UNIVERSIDAD CARLOS III DE MADRID

DEPARTAMENTO DE INFORMÁTICA

GRADO EN INGENIERIA INFORMÁTICA

TRABAJO DE FIN DE GRADO

Clasificación Automática del tipo de árbol a partir de las características de sus hojas

Autor: Mario Carmona Benítez

Tutores: Antonio Berlanga de Jesùs & Miguel Ángel Patricio

Colmenarejo, 04 de septiembre de 2014

Universidad Carlos III de Madrid
www.uc3m.es

Agradecimientos

Quiero agradecer este proyecto a:

Mi tutor Antonio Berlanga y co-tutor Miguel Ángel Patricio, por dirigirme muy bien en el proyecto y en la carrera y por estar siempre disponibles cuando los he necesitado.

A todos mis profesores de carrera por darme los conocimientos necesarios para poder realizar este proyecto. En especial a Enrique y José por enseñarme una forma distinta de pensar.

A todos esos genios que han logrado que la tecnología llegue al punto en el que esta hoy en día (Alan Turing, Charles Babbage y muchos otros)

A mis compañeros de clase; Eladio, Alba, Carlos, Diego y David por ayudarme y ser capaces de aguantarme a lo largo de la carrera.

Por último a mi familia y a mi pareja Leticia por su apoyo y ánimos.

Resumen/abstracto

Este proyecto realizara una investigación sobre cuáles son las mejores formas para poder llegar a identificar un árbol de forma automática a través de una única foto. Para ello se analizaran los diferentes patrones o características de las distintas fotos que se pueden obtener de un árbol para identificarlo (hoja, tronco o árbol completo) teniendo en cuenta los grandes factores que afectan al crecimiento de los árboles.

En el trabajo se usaran varios tipos de aprendizaje automático, supervisado y no supervisado y dentro de cada uno se usaran dos modelos distintos, clusterizacion y clasificación, con los que finalmente usaran diferentes algoritmos (redes de neuronas, arboles de decisión, máquinas de soporte vectorial, etc.). Todo esto con el fin de poder evaluar los resultados desde distintos puntos de vista.

Palabras Claves

OpenCV, Inteligencia Artificial, Visión Artificial, Aprendizaje Automático, Hojas, Reconocimiento de Árboles, Supervisado, No Supervisado.

Glosario de términos

Centroide o baricentro de un objeto X perteneciente a un espacio n-dimensional es la intersección de todos los hiperplanos que dividen a X en dos partes de igual n-volumen con respecto al hiperplano. Informalmente, es el promedio de todos los puntos de X.

Inteligencia Artificial: Es un área que a través de ciencias como la informática, la lógica y la filosofía, estudia la creación y diseño de entidades capaces de razonar por sí mismas utilizando como paradigma la inteligencia humana.

Indexar: Ordenar una serie de datos o informaciones de acuerdo a un criterio común a todas ellas para facilitar su consulta y análisis.

Algoritmo: Es un conjunto prescrito de instrucciones o reglas bien definidas, ordenadas y finitas que permite realizar una actividad mediante pasos sucesivos.

Software: Representa toda la parte inmaterial o intangible que hace funcionar a un ordenador para que realice una serie de tareas específicas.

Ruido: En informática, datos sin significado, generados simplemente como subproductos no deseados de otras actividades.

Java: Es un lenguaje de programación de propósito general, concurrente, orientado a objetos y basado en clases.

Pixel: Es la menor unidad homogénea en color que forma parte de una imagen digital, ya sea esta una fotografía, un fotograma de vídeo o un gráfico.

Vector: Es una magnitud física definida por un punto del espacio donde se mide dicha magnitud, además de un módulo, su dirección y su sentido.

Matriz: En programación, una matriz (llamado en inglés array) es una zona de almacenamiento continuo, que contiene una serie de elementos del mismo tipo.

Morfológico: Es tanto la disciplina que estudia la generación y las propiedades de la forma como una rama de la biología que estudia la forma o estructura de los seres vivos.

Peciolo: Es el rabillo que une la lámina de una hoja a su base foliar o al tallo

Nervadura: Es la distribución de los nervios que componen el tejido vascular de la hoja de una planta.

SIGLAS

SVM: Support Vector Machines (en español, máquinas de soporte vectorial).

EM: Expectation maximazacion (en español, esperanza maximización).

KDD: knowledge Discovery in Databases (en español, proceso de extracción de conocimiento).

ANN: Artificial neural networks (en español, red de neuronas artificiales).

RGB: Red, Green, Blue (en español, rojo, verde y azul).

Índice

Índice de ilustraciones

Índice de Tablas

Índice de Graficas

1 Introducción:

1.1 Contexto, objetivos y motivación.

Actualmente utilizamos la tecnología para realizar de forma más fácil muchas de las tareas que se hacen a lo largo del día, desde comunicarnos con otras personas hasta pasando por un seguimiento del ejercicio realizado dicho día. Y de estas tareas cada vez son más las que hacen uso de la inteligencia artificial y sobretodo de la visión artificial para identificar objetos. En la actualidad existen muchas aplicaciones que usan la inteligencia artificial para poder identificar distintas cosas, como puede ser una persona, un objeto concreto, una voz específica, etc. Pero generalmente el uso de esta tecnología se limita a poder identificar objetos concretos o con características muy marcadas (nos referimos por ejemplo a una cara o un cuerpo que siempre seguirán un patrón general; dos ojos, una nariz, una boca, dos orejas, etc.).

Esto que desde el punto de vista de una persona parece muy simple (abstraer patrones y clasificarlos con un sentido específico) desde el punto de vista computacional es muy completo, principalmente por la gran cantidad de variables distintas que puede poseer un único objeto y la cantidad de objetos existentes. A pesar de lo simple que nos parece poder reconocer un tipo de animal, el ojo humano tiene más de dos millones de terminaciones nerviosas que comunican la información recogida al cerebro para después poder interpretarla.

Por otra parte, la utilidad de este proyecto no se limita únicamente a algo estético, sino que al poder extrapolarse a muchos temas distintos (como pueden ser animales, insectos, vehículos) podría suponer una gran ayuda para la población tanto en ámbito profesional, como en lo personal, pues una persona con dificultades visuales o de memoria podría beneficiarse de esta tecnología. En el futuro es muy posible que podamos encontrar todo tipo de herramientas que usen la visión artificial para facilitarnos la vida y puede llegar a suponer un cambio tan grande como en su día fue la aparición del Smartphone (inicialmente como móvil), por lo que no solo supondrá un cambio social, sino que también afectara mucho a la economía global, es por ello que cada investigación sobre este campo, es un paso más hacia ese “futuro”.

Por último, el objetivo de este proyecto es poder identificar objetos abstractos, que no tengan siempre la misma forma idéntica, en nuestro caso se realizara con los árboles. Para alcanzar dicho objetivo se analizaran las distintas fotografías características que se pueden tomar de los árboles; Árbol entero y hoja.

2 Estado del arte

2.1 Visión Artificial

La visión artificial es un subcampo de la inteligencia artificial e incluye métodos de adquisición, procesamiento, análisis y comprensión de una imagen a través de un computador y con ello producir información útil para finalmente poder entender una parte especifica de dicha imagen (un objeto, una característica, un patrón, etc.), de esta forma, el computador es capaz de relacionarse con su entorno, de la misma forma que lo haría un animal o un humano. En consecuencia, el ordenador ganaría la capacidad de "ver", abriendo una amplia gama a la resolución de nuevos problemas como podría ser el seguimiento de una persona u objeto en una secuencia de vídeo (por lo cual de inicio el ordenador ya debería ser capaz de reconocerlo), la estimación de movimiento (por ejemplo predecir el movimiento que hará una pelota en un lanzamiento), también la restauración automática de una imagen, etc.

La visión se compone de dos partes (tanto para los humanos como para el computador); Una primera parte de obtención de la imagen, para la cual los humanos emplean los ojos (órgano con una gran complejidad) y en donde las maquinas emplean las cámaras. La segunda parte consiste en la interpretación de la imagen, en la que hay que distinguir objetos, extraer información de la escena, situar la localización de la imagen, distinguir entre el fondo de la imagen y una parte específica, etc. En los humanos se resuelve gracias al cerebro el cual ha sido entrenado desde el momento en el que abrimos los ojos por primera vez, en cambio, a un ordenador no se le puede entrenar de esta manera dado que necesitaría una gran capacidad, mucho tiempo y un supervisor que asociara las imágenes a la información que debería extraer de ellas. Por esta razón se trata de simplificar este proceso en el computador, para ello primero se acota el mundo en el que va a trabajar dicho ordenador (dejando una única materia como pueden ser árboles, caras de personas, coches, matriculas, formas geométricas, etc.), el siguiente paso es preprocesar la imagen simplificándola, pues el ordenador tendrá más facilidad de identificar un objeto especifico si se elimina el fondo de la imagen, si se resalta o si se elimina el ruido que pueda tener (como por ejemplo la sombra del propio objeto). Por último, una vez se tiene la imagen con el aspecto deseado, gracias a varios algoritmos y técnicas el ordenador intenta comprender de forma automática dicha imagen.

Uno de los grandes problemas de la visión artificial es la cantidad de materias o campos distintos con las que tiene que trabajar, al fin y al cabo se trata de poder interpretar la realidad (el mundo que nos rodea). Es por ello por lo que gran parte del tiempo utilizado en un problema de visión artificial se gasta en documentarse sobre la materia que se va a querer trabajar.

A continuación una imagen sobre algunas de las relaciones de la visión artificial y otros campos.

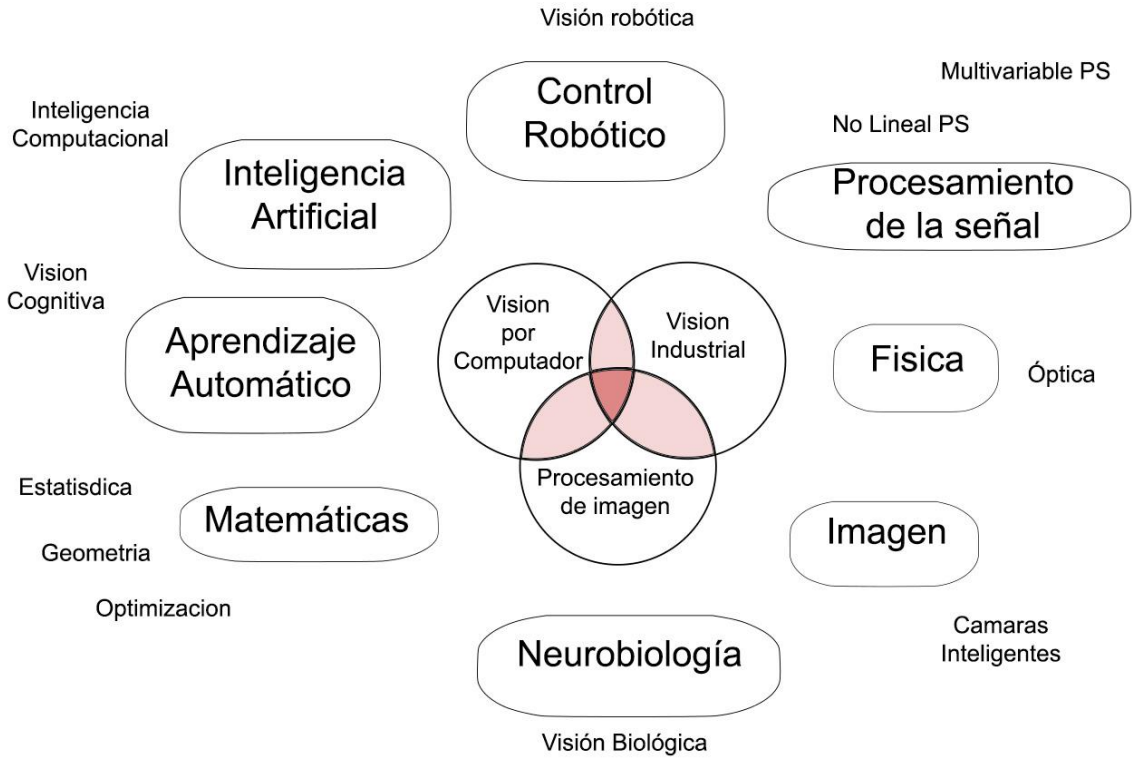


Ilustración 1 - Campos relacionados con la visión artificial

Este campo, está evolucionando constantemente y cada año salen nuevas aplicaciones donde utilizarlo, esto es por su gran cantidad de usos, a continuación expondré algunos ejemplos de aplicaciones en los que se usa la visión artificial.

- Reconocimiento de fotografías de internet. Por ejemplo existe el buscador de *Google Goggles* que es un servicio de Google disponible para Android[1] que permite reconocer cualquier objeto mediante fotos realizadas con un móvil y devolver resultados de búsqueda e información relacionada. Esto funciona con un catálogo de fotos que previamente Google ha catalogado, no funciona con todas las fotos indexadas en internet pues eso actualmente sería imposible.

