[-inTraining,]
143.      pruebas  <- dataSetUse[1:10,]
144.
145.
146.      #Selecciono los Datos para Training and Test
147.
148.
149.      #Opciones para FOREST TYPE
150.      if(opcionDataSet==4){
151.        filtTraining<-foresttypeTrain
152.        filtTesting<-foresttypeTest
153.
154.      }
155.
156.      #Opciones para Shuttle Dataset
157.      if(opcionDataSet==5){
158.              filtTraining<-training
159.              filtTesting<-testing
160.
161.      }
162.
163.
164.      #Opciones para letters Dataset
165.      if(opcionDataSet==7){
166.              filtTraining<-training
167.              filtTesting<-testing
168.
169.      }
170.
171.      #Opciones para Dermatology Dataset
172.      if(opcionDataSet==8){
173.              filtTraining<-training
174.              filtTesting<-testing
175.
```

```
176.        }
177.
178.
179.        # Validacion Cruzada:
180.        fitControl <- trainControl(## 10-fold CV
181.                method = "repeatedcv",
182.                number = 10,
183.                ## repeated ten times
184.                repeats = 10)
185.
186.
187.        # 3) Genero la Estructura Jerarquica
188.
189.        #Entreno el modelo:
190.        modArbolDecision <-
     train(tipo ~., data = filtTraining , method = modeloTipo , trControl = fitCon
    trol)
191.
192.        #Predicciones
193.        ptm <- proc.time()
194.        #dataSetprediccion<-predict(modArbolDecision,newdata = pruebas)
195.        dataSetprediccion<-predict(modArbolDecision,newdata = filtTesting)
196.        print("tiempo prediccion original")
197.        print(proc.time() - ptm)
198.
199.        #Matriz de Confusion:
200.        confusionMatrix(dataSetprediccion,filtTesting$tipo)
201.
202.        #Obtengo los valores de la tabla:
203.        tablaConfusion=table(dataSetprediccion,filtTesting$tipo)
204.
205.        #Obtengo el valor del AUC
206.        library("AUC")
207.        auc(roc(dataSetprediccion,filtTesting$tipo))
208.        rocOriginal=roc(dataSetprediccion,filtTesting$tipo)
209.        plot(rocOriginal)
210.
211.
212.        #Muestro las etiquetas:
213.        etClases=colnames(tablaConfusion)
214.
215.        #Visualizacion de la Matriz de Confusion
216.        require(ggplot2)
217.
218.        #Visualizo la Matriz de Confusion:
219.        confusion <- tablaConfusion
220.
221.        #Muestro la matriz de confusion en tabla de colores
222.        image(confusion[,ncol(confusion):1], axes=FALSE)
223.        axis(2, at = seq(0, 1, length=length(colnames(confusion))), labels=coln
    ames(confusion))
224.        heatmap(t(confusion)[ncol(confusion):1,], Rowv=NA,Colv=NA, col = heat.c
    olors(256))
225.
226.        #Obtengo la Matriz de Diferencias:
227.        source('C:/Users/DanielAndres/Desktop/Tesis2/version2Desarrollo/Desarro
    llo Clasificadores/V7/generarMatrizDiferencias.R')
228.        metodoDistancias
229.        matrizDiferencias<-
    mconfusion2oMDiferencias(tablaConfusion,metodo = metodoDistancias)
230.
231.        #Encuentro la matriz de distacias:
232.        d<-as.dist(matrizDiferencias)
233.
234.        #Realizo el clustering Jerarquico
```

```
235.        #?hclust
236.
237.        hc<-hclust(d)
238.        #Es posible usar diferentes tipos de distancias
239.        hc=hclust(d,method="average")
240.        plot(hc)
241.
242.
243.        #4) Creo el Clasificador Jerarquico a partir de la estructura generada