- Reconocimiento de caracteres, muy utilizado para el escaneo de textos para así poder ahorrarse tiempo a la hora digitalizarlo o para identificación de firmas, también es muy útil para ayudar a los discapacitados visuales para poder leer textos. Hace unos años, estos eran capaces de distinguir solo dentro de uno cierto tipo de fuente, pero con el paso del tiempo y la mejora de la tecnología, ya son capaces de reconocer una amplia gama de fuentes diferentes. Por otro lado, actualmente la seguridad es un campo muy importante en las vidas de todos los ciudadanos, y es en este campo donde el reconocimiento de firmas tiene más prioridad para así poder garantizar la protección de cada persona. El

[1] Android: Es el sistema operativo basado en el kernel de Linux diseñado principalmente para dispositivos móviles con pantalla táctil, como teléfonos inteligentes o tabletas

funcionamiento está basado en el estudio biométrico de las firmas de los individuos a los cuales se les quiere poder identificar más tarde.

- Muy parecido también es el reconocimiento de matrículas utilizado por ejemplo en los aparcamientos o por los ayuntamientos para controlar la entrada y salida de vehículos de forma automática.

Ilustración 2 - Reconocimiento de matriculas

- Como ya se ha comentado anteriormente, la visión artificial está cada día más ligada directamente a la ayuda al sistema visual humano, y no solo para aquellos que tienen un pequeño defecto visual sino que para todos aquellos que han perdido completamente la visión, pues se está desarrollando unos microchips que pueden sustituir a los fotorreceptores y así devolver la visión a muchas personas.

- Además, en la industria se usa para reconocer si en una cadena de producción el resultado final tiene algún defecto o no, esto provoca que la empresa pueda ahorrarse tiempo, dinero y mejorar su productividad.

Aparte de los anteriores ejemplos, también se usa la visión artificial para reconocimiento de rostros, expresiones faciales, de huellas dactilares, de objetos, o para aplicaciones para campos tan diferentes como el aeroespacial (para darles visión a los robots), Automovilismo (para la conducción automática desarrollado por Google), criminología, logística, medicina (detectar posibles canceres), deportes, etc.

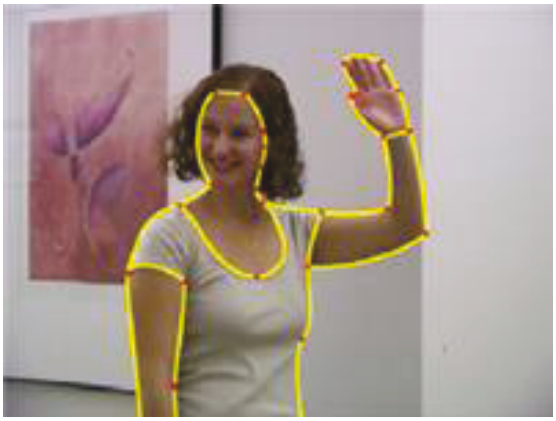


Ilustración 3 - Reconocimiento de silueta

2.2 Minería de datos y aprendizaje automático

La minería de datos

Data mining en inglés, es una etapa del KDD (knowledge Discovery in Databases) en español "proceso de extracción de conocimiento" que se refiere al proceso no-trivial de descubrir conocimiento e información potencialmente útil dentro de los datos contenidos en algún repositorio de información.

La minería de datos esta defina por los autores Clark y Boswell [1] como:

> *El proceso de extracción de conocimiento útil y comprensible, previamente desconocido a partir de granes volúmenes de datos almacenados en distintos formatos.*

A esto agregamos que ha de ser de manera automática o semiautomática. De esta forma, se tiene el objetivo de encontrar patrones repetitivos, tendencias, reglas, secuencias, etc., Con los que poder explicar el comportamiento de los datos en un determinado contexto, esto significa que la minería de datos tiene como objetivo principal encontrar modelos entendibles a partir de ciertos datos.

La minería de datos se utiliza como tecnología de apoyo para la mayoría de aplicaciones con inteligencia artificial (y para muchas otras que no), esto es porque ha resuelto dos grandes problemas, el de trabajar con un conjunto de datos para extraer y descubrir información de interés y la de usar técnicas adecuadas para explorar, comprender, analizar e identificar comportamientos que faciliten una mejor comprensión de los datos [3] [4].

La minería de datos ha tenido un crecimiento tan elevado gracias a varios factores como pueden ser el crecimiento de las bases de datos, la evolución de la potencia de los ordenadores, la fiabilidad del almacenamiento de estos, la disponibilidad de dichos datos a través de internet, la competitividad de las empresas en una economía capitalista y el gran desarrollo de las técnicas y herramientas relacionadas con la minería de datos.

A continuación (ilustración 4) se explica el proceso típico que se suele seguir cuando se trabaja con la minería de datos [2].

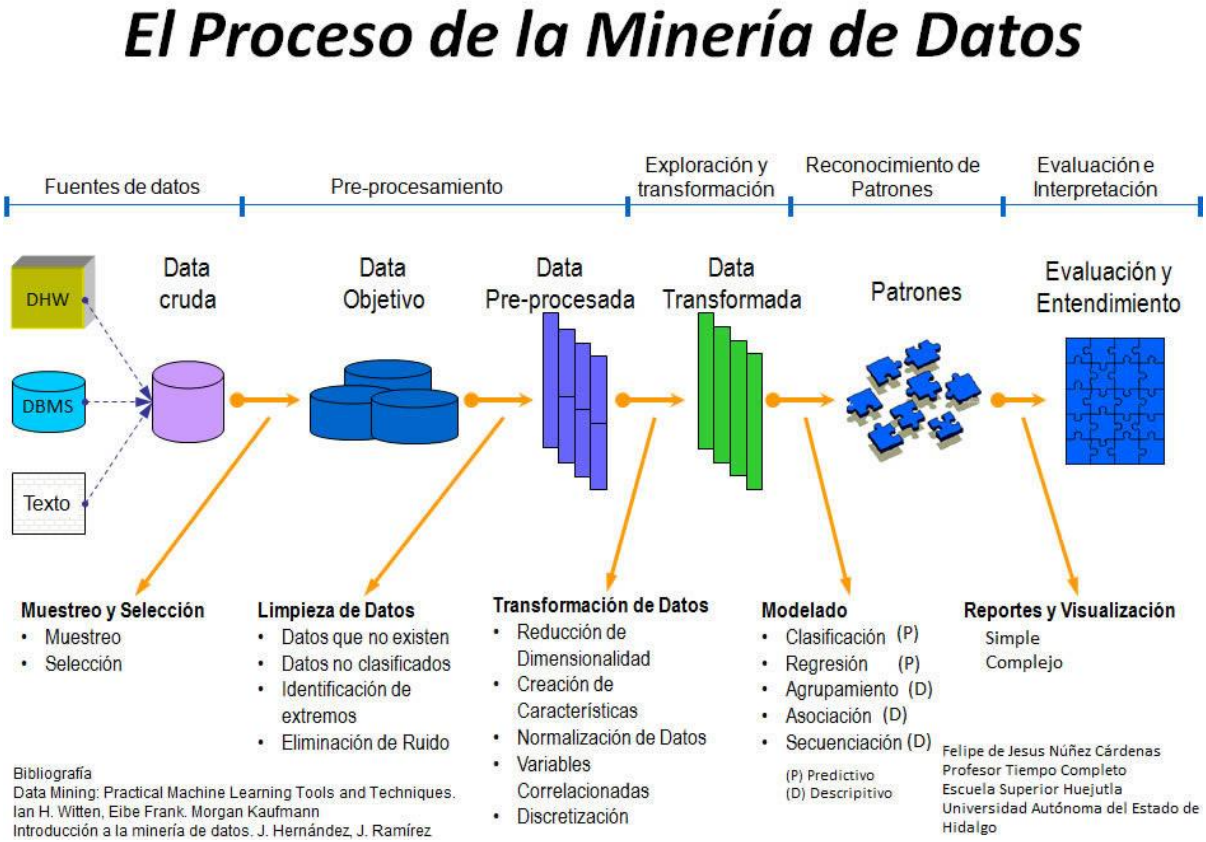


Ilustración 4 - El proceso de la minería de datos

Primero hay que tener una base de datos (muestreo) y así poder seleccionar aquellos con los que vamos a trabajar, cuanta más amplia y rigurosa sea la base de datos mejor se podra trabajar.

El siguiente paso es la transformación de los datos o también llamado el preprocesamiento, primero hay que realizar varios procesos de limpieza, como puede ser eliminación de datos erróneos o que generen ruido, identificación de casos extremos, o incluso saber cuales son los datos contrarios a los que se necesitan. Después se realizan varias transformaciones a los datos con el objetivo de preparar los datos al problema al que nos vamos a enfrentar.

El tercer paso es seleccionar y aplicar la técnica de minería de datos que mejor resultado pueda aportar para el problema, las cuales se explicaran más adelante (clasificación, regresión, agrupamiento, etc.).

A continuación es la extracción de conocimiento o reconocimiento de patrones, que gracias al paso anterior se obtiene un modelo de conocimiento que representa patrones de comportamiento observados en los valores de las variables del problema o relaciones de asociación entre dichas variables. También pueden usarse varias técnicas a la vez para generar distintos modelos, aunque generalmente cada técnica obliga a un preprocesado diferente de los datos.

Por último hay que evaluar e interpretar los datos, comprobando que los resultados se asemejan con la solución que se esperaba, en caso contrario habría que plantear el porqué de esto y si es posible realizar otro modelo que arrojara mejores resultados.

Hay dos tipos de problemas que se pueden resolver con la minería de datos, los predictivos y los descriptivos.

- Predictivos**:** A estos problemas se les denomina como problemas de aprendizaje supervisado, dado que el propio conocimiento proporciona al sistema la respuesta deseada. Dentro de este tipo de problemas se pueden hacer la siguiente distinción: Regresión, clasificación, predicción y Analisis de serie de tiempo.

- Descriptivos**:** Estos problemas suelen ser de carácter no supervisado tratas de describir los datos existentes. Dentro de este tipo de problemas se pueden hacer la siguiente distinción: Clusterizacion (agrupamiento), reglas de asociación, correlación o detección de secuencia y reducción de dimensiones.

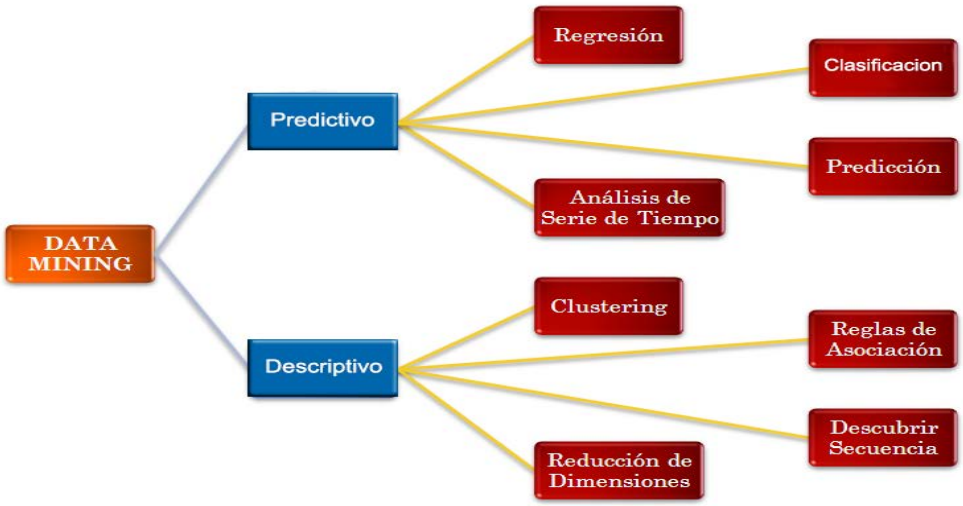


Ilustración 5 - Minería de datos; Predictivo, Descriptivo

El aprendizaje automático

Maching Learning en inglés, es la rama de la Inteligencia Artificial que se dedica al estudio de los programas que aprenden o evolucionan automáticamente basados en su experiencia de acuerdo con alguna medida, para realizar una tarea determinada cada vez mejor. El objetivo principal del aprendizaje es utilizar la evidencia conocida (ejemplos previos) para poder crear una hipótesis y poder dar una respuesta a nuevas situaciones no conocidas.

Esta técnica suele ser usada sobretodo en problemas en los que no existen algoritmos específicos para obtener una solución, o en problemas mal definidos, esto se debe a que el aprendizaje automatico se basa en el conocimiento a través de ejemplos y por tanto se basa en esos ejemplos para obtener la solución. Es por esto por lo que en muchas ocasiones el campo de actuación del aprendizaje automático se solapa con el de la

estadística, ya que las dos disciplinas se basan en el análisis de datos y la obtención de resultados a través de dicho analisis.