244.        # Construyo el Clasificador
245.        library(dendextend)
246.
247.        dendograma<-as.dendrogram(hc)
248.        listaClasificadores<-{}
249.        source('C:/Users/DanielAndres/Desktop/Tesis2/version2Desarrollo/Desarro
    llo Clasificadores/V7/RecorrerArbolDaniel.R')
250.
251.        ptm <- proc.time()
252.        resultClasificadores<-
    recorrerArbol(dendograma,listaClasificadores,modeloEntrenamiento = modeloTipo)

253.        proc.time() - ptm
254.
255.        #Obtengo el clasificador Raiz:
256.        source('C:/Users/DanielAndres/Desktop/Tesis2/version2Desarrollo/Desarro
    llo Clasificadores/V7/obtenerClasificador.R')
257.        posicionRaiz=obtenerClasificador(resultClasificadores,"raiz")
258.        posicionRaiz
259.
260.        #5 ) Realizo la prediccion con los clasificadores:
261.        clasJerarquicoPrediccion<-
    predict(resultClasificadores[posicionRaiz],newdata = filtTesting)
262.        source('C:/Users/DanielAndres/Desktop/Tesis2/version2Desarrollo/Desarro
    llo Clasificadores/V7/obtenerResultadoDeClaJerar.R')
263.
264.        #Mido el Tiempo de Procesamiento
265.        ptm <- proc.time()
266.        clasJerarquicoPrediccion<-
    obtenerResultadoDeClaJerar(filtTesting,clasJerarquicoPrediccion,resultClasific
    adores)
267.        proc.time() - ptm
268.        print("tiempo prediccion jerarquico")
269.        print(proc.time() - ptm)
270.
271.        clasJerarquicoPrediccion[[1]] <-
     factor(clasJerarquicoPrediccion[[1]], sort(levels(clasJerarquicoPrediccion[[1
    ]])))
272.
273.
274.        #Igualo los niveles (Ya que en las predicciones pueden no aparecer todo
    s)
275.        predicciones<-as.character(clasJerarquicoPrediccion[[1]])
276.        original<-as.character(filtTesting$tipo)
277.        original<-factor(original)
278.        predicciones<-factor(predicciones,levels = levels(original))
279.
280.
281.        #Muestro los resultados a traves de una Matriz de Confusion:
282.        confusionMatrix(predicciones,original)
283.        tablaConfusionClustJerarq=table(predicciones,original)
284.
285.        #Obtengo el AUC
286.        auc(roc(predicciones,original))
287.        plot(roc(predicciones,original))
```

```
288.
289.        if(metodoDistancias == "propio"){
290.                rocSimilidad=roc(predicciones,original)
291.                plot(rocSimilidad)
292.        }
293.        if(metodoDistancias == "bray"){
294.                rocBray=roc(predicciones,original)
295.                plot(rocBray)
296.        }
297.        if(metodoDistancias == "chi"){
298.                rocChi=roc(predicciones,original)
299.                plot(rocChi)
300.        }
301.        if(metodoDistancias == "euclidea"){
302.                rocEuclidea=roc(predicciones,original)
303.                plot(rocEuclidea)
304.        }
305.
306.
307.        #Resultado de Clasificador Jerarquico:
308.        tablaConfusionClustJerarq
309.        #Resultado de Clasificador Plano:
310.        tablaConfusion
311.
312.
313.
314.        #Grafico Curva Roc de Todos los Clasificadores
315.
316.        if(mostrarRoc==1){
317.                plot(rocOriginal, lty=1, col="black",main="Curvas ROC")
318.                plot(rocSimilidad,lty=2,col="red",add=TRUE)#rojo
319.                plot(rocBray, lty=3, col="blue",add=TRUE)#azul
320.                plot(rocChi, lty=4, col="green",add=TRUE)#verde
321.                plot(rocEuclidea, lty=5, col="pink",add=TRUE)#pink
322.                plot_colors <- c("black","red","blue","green","pink")
323.                legend("bottomright",c("Plano","Similidad","Bray","Chi-
    Cuadrado","Euclidea"),cex=0.8,col=plot_colors,lty=1:5,lwd=2, bty="n")
324.        }
```

Anexo 5

Implementación de Métodos para el Cálculo de Distancias.