En la actualidad el aprendizaje automático se aplica en diferentes entornos, a continuación algunos ejemplos:

- Banca, para saber si conceder un credito o no.
- Domótica[2], para autorregular el consumo de energía.
- Marketing, para los hábitos de compra de las personas.
- Sanidad, para como diagnosticar mejor una enfermedad.
- Procesamiento del lenguaje natural para así analizar la síntesis y morfología de un texto.
- Juegos, para generar bots[3] que sean capaces de ganar el juego a otros usuarios.
- Conducción autónoma de vehículos.
- Centros científicos, para analizar la gran cantidad de datos que tienen.
- Robótica, intentando imitar el comportamiento humano en distintos robots.

Existen varios tipos de algoritmos de aprendizaje automatico: Aprendizaje supervisado, no supervisado y semisupervisado. Aunque en este proyecto solo vamos a usar los dos primeros tipos.

Aprendizaje Supervisado: En este tipo de aprendizaje, los datos de ejemplo con los que se va a realizar el proceso tienen ligados la propia solución que se espera. De esta forma, el objetivo del aprendizaje supervisado es generar y entrenar un modelo para más adelante poder predecir el valor de salida (solución) de los nuevos datos de entrada (en los cuales ya no se tiene la solución).

Para resolver un problema con aprendizaje automatico, se suelen seguir los siguientes pasos, aunque no estrictamente en este orden:

1. Determinar el problema que se quiere resolver y los ejemplos con los que se va a entrenar.
2. Determinar cuál va a ser el vector característica de entrada.
3. Obtener el conjunto de datos de entrenamiento, (anteriormente descrito en el apartado de la minería de datos)
4. Determinar la técnica de aprendizaje automática que se va a usar
5. Entrenar el modelo y comprobar los resultados.

En este proyecto con el aprendizaje supervisado se usaran dos tipos de modelos diferentes, clusterización también llamado agrupación o clasificación, entre los que se explicaran la red de neuronas (perceptrón multicapa), SVM (maquinas de vectores de soporte), arboles de decisión, y clasificador bayesiano (Naïve Bayes).

[2] Se llama domótica al conjunto de sistemas capaces de automatizar una vivienda
[3] Bot es un programa informático, imitando el comportamiento de un humano

Clusterización: Es la acción de agrupar los datos por clusters. Que se explicara más adelante en la sección de aprendizaje no supervisado, aunque también pertenece al ámbito del aprendizaje supervisado.

Clasificación: la cual es la acción o el efecto de ordenar los datos por clases.

- **Árbol de decisión**: Un árbol de decisión tiene entradas las cuales son una situación descrita por medio de un conjunto de atributos y a partir de esto devuelve una respuesta. Los valores que pueden tomar las entradas y las salidas pueden ser valores discretos o continuos, los cuales supondrán que algoritmos se pueden usar para cada problema. Un árbol de decisión indica las acciones a realizar en función del valor de una o varias variables. Es una representación en forma de árbol cuyas ramas se bifurcan en función de los valores tomados por las variables y que terminan en una acción concreta.

  A continuación un pequeño ejemplo del funcionamiento de un árbol de decisión.

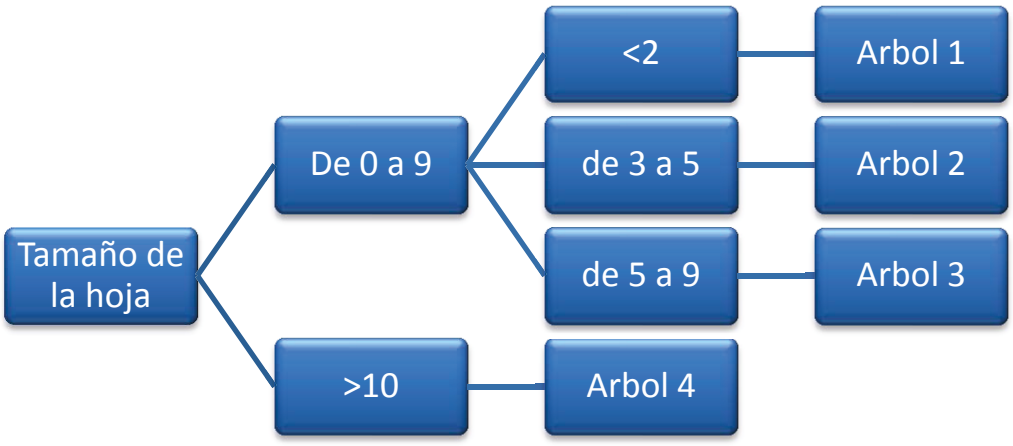


Ilustración 6 - Ejemplo árbol de decisión

Dentro de esta catergoría se utilizara el algoritmo J48 el cual es una implementación libre en Java del algoritmo C4.5, el cual es una extensión del algoritmo ID3, ambos desarrollados por Ross Quinlan [5]. Este modelo intenta encontrar el árbol más sencillo capaz de separar mejor los ejemplos, este algoritmo construye el árbol basándose en el concepto de la entropía de información[4], en cada nodo del árbol, el algoritmo elige un atributo de los datos con el cual pueda dividir más eficientemente el conjunto de muestras en subconjuntos. El algoritmo divide recursivamente en subconjuntos más pequeños hasta que dicho subconjunto no aporta nada nuevo al dividirlo. Para no sobreadecuarse el árbol a los ejemplos de entrada, al árbol se le realiza una poda en caso de que haya generado muchos nodos hojas.

[4] La entropía es la cantidad media de información que recibimos con cada mensaje, siendo este, un evento, un atributo, una muestra, etc.

La razón por la que se ha elegido el J48 en vez del ID3 o C4.5 es porque el J48 es capaz de trabajar con valores continuos.

- **Red de neuronas**: Según Dr. Haykin, S.[6] [7].

> *"Una red neuronal es un procesamiento distribuido masivamente paralelo que tiene una tendencia natural para almacenar conocimiento empírico y hacerlo disponible para el uso."*

Para poder entender las redes de neuronas artificiales primero hay que entender las biológicas. En 1888 Ramón y Cajal[36] descubridor de la estructura celular de la neurona defendío que las neuronas se interconectaban entre si de forma paralela en vez de en circuito.
Una neurona cerebral puede recibir unas diez mil entradas y enviar a su vez su salida a cientos de neuronas, a esto se le llama sinapsis.

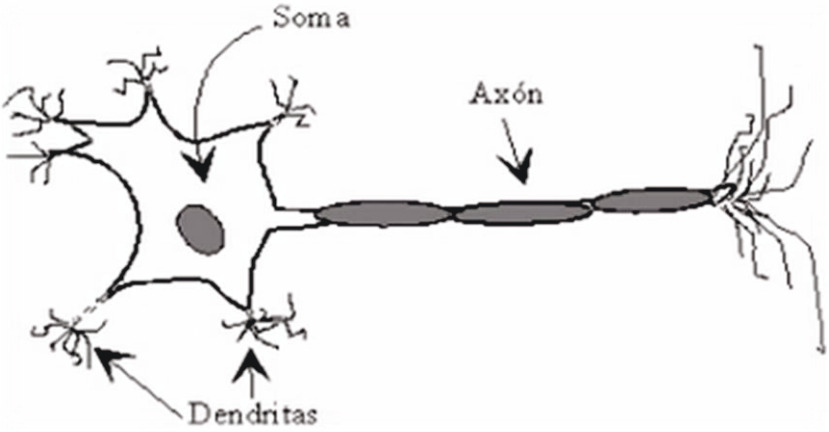


Ilustración 7 - Neurona biológica

La neurona consta de un cuerpo celular (soma) del que surge un árbol de ramificaciones (dendritas) y una conexión llamada axón. Estas partes son similares al de un procesador de información simple, donde el canal de entrada son las dendritas, el procesador es el soma y el canal de salida es axón.

Estas neuronas por separado no pueden realizar grandes tareas pero en conjunto y bien entrenadas son las que dan a los seres vivos la capacidad de pensar, moverse, oler, sentir, ver, etc.

Por otra parte, la neurona artificial imita el funcionamiento de la neurona biológica. La neurona se compone de un conjunto de entradas x de n componentes, las cuales tienen ligadas unos pesos (w_{ij}) que irán modificando su valor según el entrenamiento o experiencia de la red neuronal (hay que tener cuidado de no sobreespecializar a las redes de neuronas porque si no los pesos pueden ajustarse exclusivamente a los ejemplos de entrada). Estas están unidas al cuerpo de la neurona, que se compone de varios elementos, el primero es la regla de

propagación que suele consistir en la suma ponderada del producto escalar del vector entrada y el vector de los pesos.

$$hi(t) = \sum w_{ij} x_j$$

La otra regla más usada es la distancia euclidia entre ambos vectores.

$$hi(t) = \sum (x_j w_i)^2$$

El segundo elemento del que se compone es la función de activación, que si el resultado de entrada cumple los requisitos, la neurona se activara y proporcionara una salida en función del estado de activación.

Ilustración 8 - Neurona artificial

En conjunto las neuronas se distribuyen en capas paralelas y es a lo que se le llama red de neuronas artificiales (ANN)[8] [9] [14] y están formadas por las siguientes capas:

- Capa de entrada, formada por las neuronas que reciben los datos de entrada.
- Capas ocultas, son las capas que se situan entre la capa de entrada y la capa de salida.
- Capa de salida, que son las que proporcionan el valor de salida y por tanto el resultado de la ANN.

Cuando la ANN esta formada por múltiples capas, toma el nombre de perceptrón multicapa.

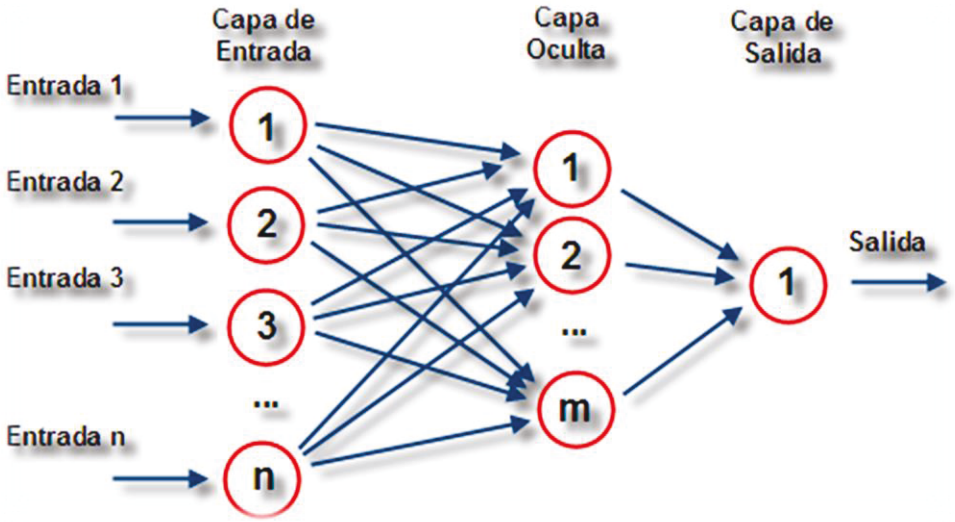


Ilustración 9 - Red de neuronas multicapa

Los pasos a seguir para el aprendizaje de la ANN son:

1. Inicialización aleatoria de los pesos.
2. Se toma un patrón de entrada-salida.
3. Se calcula la salida de la red.
4. Si la clasificación es correcta, no se modifican los pesos.
   Si la clasificación es incorrecta, se modifican los pesos.
5. Se vuelve al paso 2 hasta completar el conjunto de patrones de entrenamiento.
6. Se repiten los pasos anteriores hasta alcanzar el criterio de parada.

- **Clasificador Bayesiano:** Un clasificador bayes ingenuo (naïve bayes) se basa en la distribución probabilística fundamentada en el teorema de Bayes enunciado por Thomas Bayes [10]:

$$P(A_i|B) = \frac{P(B|A_i)P(A_i)}{P(B)}$$

Donde $P(A_i)$ son las probabilidades a priori, $P(B|A_i)$ es la probabilidad de B en la hipóteis A_i y $P(A_i|B)$ son las probabilidades a posteriori.

Se podría traducir como que las características de evento/objeto son idenpendientes de las demás características y que cada una de estas contribuye de forma independiente a la probabilidad de que pertenezcan a dicho evento/objeto.

- **Máquinas de Soporte Vectorial (SVM)**[12] son un conjunto de algoritmos desarrollados por Vladimir Vapnik[11], que representan los puntos de muestra (los ejemplos de entrenamiento) en el espacio, y separando las clases lo máximo que pueda entre ellas, para que cuando se quiere clasificar una nueva muestra, se comprueba cuál es la clase que tiene una mayor proximidad a esta. Para poder

realizar esta tarea el SVM construye hiperplanos en un espacio de dimensiones muy alta que es donde colocara las distintas clases.