```
mconfusion2oMDiferencias <- function(tablaConfusion,metodo="chi"){

        #Metodo Propuesto
        # 1
        if(metodo=="propio"){
                #Genero la matriz Totales:
                nrocolumnas<-ncol(tablaConfusion)
                nrofilas<-nrow(tablaConfusion)
                matrizTotales<-{}

                for(j in 1:nrofilas){
                        matrizTotales[j]<-sum(tablaConfusion[j,])
                }
                print(matrizTotales)

                #Genero Matriz de Normalizada
                nrocolumnas<-ncol(tablaConfusion)
                nrofilas<-nrow(tablaConfusion)
                matrizNormalizada<-tablaConfusion
                for(i in 1:nrocolumnas){
                        for(j in 1:nrofilas){

                                #Evito la division para cero
                                if(matrizTotales[j]==0){
                                        matrizNormalizada[j,i]<-0
                                }else{
                                        matrizNormalizada[j,i]<-tablaConfusion[j,i]/matrizTotales[j]
                                }


                        }
                }
                print(matrizNormalizada)

                #Genero Matriz de Similitud
                nrocolumnas<-ncol(matrizNormalizada)
                nrofilas<-nrow(matrizNormalizada)
                matrizSimilitud<-matrizNormalizada
                matrizSimilitud[,]<-0
                for(i in 1:nrocolumnas){
                        for(j in 1:nrofilas){
                                if(j>i){
                                        confusiones<-matrizNormalizada[j,i]+matrizNormalizada[i,j]
                                        aciertos<-matrizNormalizada[j,j]+matrizNormalizada[i,i]
                                        matrizSimilitud[j,i]<-confusiones/2

                                }

                        }
                }
                print(matrizSimilitud)

                matrizDistancias<-1-matrizSimilitud
                print(matrizDistancias)
        }

```

```
58.
59.
60.         #Metodo Distancia Euclidea
61.         # 2
62.         if(metodo=="euclidea"){
63.                 #Genero la matriz Totales:
64.                 nrocolumnas<-ncol(tablaConfusion)
65.                 nrofilas<-nrow(tablaConfusion)
66.                 matrizTotales<-{}
67.
68.                 for(j in 1:nrofilas){
69.                         matrizTotales[j]<-sum(tablaConfusion[j,])
70.                 }
71.                 print(matrizTotales)
72.
73.                 #Genero Matriz de Normalizada
74.                 nrocolumnas<-ncol(tablaConfusion)
75.                 nrofilas<-nrow(tablaConfusion)
76.                 matrizNormalizada<-tablaConfusion
77.                 for(i in 1:nrocolumnas){
78.                         for(j in 1:nrofilas){
79.
80.                                 #Evito la division para cero
81.                                 if(matrizTotales[j]==0){
82.                                         matrizNormalizada[j,i]<-0
83.                                 }else{
84.                                         matrizNormalizada[j,i]<-
    tablaConfusion[j,i]/matrizTotales[j]
85.                                 }
86.
87.
88.
89.
90.                         }
91.                 }
92.                 print(matrizNormalizada)
93.
94.                 #Genero Matriz de Distancia Euclidea
95.                 nrocolumnas<-ncol(matrizNormalizada)
96.                 nrofilas<-nrow(matrizNormalizada)
97.                 distancias<-{}
98.
99.
100.                        #Distancia Euclidea:
101.                        for(j in 1:(nrofilas-1)){
102.                                ini<-j+1
103.                                for(k in ini:nrofilas){
104.                                        vectResta<-matrizNormalizada[j,]-
    matrizNormalizada[k,]
105.                                        vectCuadrado<-(vectResta)^2
106.                                        vectTot<-sum(vectCuadrado)
107.                                        distancia<-(vectTot)^(1/2)
108.                                        distancias<-c(distancias,distancia)
109.                                }
110.                        }
111.
112.
113.                        #Creo la matriz de distancias
114.                        matrizDistancias<-matrizNormalizada
115.                        matrizDistancias[,]<-1
116.
117.                        k<-1
118.                        for(i in 1:(nrocolumnas-1)){
119.                                ini<-i+1
120.                                for(j in ini:nrofilas){
```