A continuación un ejemplo de dos dimensiones

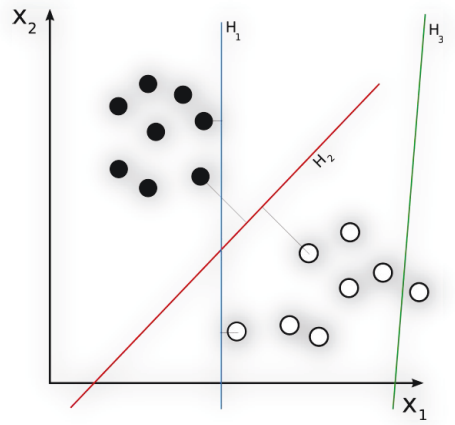


Ilustración 10 - Ejemplo, SVM de dos dimensiones

Los modelos basados en SVMs están estrechamente relacionados con las redes neuronales y en comparación los SVM suelen ser mejores, y entre las diferencias encontramos las siguientes:

- El svm tiene un entrenamiento menos costoso y más eficiente.
- Es muy robusto a la hora de generalizar, realizando menos heurística.
- Los SVM transforman el espacio es muchas más dimensiones.
- Las ANN tienen un espacio de búsqueda con múltiples mínimos locales (pudiendo impedir encontrar soluciones mejores) mientras el SVM solo tiene un único mínimo local.
- Se puede escoger la función de kernel y el parámetro de coste C.

A pesar de que parezca que el SVM es mucho mejor, esto no significa que siempre vaya a obtener mejores resultados, pues ambos clasifican muy eficientemente y ambos funcionan muy bien en problemas tipicos.

La manera en la que se separa un plano puede ser fácil de realizar mediante una linea, un plano o un hiperplano de n dimensiones. Pero en la practica en problemas reales, esta forma de dividir el plano no es tan fácil, siendo por ello por lo que sé recure a la función Kernel. Existen varios tipos, aunque los más usados son los siguientes:

- Polinomial $K(X_i, X_j) = (X_i X_j)^2$

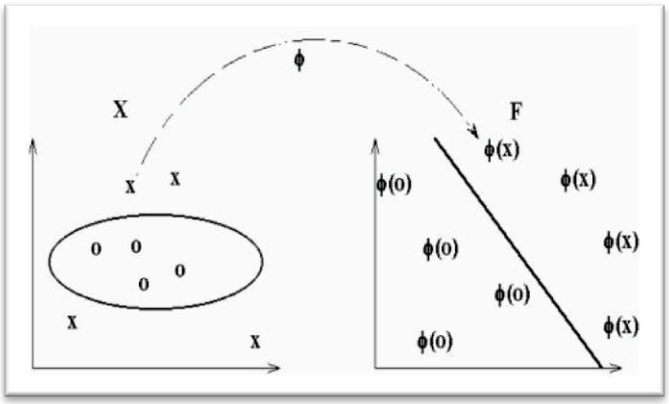


Ilustración 11 - Función de kernel polinomial

- Perceptrón $K(X_j, X_j) = \|X_i - X_j\|$

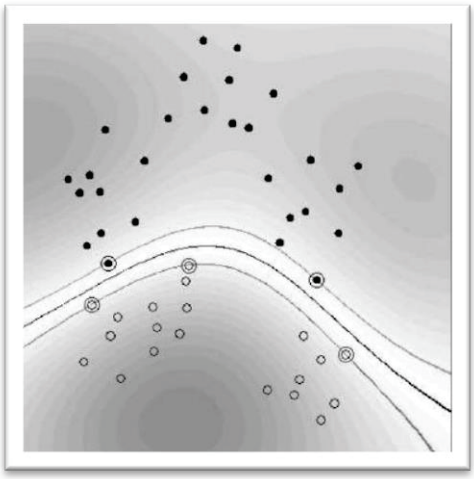


Ilustración 12 - Función de kernel perceptrón

- Función de base radial gaussiana $K(X_i, X_j) = exp\frac{(-(X_i - X_j)^2}{2(sigma)^2}$

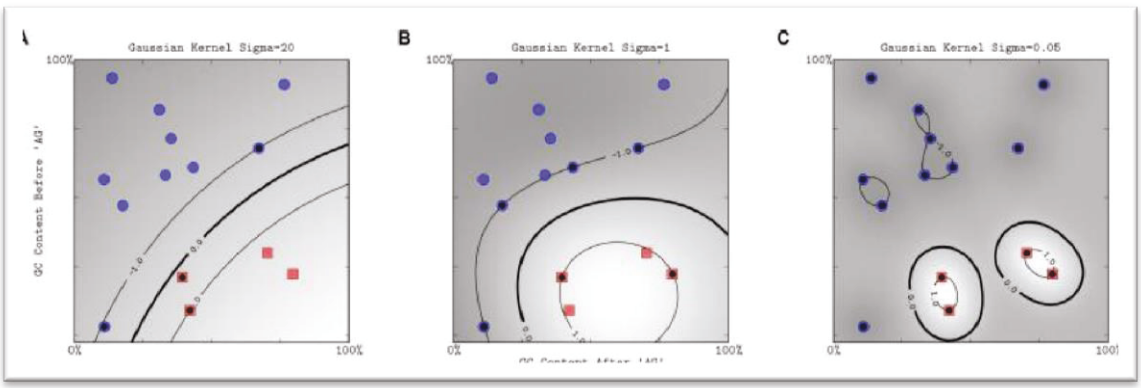


Ilustración 13 - Función kernel de base radial gaussiana

- Sigmoid $K(X_i, X_j) = \tanh(X_i X_j - \theta)$

Aprendizaje No Supervisado*:* Este aprendizaje se distingue del supervisado porque no existe un conocimiento a priori, o sea sé, los datos de entrada no tienen ligada la solución. Es por esto por lo que el aprendizaje no supervisado es útil para la compresión de los datos, por ejemplo, si en un problema sabemos cuál es la solución y se quiere comprobar que el vector característica generado es optimo, se puede realizar las pruebas con el aprendizaje no supervisado para ver si esos datos los agrupa en tantas clases diferentes como las existentes en el problema. Generalmente se emplea la agrupación (clustering) que ya se ha mencionado anteriormente, a continuación se verán algunos algoritmos de clusterizacion.

- **K-Medias,** es un método de clustering presentado por MacQueen [13] en 1967 en el que intenta particionar en K clusters (agrupaciones) los distintos datos de entrada, los cuales pertenecerían al cluster más cercano.

  Su funcionamiento se puede dividir en cuatro pasos:

  - Se eligen K ejemplos (datos) que actúan como semillas de k clases
  - Cada ejemplo del resto se añade a la clase más similar (más cercana).
  - Cuando se termina, se calcula el centroide de cada clase, que pasan a ser las nuevas semillas.
  - Se repite hasta que se llega a un criterio de convergencia.

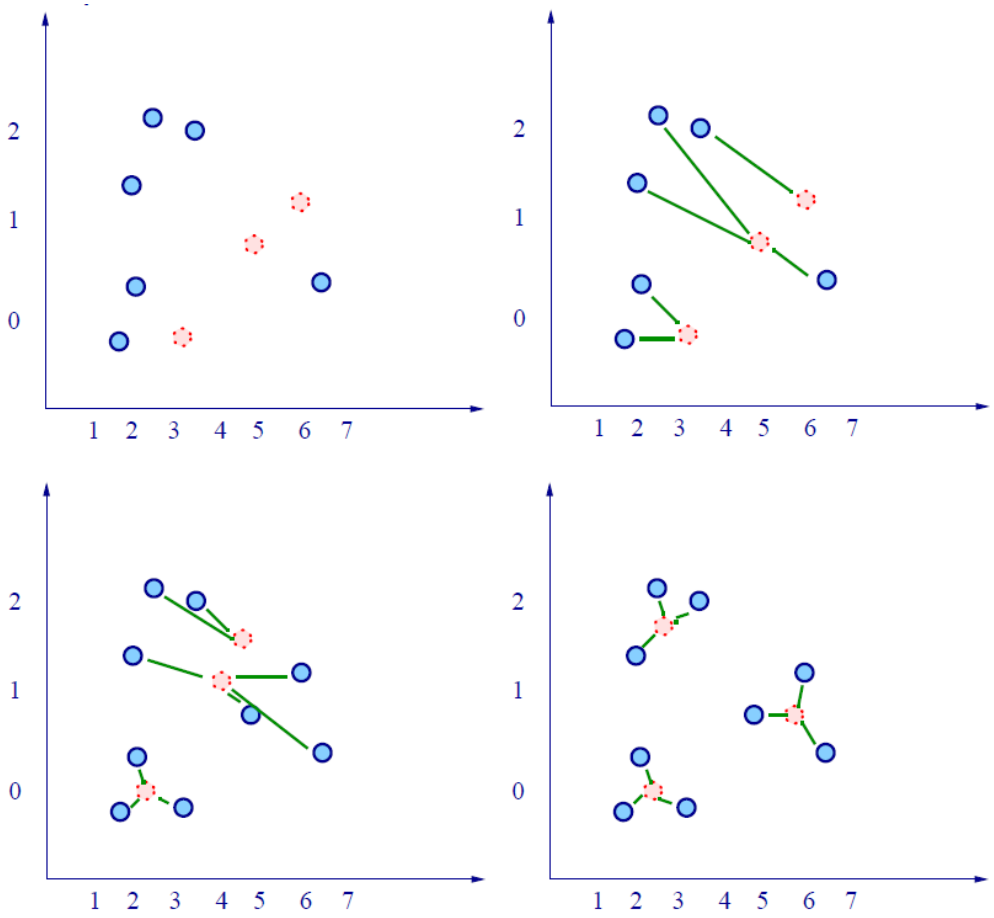


Ilustración 14 - Ejemplo clusters K-medias

- **Cobweb,** fue inventado por el profesor Douglas H. Fisher [15] y es un sistema progresivo jerarquica en el que organiza incrementalmente las observaciones en un árbol de clasificación. Cada nodo del árbol representa una clase y se etiqueta por un concepto probabilístico que resume las distribuciones atributo-valor de los objetos clasificados en el nodo. Cada nodo puede fusionarse con otro, o puede dividirse, generando al final un número concreto de clusters que dependen de varios factores como el acuity (agudeza) y el corte.

- **Esperanza-Maximización (EM),** este algoritmo fue expuesto por Arthur Dempster, Nan Laird y Donal Rubin [17], en 1977, aunque anteriormente ya había sido usado por otros autores. Es un algoritmo iterativo de carácter general para la estimación máximo verosímil en problemas con datos incompletos pues depende de variables no observables. El rango de problemas que puede abordar es muy amplio.

  El algoritmo EM empieza adivinando los parámetros de las distribuciones y los usa para calcular las probabilidades de que cada objeto pertenezca a un cluster y usa esas probabilidades para reestimar los parámetros de las probabilidades, hasta converger (se puede empezar adivinando las probabilidades de que un objeto pertenezca a una clase) [16].

Otras técnicas: Por otra parte, por encima del aprendizaje supervisado y no supervisado, se usaran un conjunto de técnicas que usaran los algoritmos anteriormente explicados para intentar obtener mejores resultados. A esto se le llama conjunto de clasificadores, y su objetivo es el combinar las decisiones individuales de cada clasificador en una respuesta común y conseguir así una solución más precisa [18].

Entre estas técnicas se encuentra el Bagging, el Stacking, el Bosting y muchas más, en el proyecto se experimentara con la mayor cantidad de algoritmos posibles para obtener la mejor solución.

- El Boosting creado en 1990 pretende mejorar la precisión de clasificadores débiles. Este método se especializa en volver a clasificar los ejemplos mal clasificados anteriormente para después combinar la decisión de los clasificadores por voto ponderado.

- El Bagging utiliza réplicas del conjunto de entrenamiento (divide el conjunto de entrenamiento en varios conjuntos) y genera un clasificador para conjunto para más tarde combinar las decisiones de los clasificadores por voto mayoritario

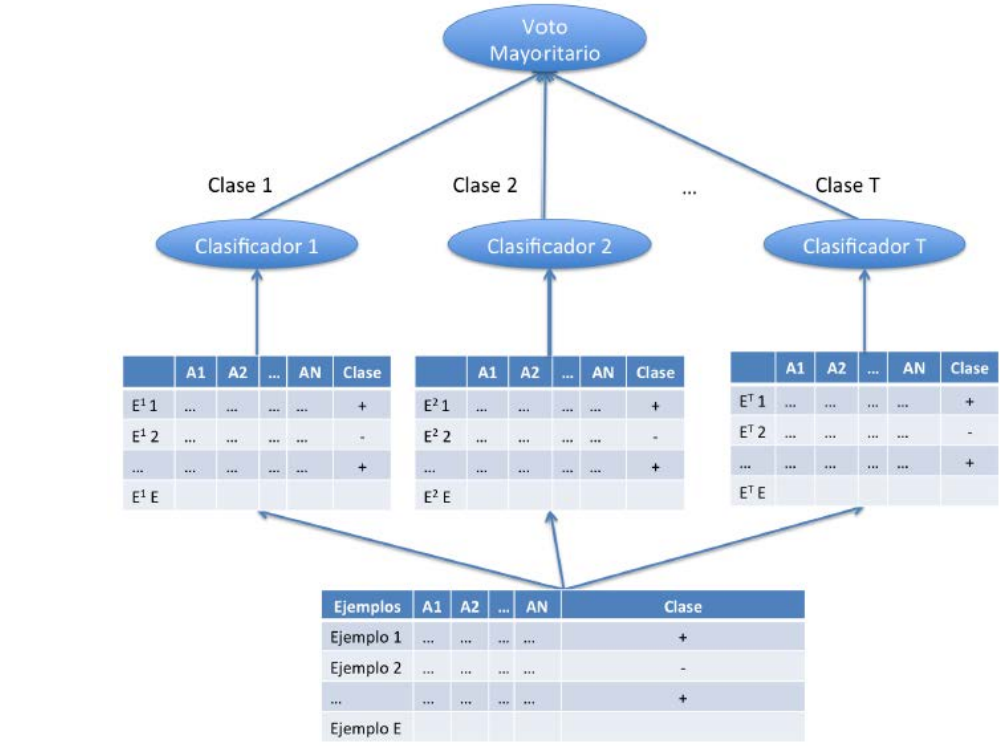


Ilustración 15 - Ejemplo técnica bagging

- El Stacking va un paso más allá que el bagging o el boosting, pues lo que realiza es generar distintos clasificadores (con distintos algoritmos) para obtener la solución de cada uno de ellos y más tarde combinar su solución con un clasificador superior (llamado metaclasificador).

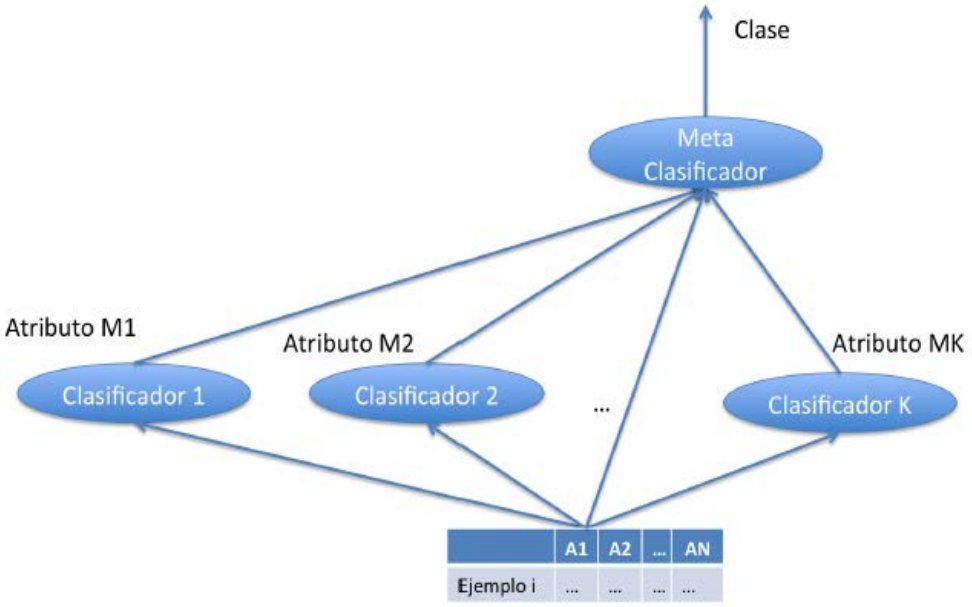


Ilustración 16 - Ejemplo técnica stacking

2.3 Reconocimiento de formas (Proyectos similares)

En este apartado se hablaran de otros proyectos que han trabajado sobre el mismo tema, el reconocimiento de árboles a través de una fotografía.

No se ha encontrado ningún caso que hayan intentado realizar este proyecto con la foto general de un árbol (el apartado 3.3.6 se centra en ese problema) en cambio, en el reconocimiento de árboles a través de la imagen de una hoja sí que existen varios proyectos.

El más destacado, no por sus resultados sino por su amplitud es la aplicación para IPhone *Leaf Recognition* [19], que es un proyecto entre tres universidades de Estados Unidos (Columbia University, University of Maryland y Smithsonia Institution). Tienen un conjunto de datos muy amplio (más adelante se hablara de él), con más de ciento ochenta especias de árboles distintos, aunque solo trabajan con fotos de algunas regiones de América. Se desconocen la exactitud de sus resultados y las técnicas utilizadas.

Un proyecto parecido y con buenos resultados es el de *Flavia* [20], desarrollado en Matlab[5] es un software libre que haciendo uso de una red de neuronas probabilística tiene como objetivo identificar varias especies de árboles asiáticos a través de una fotografía de la hoja. La precisión de este proyecto ronda el 93% y su conjunto de datos tiene más de treinta clases distintas.

Se encontró un proyecto en internet que también trata de reconocer arboles a través de la imagen de una de las hojas. Con quince clases distintas sus resultados son muy cercanos al 100% de acierto utilizando la técnica de SVM vista anteriormente. Uno de los problemas encontrados en este caso es la poca documentación que hay sobre esta investigación, pues no se ha encontrado persona ni identidad ligada a dicho proyecto [21].

Otro proyecto interesante es el *Neuronal Network based recognition system of leaf images* [22] que empezó a desarrollarse en el 2001, aunque no hay actualizaciones desde 2006. Este proyecto es interesante porque la forma en la que trata de identificar el árbol al que corresponde la imagen de la hoja es diferente al resto de estudios. En este caso, se centra en analizar el borde de la hoja para obtener información directamente de ese borde en vez de en la forma generar de la hoja. No se sabe cuáles fueron los resultados obtenidos, aunque en alguna de las imágenes de muestra del proyecto se puede intuir que sus resultados dependían mucho de cada hoja.

[5] **Matlab** es una herramienta de software matemático que ofrece un entorno de desarrollo integrado con un lenguaje de programación propio

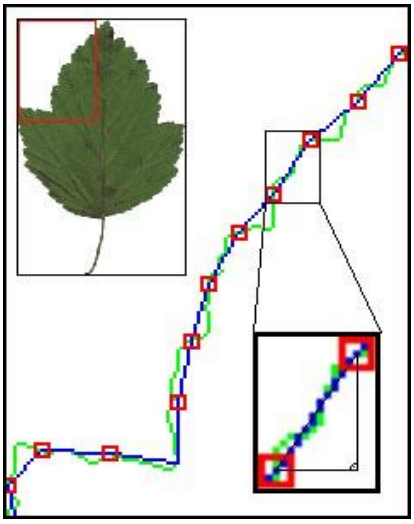


Ilustración 17 - Neuronal Network based recognition system of leaf images [22]

Por último, existen diversos proyectos basados en Sistemas Expertos[6], para intentar identificar árboles, pero este método necesita de las respuestas de una persona para poder dar un resultado final, por lo que aun teniendo buenos resultados se aleja mucho del objetivo del proyecto.

2.4 Características morfológicas de los arboles

En esta sección se analizara los distintos árboles que podríamos llegar a distinguir y las características más destacadas con las que se pueden llegar a distinguirlos a simple vista, con el fin de comprender mejor el problema y su complejidad.

Lista de árboles que se pueden identificar visualmente

Hay que tener en cuenta que en todo el mundo hay más de 60.000 tipos de árboles diferentes, pero que en España se pueden reducir a 80 básicos (visualmente) de los cuales la gran mayoría son subclases de estos mismos, por lo que se puede reducir a un total de 30 árboles. De la lista que se obtiene, hay varios árboles que son muy parecidos entre ellos y que realmente incluso a simple vista es muy costoso poder distinguir, por lo que lo mejor es considerarlos como uno solo árbol.

Nº	Árbol	Descripción/Parecido/Problema/Comentario
1	Abedul común Pubescente Álamo	Hay muchos tipos diferentes de abedules. Estos árboles se diferencian por el tronco. El álamo y abedul son difíciles de diferenciar a simple vista.
2	Abeto blanco Pinsapo	Hay muchos tipos de abetos parecidos entre ellos.
3	Acebo	De naturaleza extraña en España.
4	Olivo	Hay diferentes tipos pero se distinguen por el fruto y tamaño de hoja.

[6] Los sistemas expertos intentan emulan el razonamiento de un experto en un dominio concreto a base de preguntas.

5	Alcornoque	
6	Algarrobo	
7	Aliso común	
8	Almez	
9	Arce	Hay muchos tipos diferentes de arces. Lo más distintivo es su semilla y sus hojas
10	Avellano	
11	Carpe	Muy parecido al haya y pertenece a la familia del abedul
12	Castaño	Hojas y flores muy características.
13	Chopo	Podría confundirse con álamo o abedul si no fuera porque las hojas cubren el tronco (tronco vestido)
14	Cedro	Hay varios tipos y uno de ellos tiene forma parecida al del abeto
15	Cerezo	De los más fáciles de distinguir gracias al color de las hojas
16	Prunus	Es un tipo de cerezo pero de color rojo
17	Ciprés	Muy típico y muy distinguible a simple vista.
18	Encina Quejigo	Es muy difícil poder distinguirlos a simple vista. Pero con la hoja se distinguen fácilmente.
19	Enebro	Se puede confundir con un tipo de abeto
20	Fresno	Hay muchos tipos de fresnos parecidos entre ellos.
21	Haya	Son muy grandes y cambian mucho en otoño
22	Laurel	Lo más característico es la hoja y su olor
23	Lluvia de oro	De color completamente amarillo
24	Mimosa	Verde pero prácticamente cubierto de amarillo cuando está en flor. La hoja es muy característica
25	Nogal	
26	Olmo	
27	Palmeras	Fácilmente distinguibles
28	Pino piñonero	
29	Pino silvestre	
30	Roble	
31	Sabina	Igual que el enebro pero de mayor porte
32	Sauce	Inconfundible por su forma y sus hojas son características

Tabla 1 - Lista de árboles más comunes en España

Poder identificar esta lista no es el objetivo del proyecto, pero sirve para comprender mejor la complejidad del proyecto.

Características de los arboles

En esta sección analizaremos cuales son las distintas características morfológicas de los arboles extraíbles de una fotografía (que no son las mismas que usaremos para más adelante poder clasificarlos). También analizaremos los pros y los contras de estas características y cómo podríamos tratarlas desde un punto de vista computacional y en caso de alguna fuera viable usarla para el proyecto.

Este análisis se hará sobre dos tipos de fotografías, primero sobre una imagen de un árbol entero y posteriormente sobre la imagen una simple hoja.

Árbol Entero: [24] Si queremos identificar un árbol con una foto general y única es necesario poder fotografiar todo el árbol, a poder ser con la mejor iluminación posible y no hacer la foto a contraluz. Uno de los problemas que se puede encontrar al realizar la foto es que esta contenga otros árboles detrás, generando un ruido en la muestra.

Ya centrándose en como poder distinguirlos, se han encontrado tres características que sobresalen del resto.

Color: El color es una de las características más distinguibles que puede poseer un árbol.

Si se incluye esta característica se podrían identificar ciertos arboles de forma más sencilla (llamativos), pero existen dos problemas principales, con las estaciones, un mismo árbol puede cambiar de color sus hojas. Y dependiendo de cómo este la luz, el color puede parecer uno u otro.

Ilustración 18 - Efecto estaciones del año

Para poder utilizar esta característica se podrían agrupar los colores en 4 o 5 colores básicos (verdes, amarillos, rosas, rojizos) de esa forma nos evitamos que la luz pueda cambiar de tonalidad el árbol y sería más fácil identificar a los árboles de hoja perenne. Por otra parte, la extracción de esta característica sería muy fácil usando el histograma. También podríamos ayudarnos de la fecha en la que se tomó la fotografía para evitar las estaciones. Pero no es una característica muy importante para poder llegar a diferenciar un árbol de otro por lo que más adelante no la utilizaremos.

En las siguientes fotos, presento cuatro arboles con colores muy diferentes entre ellos.

Ilustración 19 - Cerezo – Lluvia de oro/Mimosa – Encina – Prumus

Tronco: Hay que tener en cuenta dos tipos generales de troncos.

Ilustración 20 - Tipos de troncos

Tronco Vestido: Son aquellos árboles en los que las hojas nos impiden ver su tronco (no tiene por qué cubrir la totalidad del tronco pero si la gran mayoría de este)

Tronco Desnudo: Son aquellos árboles en los que se puede ver el tronco con bastante facilidad dado que la copa del árbol no lo cubre.

El tronco no tiene relación con la forma o tipo de árbol sino con la edad, cuanto más viejo sea el árbol más grueso es el tronco, por lo que tenemos que descartar una relación anchura (tronco), altura (árbol).

Hay pocos árboles que sean distinguibles únicamente por su tronco, por lo que se puede asegurar que el proyecto no sería viable usando solo la foto del tronco.