```
                                        matrizDistancias[j,i]<-distancias[k]
                                        k<-k+1

                                }
                        }


                        print(matrizDistancias)
                }


                #Metodo Distancia Bray-Curtis
                # 3
                if(metodo=="bray"){

                        #Genero la matriz Totales:
                        nrocolumnas<-ncol(tablaConfusion)
                        nrofilas<-nrow(tablaConfusion)
                        matrizTotales<-{}

                        for(j in 1:nrofilas){
                                matrizTotales[j]<-sum(tablaConfusion[j,])
                        }
                        print(matrizTotales)

                        #No se realiza una normalizacion en este caso
                        matrizNormalizada<-tablaConfusion

                        #Genero Matriz de Distancias
                        nrocolumnas<-ncol(matrizNormalizada)
                        nrofilas<-nrow(matrizNormalizada)
                        distancias<-{}


                        #Distancia Bray Curtis:
                        for(j in 1:(nrofilas-1)){
                                ini<-j+1
                                for(k in ini:nrofilas){
                                        vectResta<-abs(matrizNormalizada[j,]-
    matrizNormalizada[k,])
                                        total<-sum(vectResta)
                                        distancia<-
    total/(matrizTotales[j]+matrizTotales[k])
                                        distancias<-c(distancias,distancia)
                                }
                        }


                        #Creo la matriz de distancias
                        matrizDistancias<-matrizNormalizada
                        matrizDistancias[,]<-1

                        k<-1
                        for(i in 1:(nrocolumnas-1)){
                                ini<-i+1
                                for(j in ini:nrofilas){
                                        matrizDistancias[j,i]<-distancias[k]
                                        k<-k+1

                                }
                        }
```

```
                        print(matrizDistancias)
                }


                #Metodo Distancia Chi-Square
                # 4
                if(metodo=="chi"){

                        #Genero la matriz Totales:
                        nrocolumnas<-ncol(tablaConfusion)
                        nrofilas<-nrow(tablaConfusion)

                        matrizTotalesFilas<-{}
                        for(j in 1:nrofilas){
                                matrizTotalesFilas[j]<-
    sum(tablaConfusion[j,])
                        }
                        print(matrizTotalesFilas)

                        matrizTotalesCol<-{}
                        for(i in 1:nrocolumnas){
                                matrizTotalesCol[i]<-sum(tablaConfusion[,i])
                        }
                        print(matrizTotalesCol)

                        totalFilasCol<-sum(matrizTotalesFilas)


                        #Genero Matriz de Normalizada
                        nrocolumnas<-ncol(tablaConfusion)
                        nrofilas<-nrow(tablaConfusion)
                        matrizNormalizada<-tablaConfusion
                        for(i in 1:nrocolumnas){
                                for(j in 1:nrofilas){


                                        #Evito la division para cero
                                        if(matrizTotalesFilas[j]==0){
                                                matrizNormalizada[j,i]<-0
                                        }else{
                                                matrizNormalizada[j,i]<-
    tablaConfusion[j,i]/matrizTotalesFilas[j]
                                        }

                                }
                        }
                        print(matrizNormalizada)

                        #Obtengo Promedio
                        vectPromedios<-{}
                        for(i in 1:nrocolumnas){
                                vectPromedios[i]<-
    matrizTotalesCol[i]/totalFilasCol
                        }
                        print(vectPromedios)


                        #Genero Matriz de Distancia
                        nrocolumnas<-ncol(matrizNormalizada)
                        nrofilas<-nrow(matrizNormalizada)