Forma o silueta: Generalmente los arboles (y arbustos) poseen siluetas distintivas y están definidas por la forma de la copa, en las que se pueden distinguir 6 básicas, aunque hay un problema notable en esta característica, pues tras consultar en varios libros, se puede asegurar que la forma de un árbol también depende de factores como pueden ser, si este pertenece a un bosque o está solo, si se le ha podado a menudo o si por el contrario no ha recibido poda alguna y por último cuanta luz y por donde le llega al árbol.

Es por esto por lo que computacionalmente el proyecto debería en identificar únicamente las copas de los árboles que se encuentren en solitario (sin pertenecer a un bosque o tener muchos más árboles a su alrededor), lo cual reduciría mucho las posibilidades de identificación de los árboles.

Formas de Árboles (tipo y ejemplos)
Globosa: Plátano / Catalpa / Olivillo

Elipsoidal: Árbol de cielo / Haya / Liquidámbar

Columnar: Criptomeria / Álamo lombardo / Tuya

Colgante: Abedul péndulo/ Sauce llorón / Aguaribay

Triangular: Cedro Deodara / Castaño de las Indias / Sequoia

Sombrilla: Pino piñonero/ Paraíso / Castaño

Ilustración 21 - Tipos de copas

Sin duda alguna, si queremos identificar un árbol con una sola foto general, esta es la característica más importante, la cual si la juntamos con las tonalidades básicas del color y los dos tipos de troncos ya explicados se pueden identificar la mayoría de los arboles típicos.

Hoja: Los problemas que se encuentran a la hora de buscar características morfológicas en las fotos de las hojas son la gran diferencia que puede existir entre unas hojas u otras, pues hay casos en los que las hojas se componen de forma individual, en otras en forma de ramillete e incluso otras se componen de varias hojas en si (como es el caso de la mimosa).

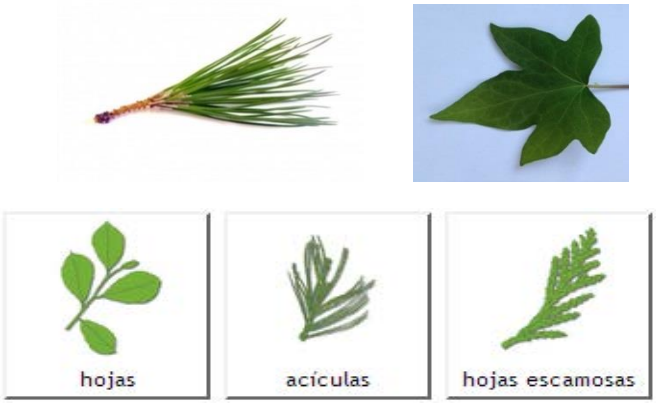


Ilustración 22 – Ejemplo tipos de hojas

Para solucionar este problema, las hojas se analizaran desde la forma individual. En el caso de tener que analizar una hoja de estructura compuesta, se deberá fotografiar la mínima expresión de la composición (refiriéndose a hojas).

También es un problema la orientación en la que se toma la foto, pues no es lo mismo una foto con el peciolo hacia un lado o hacia otro.

Aun con esto, se pueden distinguir varias características a simple vista.

Color: Al igual que con la foto del árbol general, puede ser una característica que nos ayude a distinguir ciertas especies, pero en este caso, al realizar las fotos de forma más específica (a una única hoja de todo el árbol) se acentúa aún más el hecho de que los árboles cambien de tonalidades a lo largo de las estaciones.

Por otra parte, hay que tener en cuenta el deterioro de la hoja desde el momento en el que sé extrae del árbol, pues poco a poco y por culpa de la perdida de agua, la hoja puede cambia de color y en ocasiones de forma.

Ilustración 23 - Cambio de color en hojas

En la ilustración 23 se puede observar el cambio de tonalidad de la hoja de un árbol a lo largo de las estaciones y en la ilustración 24 se puede distinguir el cambio de color con el deterioro de la hoja.

Ilustración 24 - Deterioro de la hoja

Forma: Las hojas tienen tres características en cuanto a su aspecto; La forma, el margen (borde) y la venación [23].

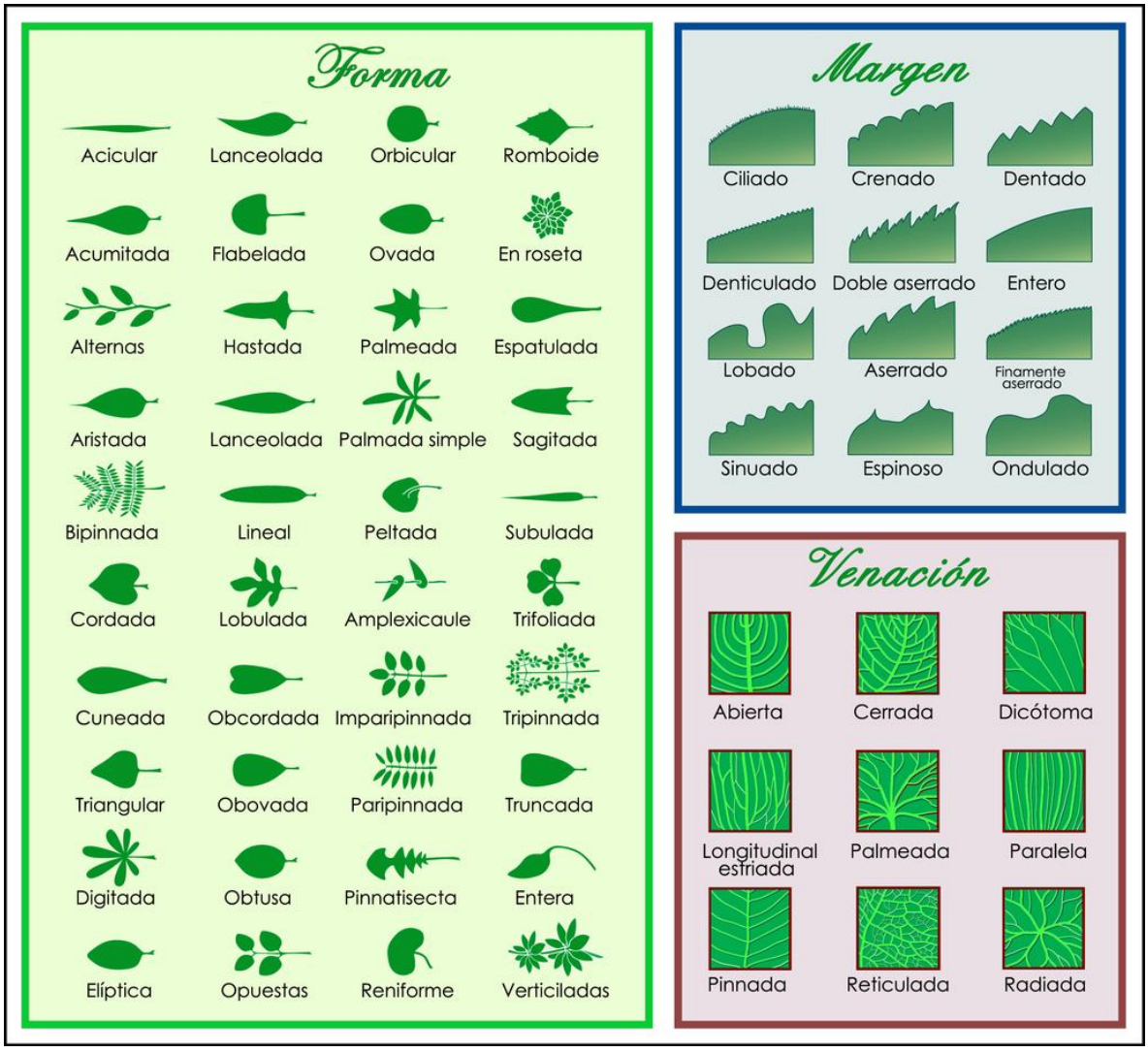


Ilustración 25 - Morfología de las hojas

Como se puede ver en la ilustración 25, hay muchas de las formas que son realmente muy parecidas (como por ejemplo la ovada y la obtusa) es por eso por lo que a la hora de extraer esta característica lo mejor es reducir la lista. Lo mismo ocurre con el margen (crenado y sinuado) por lo que también habría que reducir el número de posibilidades. Esta no es la única forma de clasificar las formas de las hojas pues existen otras variables en las que se tiene en cuenta por una parte la forma de la parte superior de la hoja (Ápice) y por otra la forma de la base de esta.

Por otra parte, encontramos la venación, que puede ser una característica muy importante a la hora de diferenciar entre la hoja de un árbol u otro, pero aquí nos encontramos que para poder distinguirlo es posible que para algunas hojas la imagen tenga que tener una alta calidad, además de que aun teniendo dos hojas con la misma forma y margen y que solo se distinga por la venación es muy extraño por lo que no es de mucha importancia si esta última característica no se tiene en cuenta.

3 Desarrollo

3.1 Herramientas y tecnologías aplicadas

Para poder realizar el proyecto, se han tenido que utilizar varias tecnologías y herramientas diferentes. El proyecto está implementado en el lenguaje Java por lo que se han utilizado algunas de las librerías básicas que proporciona dicho lenguaje. El motivo por el cual se escogió Java se debe a que es el lenguaje oficial de Android y dispone de muchas librerías, y aunque en el proyecto no tratemos de crear una aplicación, esta sería la mejor salida (utilidad) para el proyecto y en un principio ese era el objetivo.

Ilustración 26 - Logo java

Por otra parte, se han utilizado varias de las librerías proporcionadas por OpenCV[7]. Esta tecnología puede ser usada desde Java, C++, C#, etc. Por lo que se pudieron usar directamente las librerías nativas que proporciona OpenCV, aunque si el proyecto se hubiera escrito en C++, la dificultad de uso hubiera sido más baja, pues la documentación y OpenCV, está más dirigida hacia dicho lenguaje.

Una herramienta muy importante que se ha usado en el proyecto ha sido el software de Weka[8], el cual contiene una colección muy amplia de herramientas de visualización, modelos predictivos, clasificación, regresión, clustering, reglas de asociación y visualización.

Ilustración 27 - Logo WEKA

Finalmente, se ha utilizado el Photoshop[9] como editor de imágenes para poder modificar a nuestras necesidades las distintas imágenes que se han usado a la hora de crear el conjunto de datos.

3.2 Estructura del Proyecto

En este apartado se tratara de explicar de forma breve cual es la estructura y cómo funciona el proyecto, el cual se divide en seis pasos distintos.

Primero se ha realizado un conjunto de datos con las imágenes de varios árboles (árboles u hojas) distintos. Después, se ha hecho un pretratamiento de las imágenes para que se adapten a lo que necesita el programa. A continuación, mediante el software creado, se trata a las imágenes nuevamente y se extraen las características con las que se reconocerán más adelante los distintos tipos de árboles. Al realizar el anterior paso, se obtiene un conjunto de datos con los vectores característica de las distintas instancias de cada árbol. Seguidamente se experimenta cual es el modelo de aprendizaje que mejor

[7] OpenCV es una biblioteca libre de visión artificial originalmente desarrollada por Intel. Disponible en http://opencv.org/
[8] Weka es una plataforma de software libre para el aprendizaje automático y la minería de datos. Disponible en http://www.cs.waikato.ac.nz/ml/weka/
[9] Photoshop es un editor de gráficos rasterizados desarrollado por Adobe Systems principalmente usado para el retoque de fotografías y gráficos

resultados dará y se entrena para así obtener el modelo final (Red de neuronas, clasificación, clusterización, etc.).

Una vez realizados los anteriores pasos, cuando se tenga una nueva imagen para identificar, se le realizaran los mismos pasos que se le realizaron al conjunto de datos (aunque esta vez no hay que escoger ni entrenar un modelo de aprendizaje) para así obtener su vector característica y comprobar con el modelo que se había escogido finalmente a que clase (árbol) pertenece la nueva imagen.

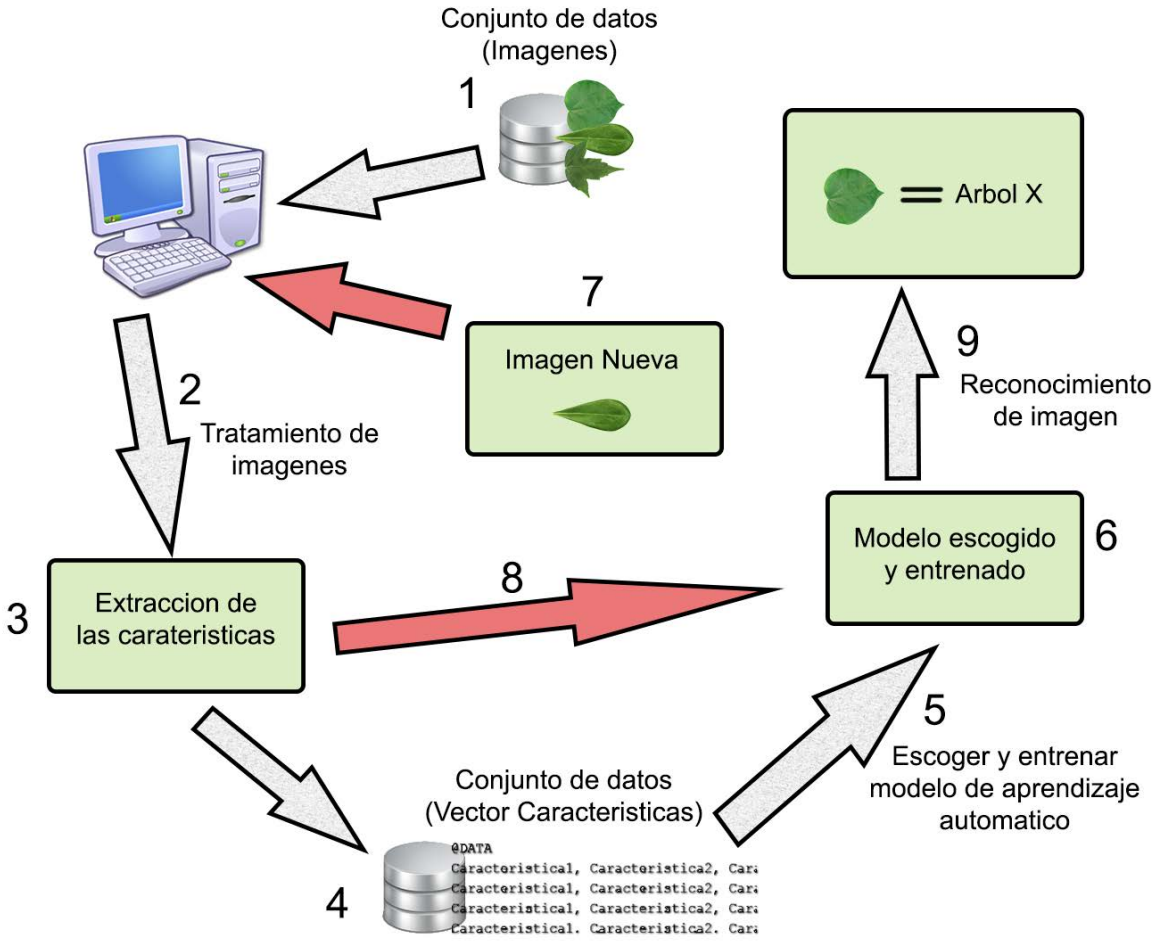


Ilustración 28 - Estructura del proyecto

3.3 Tratamiento de imagen

En cualquier proceso de visión artificial es muy importante el tratamiento de imágenes, sin realizarlo, una imagen que puede proporcionar mucha información útil, posiblemente todo lo que obtendría de ella sería nada más que ruido o información inútil. Una misma imagen con un nivel de brillo u otro, o un tamaño u otro, puede producir una mala extracción de las características de las imágenes, es por eso por lo que primero se tienen que tratar con diferentes procesos y operaciones.

Por otra parte, a la hora de capturar las imágenes hay que poner unas normas para facilitar el tratamiento de las imágenes posteriormente.

Las fotos tienen que ser del mismo tamaño (que no el contenido), en nuestro caso 1600*1600 píxeles. En el caso de los árboles, no deberán tener otros árboles detrás que nos puedan generar ruido y la foto deberá estar tomad en vertical. Para el caso de las hojas, al ser una pequeña parte extraíble del árbol, la foto se deberá hacer sobre un fondo blanco (una hoja de papel mismamente) y el peciolo (tallo) hacia la izquierda aunque se quitara para la foto.

Ilustración 30 - Ejemplo captura de árbol

Ilustración 29 - Ejemplo captura de hoja

3.3.1 Preprocesado de las imágenes

Este procesador se realiza con la ayuda de un editor de imágenes, y lo que pretende es preparar la imagen para que al obtener las características de la hoja, se haga con el menor ruido posible.

En el caso de los árboles, hay que cerciorarse de que el tamaño de la imagen sea el deseado y que en caso de que existe algún resto de otro árbol en la imagen se eliminara.

En el caso de las hojas, se limpiara el fondo, pues aun haciendo las fotos en un fondo blanco pueden aparecer manchas o irregularidades. También se colocara la hoja lo más horizontal posible y dirigiendo la hoja hacia la derecha.

En caso de que alguna de las imágenes tenga defectos como pueden ser una hoja deforme o un árbol que ha sido podado y no se ajustan a las características más típicas de la clase, dicha instancia será eliminada del conjunto de datos.

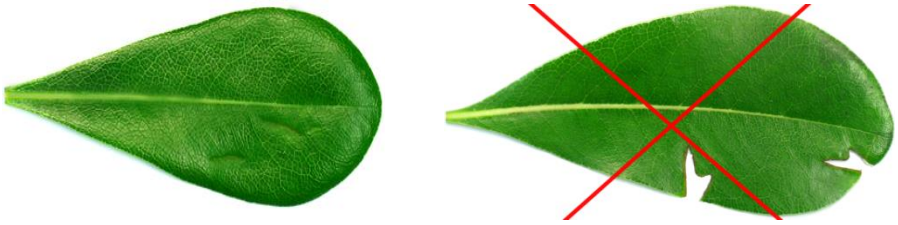


Ilustración 31 - Ejemplo hojas validad & no validas

3.3.2 Escala de grises

Anteriormente se había comentado la posibilidad de incluir el color como una de las características, pero también se dijeron los problemas que traía el hecho de que a lo largo del año un árbol u hoja pudiera cambia su tonalidad radicalmente, por lo que al final se trabajara con las imágenes en una escala de grises pues de esta forma tendremos la imagen en un formato más conveniente para poder extraer las características.

Con el estudio morfológico que se ha realizado anteriormente se llegó a la conclusión de que lo que más información nos podía dar tanto en el caso de los árboles como en el de las hojas era el contorno de dicha imagen. Para poder hallar dicho contorno de manera más sencilla lo mejor es convertir la imagen a escala de grises pues el color no es necesario para esta tarea.

De inicio la imagen se encuentra en un modelo de tipo RGB (red, green, blue), en donde cada píxel de la imagen está formado por estos tres atributos distintos donde cada uno puede tener un valor comprendido entre los rangos 0 y 255, por lo que un mismo píxel puede llegar a tener 255^3 (16.581.375) colores diferentes, en cambio, en un formato de escala de grises cada píxel está formado por un único atributo con un valor comprendido entre 0 y 255 el cual significará cual es la tonalidad de gris (entre blanco y negro incluidos) que tomara dicho píxel.

La conversión de una imagen RGB a escala de grises se puede hacer mediante varias formas, de las cuales vamos a explicar algunas y la razón de haber escogido una en concreto [25].

La primera de ellas es mediante un promedio, sumar el valor del rojo, verde y azul y dividirlo por tres, es el método más rápido, pero la imagen queda sucia, dado que el ser humano percibe la luminosidad (brillo) de manera distinta.

$$Gris = (rojo + verde + azul) / 3$$

El segundo método tiene en cuenta la corrección para el ojo humano, pues este segundo algoritmo juega con el hecho de que la densidad de cono en el ojo humano no es uniforme en todos los colores. Los seres humanos perciben el verde con más fuerza que el rojo, y el rojo más fuerte que azul. Esto tiene sentido desde el punto de vista de la biología evolutiva, pues la mayor parte del mundo natural aparece en tonos de verde, por lo que los seres humanos han evolucionado una mayor sensibilidad a la luz verde. Este método, lo que realiza es ponderar con unos valores diferentes al rojo, verde y azul.

$$Gris = (Rojo * 0{,}3 + Verde * 0{,}59 + Azul * 0{,}11)$$

Otro método, es el de la desaturación, el cual suele dar resultados más planos o más oscuros (con menos tonalidades de grises). Esto se debe a que toma el valor máximo y mínimo entre el rojo, ver y azul de un punto y hace la media de ambos.

$$Gris = (Max\,(Rojo, Verde, Azul) + Mín\,(Rojo, Verde, Azul)) / 2$$

El cuarto método posible es el de la descomposición, el cual se basa en escoger el mayor valor (descomposición máxima) o el menor (descomposición mínima) entre los valores rojo, verde y azul. Esto produce que podamos tener dos tipos diferentes de escala

de grises, en un caso, la imagen será muy oscura (descomposición máxima) y en el otro muy clara.

Descomposición máxima:

$$Gris = Max\,(Rojo, Verde, Azul)$$

Descomposición mínima:

$$Gris = Min\,(Rojo, Verde, Azul)$$

Existen más métodos distintos pero entre los explicados se ha escogido el método que mejor se adapta a la visión humana que es el que tiene en cuenta la corrección para el ojo humano y además OpenCV tiene implementado dicho método.

$$Gris = (Rojo * 0{,}3 + Verde * 0{,}59 + Azul * 0{,}11)$$

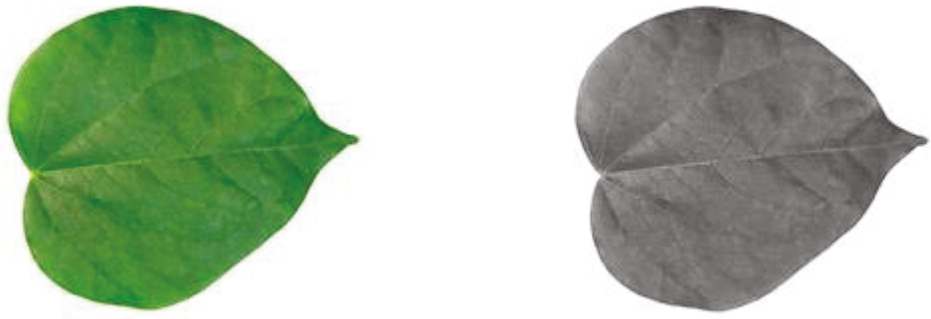


Ilustración 32 - Filtro escala de grises

3.3.3 Suavizado de la imagen

El objetivo fundamental de este paso es obtener, a partir de la imagen en escala de grises, otra final cuyo resultado sea más adecuado para poder distinguir el contorno y de esta manera eliminar el ruido de la imagen. En el caso de los árboles, se buscara suavizar las hojas o huecos en blancos que pueda tener en su interior y en el caso de las hojas eliminar la venación interior y reducir los diferentes tipos de márgenes que pueden tener cada hoja.

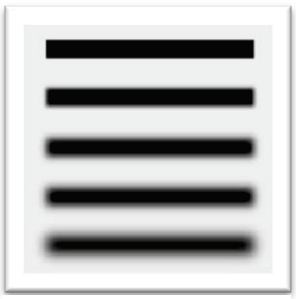


Ilustración 33 - Ejemplo suavizado

Para ello usamos el desenfoque que provoca la mezcla de colores de los píxeles vecinos (dependiendo de un radio de acción) para así eliminar detalles que anteriormente lo único que generaban era ruido.

Para realizar el desenfoque gaussiano existe una fórmula

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Donde x e y determinan las posiciones del punto en la matriz (o espacio), σ es la desviación estándar de la distribución gaussiana

En el proyecto se ha usado el método ya implementado de OpenCV "GaussianBlur", el cual tiene dos parámetros para ajustar la fuerza del desenfoque.

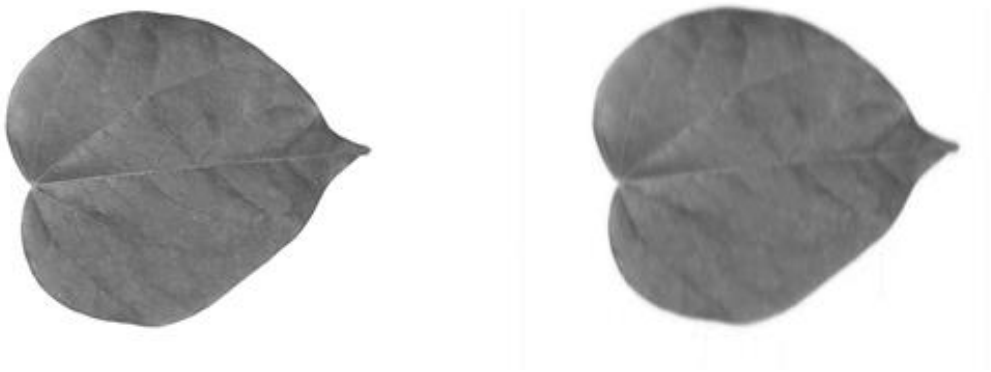


Ilustración 34 - Filtro suavizado de la imagen

3.3.4 Binarización

El objetivo de este pasó transformar la imagen que habíamos obtenido en escala de grises y suavizada con el desenfoque gaussiano a una imagen de solo dos colores (blanco y negro generalmente). Con esto lo que se pretende es eliminar más todo el ruido posible de la imagen para obtener como resultado final el de la silueta de la hoja o árbol. Para poder realizar la Binarización, es obligatorio que la imagen este en escala de grises siendo este uno de los motivos por los que anteriormente habíamos transformado la imagen a escala de grises.