                        distancias<-{}
                        #Distancia Chi Square:
                        for(j in 1:(nrofilas-1)){
```

```
                                ini<-j+1
                                for(k in ini:nrofilas){
                                        vectResta<-matrizNormalizada[j,]-
    matrizNormalizada[k,]
                                        vectCuadrado<-(vectResta)^2
                                        vectTot<-vectCuadrado/vectPromedios
                                        valSuma<-sum(vectTot)
                                        distancia<-(valSuma)^(1/2)
                                        distancias<-c(distancias,distancia)
                                }
                        }


                        #Creo la matriz de distancias
                        matrizDistancias<-matrizNormalizada
                        matrizDistancias[,]<-1

                        k<-1
                        for(i in 1:(nrocolumnas-1)){
                                ini<-i+1
                                for(j in ini:nrofilas){
                                        matrizDistancias[j,i]<-distancias[k]
                                        k<-k+1

                                }
                        }


                        print(matrizDistancias)
                }

                return(matrizDistancias)

        }
```

Anexo 6

Métodos Utilizados para el Recorrido del Árbol y creación de Clasificadores.

```
recorrerArbol <- function(dendograma,listaClasificadores,nivel="raiz",modeloEntrenamiento="J48"){
        library(caret)
        library(dendextend)
        str(dendograma)
        nroRamas<-length(dendograma %>% labels)
        nroRamas

        alturasNodos<-get_nodes_attr(dendograma, "height")
        alturasNodos <- alturasNodos[alturasNodos>0]
        alturasNodos

        plot(dendograma)

        if(!(is.na(alturasNodos[1]))){
                #Obtengo el numero de ramas
                nodo1<-cut(dendograma, h = alturasNodos[1])$upper
                #str(nodo1)
                #plot(nodo1)
                etiquetasHojas <- nodo1 %>% labels
                nroRamas<-length(nodo1 %>% labels)
                nroRamas

                #Obtengo todas las hojas del arbol
                miembrosDendograma<-dendograma %>% get_nodes_attr("label")
                miembrosDendograma <- miembrosDendograma[!is.na(miembrosDendograma)]

                #Filtro solo la informacion relevante
                filtTrainTemp<-filtTraining
                filtTrainTemp<-filtTrainTemp[filtTrainTemp$tipo %in% miembrosDendograma,]


                dendograma %>% get_nodes_attr("leaf")
                nroNodosRamas<-length(which(is.na(dendograma %>% get_nodes_attr("leaf"))))
                nroNodosRamas


        }else{
                nroRamas<-0
        }



        #Creo el clasificador para esta clase:
        library(plyr)

        #Reemplazo los valores de entrenamiento por las super clases
        salidasPosibles<-{}
        if(nroRamas>1){
                for(i in 1:nroRamas){

                        print(i)

                        #Corto por las ramas
```

```
                        nodoBajo<-
cut(dendograma, h = alturasNodos[1])$lower[[i]]
                        nodoBajo %>% get_nodes_attr("members")
                        plot(nodoBajo)

                        nodoBajo









                        ## ================================================
=======================
                        ## ================================================
=======================

                        #Verifico que el siguiente Nodo no se encuentre al mis
mo nivel antes de
                        #llamar a la recursion sobre el nodo hijooo

                        alturasNodosHijo <-
 get_nodes_attr(nodoBajo, "height")
                        alturasNodosHijo <-
 alturasNodosHijo[alturasNodosHijo>0]
                        alturasNodosHijo

                        if(length(alturasNodosHijo)>0){
                                while(alturasNodos[1]==alturasNodosHijo[1]){
                                        nodoBajoTemp<-
cut(nodoBajo, h = alturasNodos[1])$lower[[1]]

                                        plot(nodoBajoTemp)

                                        #Obtengo el grupo de nodos de esa clas
e
                                        miembros2<-
nodoBajoTemp %>% get_nodes_attr("label")
                                        #Remuevo valores NA
                                        miembros2 <-
 miembros2[!is.na(miembros2)]



                                        if(length(miembros2)!=1){

                                                #Obtengo el proximo nombre de
nodo

                                                #for (item in listaClasificado
res){
                                                #       print("El valor de l
Item es:")
                                                #       item[1]
                                                #}
                                                nombre<-
sample(1:1000,1,replace=F)
                                                nombreGen<-
paste(c("N",nombre), sep="", collapse="")

                                                        #Verifico que este no s
e encuentre en el listado
                                                        seEncuentra<-TRUE
                                                        while(seEncuentra){
                                                                seEncuentra<-
FALSE
```

```
                                                                cont1<-1
                                                                while(cont1<length(listaClasificadores)){
                                                                    if(as.character(listaClasificadores[cont1])==nombreGen){
                                                                     seEncuentra<-TRUE
                                                                     break
                                                                    }
                                                                    cont1<-cont1+3
                                                                }
                                                                if(seEncuentra)
                                                                {
                                                                    nombre<-sample(1:1000,1,replace=F)
                                                                    nombreGen<-paste(c("N",nombre), sep="", collapse="")
                                                                    #print(nombreGen)
                                                                }
                                                        }


                                                        for(valor in miembros2)
                                                        {
                                                                levels(filtTrainTemp$tipo)[levels(filtTrainTemp$tipo)==valor] <- nombreGen
                                                        }
                                                        salidasPosibles<-append(salidasPosibles,nombreGen)

                                                }else{
                                                        salidasPosibles<-append(salidasPosibles,miembros[1])

                                                }

                                                nodoBajoTemp<-cut(nodoBajo, h = alturasNodos[1])$lower[[2]]

                                                alturasNodosHijo<-get_nodes_attr(nodoBajoTemp, "height")
                                                alturasNodosHijo <- alturasNodosHijo[alturasNodosHijo>0]
                                                alturasNodosHijo


                                                if(length(alturasNodosHijo) == 0){

                                                  alturasNodosHijo[1]=0
                                                }

                                                nodoBajo<-nodoBajoTemp

                                        }
                                }

                                ## ========================================
                                ==============================
```

```
                                ## ========================================
    ==============================

                                salidasPosibles


                                #Obtengo el grupo de nodos de esa clase
                                miembros<-
    nodoBajo %>% get_nodes_attr("label")
                                #Remuevo valores NA
                                miembros <- miembros[!is.na(miembros)]


                                if(length(miembros)!=1){

                                        #Obtengo el proximo nombre de nodo

                                        #for (item in listaClasificadores){
                                        #        print("El valor de l Item es:"
    )
                                        #        item[1]
                                        #}

                                        #Genero la Nueva Etiqueta
                                        nombre<-sample(1:1000,1,replace=F)
                                        nombreGen<-
    paste(c("N",nombre), sep="", collapse="")

                                        #Verifico que este no se encuentre en e
    l listado
                                        seEncuentra<-TRUE
                                        while(seEncuentra){
                                                seEncuentra<-FALSE
                                                cont1<-1
                                                while(cont1<length(listaClasifi
    cadores)){
                                                        if(as.character(listaCl
    asificadores[cont1])==nombreGen){
                                                                seEncuentra<-
    TRUE
                                                                break
                                                        }
                                                        cont1<-cont1+3
                                                }
                                                if(seEncuentra){
                                                        nombre<-
    sample(1:1000,1,replace=F)
                                                        nombreGen<-
    paste(c("N",nombre), sep="", collapse="")
                                                        #print(nombreGen)
                                                }
                                        }


                                        for(valor in miembros){
                                                levels(filtTrainTemp$tipo)[leve
    ls(filtTrainTemp$tipo)==valor] <- nombreGen
                                        }
                                        salidasPosibles<-
    append(salidasPosibles,nombreGen)


                                }else{
```

```
                                        salidasPosibles<-
    append(salidasPosibles,miembros[1])

                                }


                                listaClasificadores<-
    recorrerArbol(nodoBajo,listaClasificadores,nivel=nombreGen)

                        }

                        plot(dendograma)

                        filtTrainTemp$tipo<-factor(filtTrainTemp$tipo)

                        clasRaiz <-
     train(tipo ~., data = filtTrainTemp , method = modeloEntrenamiento , trContro
    l = fitControl)

                        #Creo el objeto clasificador

                        clasificadorObj<-list()
                        clasificadorObj$entrada=nivel
                        clasificadorObj$clasificador=clasRaiz
                        clasificadorObj$salidas=salidasPosibles

                        #Guardo el Clasificador en una lista

                        listaClasificadores<-
    append(listaClasificadores,clasificadorObj)
                        return(listaClasificadores)

                }else{
                        return(listaClasificadores)

                }


        }
```

Anexo 7

Otros Métodos Utilizados.

```
obtenerClasificador <- function(vectorClasificadores,nombreEntrada){
        posicionEntrada=1
        posicion=0
        while(TRUE){
                #print(posicionEntrada)
                #print(vectorClasificadores[posicionEntrada])
                if(vectorClasificadores[posicionEntrada]==nombreEntrada){
                        posicion=posicionEntrada
                }
                posicionEntrada=posicionEntrada+3
                if(posicionEntrada>length(resultClasificadores)){
                        break
                }
        }
        posicion=posicion+1
        return(posicion)

}
obtenerResultadoDeClaJerar <-
 function(filtTesting,predicciones,resultClasificadores){
        print("Fantan:")
        while(sum(predicciones[[1]] %in% etClases == FALSE)>0){
                #print("###############while############################")
                #print(predicciones[[1]])
                #print("Se encuentra en:")
                #print(etClases)

                #Muestro cuantas clases faltande resolver
                faltantes<-sum(predicciones[[1]] %in% etClases == FALSE)

                for(i in 1:length(predicciones[[1]])){
                        #print("      FOR#########################")
                        #print("contador:")
                        #print(i)
                        #print("Si")
                        #print(predicciones[[1]][i])
                        #print("Encuentra en:")
                        #print(etClases)
                        if(predicciones[[1]][i] %in% etClases){
                                #print("entro IF")
                        }else{
                                #print("Entro en ELSE:")

                                posicionCla=obtenerClasificador(resultClasific
adores,predicciones[[1]][i])
                                #print("posicion :")
                                #print(posicionCla)
                                pre<-
predict(resultClasificadores[posicionCla],newdata = filtTesting[i,])

                                #Monitoreo de Progreso
                                if(i%%1000==0){
                                        print(i)
                                }
                                if(i%%9000==0){
                                        #Imprimo vista hasta un numero dado
                                        print(predicciones[[1]])
                                        print(faltantes)
                                }

```

```
58.
59.                                     #print("Preungta")
60.                                     #print("SI")
61.                                     #print(pre[[1]][1])
62.                                     #print("No ESTAS EN")
63.                                     #print(levels(predicciones[[1]]))
64.
65.                                     #print("REEMPLAZO!!")
66.                                     if(!(pre[[1]][1] %in% levels(predicciones[[1]]
    ))){
67.                                             levels(predicciones[[1]]) <-
     c(levels(predicciones[[1]]), as.character(pre[[1]][1]))
68.                                             #print("PONGO EL NIVEL")
69.
70.                                     }
71.                                     #print("PONGO EL VALORE")
72.                                     predicciones[[1]][i]<-
    as.character(pre[[1]][1])
73.                                     #print("Resultado:")
74.                                     #print(predicciones[[1]][i])
75.                                     #print("de")
76.                                     #print(predicciones[[1]])
77.
78.                             }
79.
80.                     }
81.             }
82.
83.             predicciones[[1]] = factor(predicciones[[1]])
84.             return(predicciones)
85.
86. }
```