Para realizar esta tarea, se usó el método de OpenCV "threshold" [26] el cual tiene varios parámetros a modificar. Primero hay que escoger un valor umbral (entre 0 y 255). El otro parámetro que se puede modificar en el método de OpenCV es el tipo de operación de umbral que se quiere usar, habiendo hasta 5 tipos diferentes, provocando que a veces la Binarización no dé como resultado una imagen con únicamente dos colores.

A continuación se ilustrara cómo funcionan estos procesos de fijación de umbrales. La ilustración 35 representa una imagen con valores de intensidad en sus píxeles [lo que se llamara src(x, y)] y la línea azul representa el umbral.

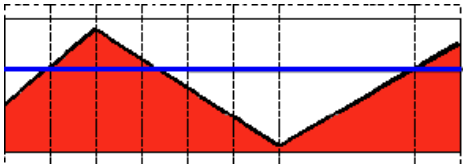


Ilustración 35 - Imagen de ejemplo de umbrales

El primero es el Threshold binary (umbral binario).

$$dst(x,y) = \begin{cases} maxVal & si\ src(x,y) > umbral \\ 0 & cualquier\ otro\ caso \end{cases}$$

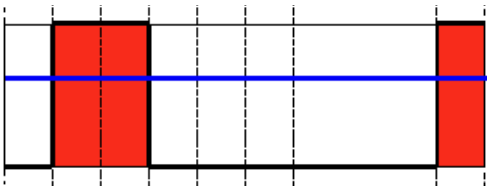


Ilustración 36 - Umbral binario

Como se puede observar, aquellos puntos en los que la intensidad sobrepasaba el umbral han tomado como valor 255 (el máximo) y los que no lo sobrepasaban el mínimo (0).

El segundo tipo es el Threshold binary inverso, que funciona de la misma manera, solo que en este caso, cuando se sobrepasa el valor es el mínimo y cuando la intensidad no es suficiente el valor es el máximo.

$$dst(x,y) = \begin{cases} 0 & si\ src(x,y) > umbral \\ maxVal & cualquier\ otro\ caso \end{cases}$$

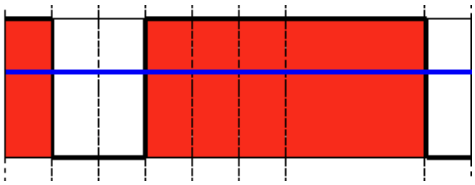


Ilustración 37 - Umbral binario inverso

La diferencia entre estos dos tipos, es poca, pero cuando una imagen es muy clara, lo mejor es usar el normal, pero si la imagen es oscura, da mejores resultados el inverso.

El siguiente tipo que se puede usar es el truncate (truncado), que lo que realiza es que cuando la intensidad sobrepasa el umbral la baja hasta el valor de este.

$$dst(x,y) = \begin{cases} umbral & si\ src(x,y) > umbral \\ src(x,y) & cualquier\ otro\ caso \end{cases}$$

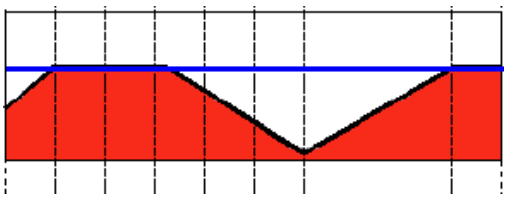


Ilustración 38 - Umbral truncado

El threshold to zero (umbral a cero) deja los valores de intensidad que superen el umbral tal y como están, pero aquellos que están por debajo los convierte al valor cero.

$$dst(x,y) = \begin{cases} src(x,y) & si\ src(x,y) > umbral \\ 0 & cualquier\ otro\ caso \end{cases}$$

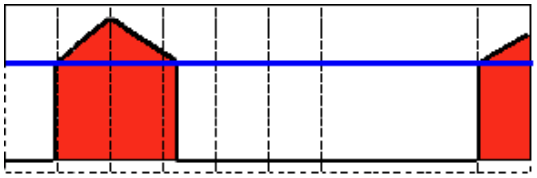


Ilustración 39 - Umbral a cero

El último tipo de procesamiento de umbral es el Threshold to zero invertido, en este caso cuando la intensidad supera el umbral convierte el valor a cero.

$$dst(x,y) = \begin{cases} 0 & si\ src(x,y) > umbral \\ src(x,y) & cualquier\ otro\ caso \end{cases}$$

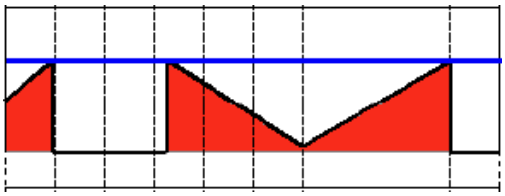


Ilustración 40 - Umbral a cero invertido

El que mejor se adapta al proyecto es el umbral binario, pues generalmente las imágenes que se recibirán son de tonalidades claras y en las pruebas es la que mejor resultados ha dado.

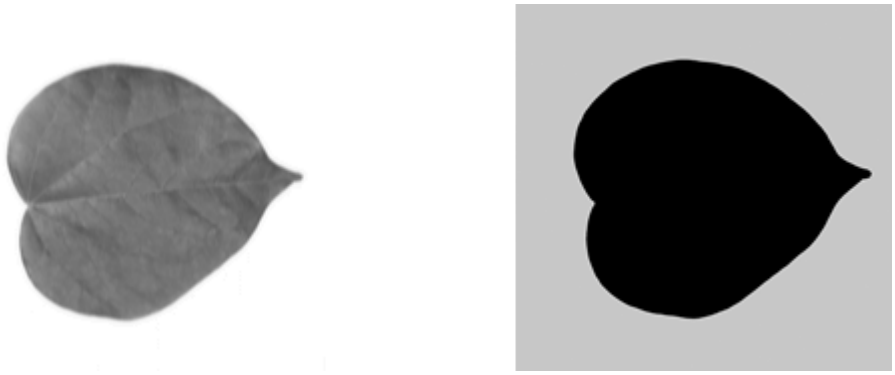


Ilustración 41- Filtro Binarización

3.3.5 Detección de bordes

Este último filtro se realiza con el objetivo de convertir la imagen binarizada en una imagen en la que solo aparezcan los bordes[10] en un color (blanco o negro generalmente) diferente del resto, de esta manera, se podrá extraer más fácilmente las características con las que más adelante se identificara el árbol u hoja.

Para que este paso tenga los mejores resultados, la imagen tiene que haber sido limpiada del máximo ruido posible, pues este paso no quitara ruido, sino que marcara más fuerte los bordes.

Existen varios algoritmos para detectar bordes en imágenes generalmente basados en el gradiente, los más conocidos se realizan con el operador de Sobel y el Canny.

[10] Los bordes de una imagen digital se pueden definir como transiciones entre dos regiones de niveles de gris significativamente distintos.

El operador Sobel [27] calcula el gradiente de la intensidad de una imagen en cada punto (píxel). Así, para cada punto, este operador da la magnitud del mayor cambio posible, la dirección de este y el sentido desde oscuro a claro. El resultado muestra cómo de abruptamente o suavemente cambia una imagen en cada punto analizado y, en consecuencia, cuán probable es que este represente un borde en la imagen y, también, la orientación a la que tiende ese borde.

Algoritmo de Canny [28] es un operador que utiliza un algoritmo de múltiples etapas para detectar una amplia gama de bordes en imágenes, basado en tres criterios:

- Uno de los criterios de detección expresa el hecho de evitar la eliminación de bordes importantes y no suministrar falsos bordes.

- Otro criterio de localización establece que la distancia entre la posición real y la localizada del borde se debe minimizar.

- El último criterio es una respuesta que integre las respuestas múltiples correspondientes a un único borde.

Aparte, el algoritmo de Canny se basa en tres grandes pasos:

- Obtención del gradiente: en este paso se calcula la magnitud y orientación del vector gradiente en cada píxel.

- Supresión no máxima: en este paso se logra el adelgazamiento del ancho de los bordes, obtenidos con el gradiente, hasta lograr bordes de un píxel de ancho.

- Histéresis de umbral: en este paso se aplica una función de histéresis basada en dos umbrales; con este proceso se pretende reducir la posibilidad de aparición de contornos falsos.

Entre estos dos tipos de operadores al final se eligió el operador Canny, el cual es más complejo computacionalmente (tarda más), pero da resultados más limpios y continuos, y como el tiempo de cómputo para cada imagen no supera ni el segundo, la complejidad computacional no supone un problema. Por otra parte, OpenCV también traía implementado el método de Canny [29] [30].

A continuación, un ejemplo entre la diferencia del operador Sobel y el Canny

Ilustración 42 - Diferencia operador Sobel y Canny

En la siguiente imagen (x) se puede observar el resultado final de aplicar la detección de bordes.

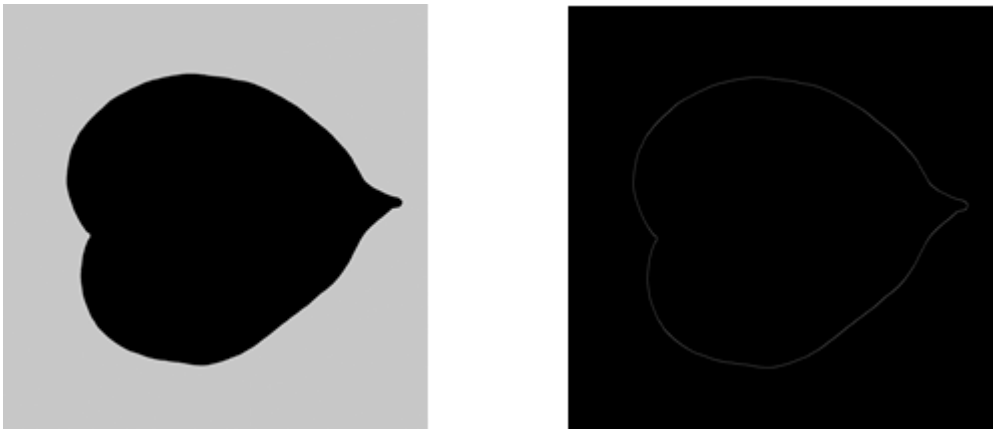


Ilustración 43 - Filtro Canny

Con esta imagen final, es con la que se trabajara a partir de este punto, y de ella se realizara la extracción de características. Como se puede ver, la imagen solo nos mostrara la silueta de la hoja marcada en color blanco sobre un fondo negro. Aunque a la hora de extraer características el proyecto se basara más en la silueta general, estos filtros también distinguen con exactitud los márgenes de la hoja.

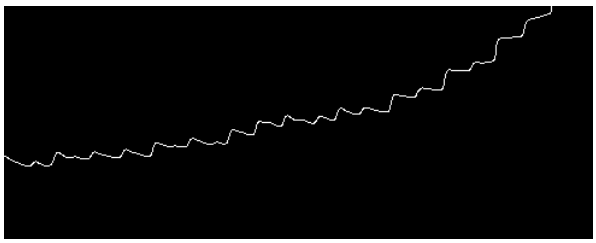


Ilustración 44 - Ejemplo borde final

3.3.6 Fallo en el tratamiento de la imagen del árbol completo

Al llegar a este punto, se observó que había un gran problema con el tratamiento de imagen del árbol completo.

El problema con los arboles es que aun eliminando la mayor parte del ruido, o la imagen del árbol está muy bien definida o está completamente aislada de todo lo demás, pues a la hora de pasar los filtros (transformar la imagen en escala de grises, suavizado Gaussiano, Binarización, y uso del algoritmo Canny) para obtener los contornos en algunos árboles, ciertos parámetros dan resultados que quedan bien pero con otros árboles (y mismos parámetros) quedan mal, a continuación algunos de los "mejores" resultados aunque con los parámetros ajustados como mejor se pudo a cada árbol (de forma individual):

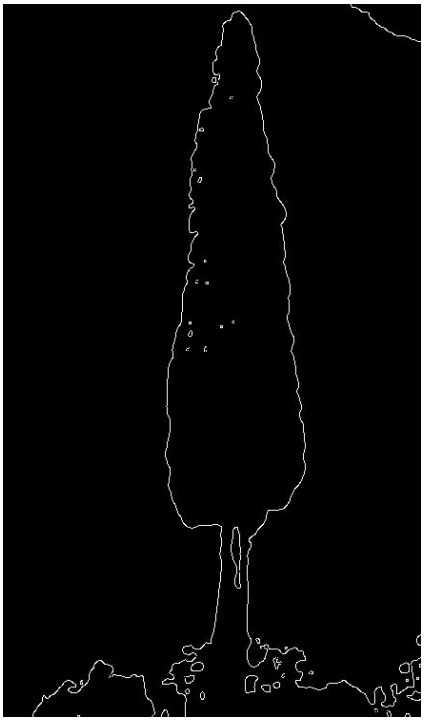


Ilustración 45 - Árbol procesado 1

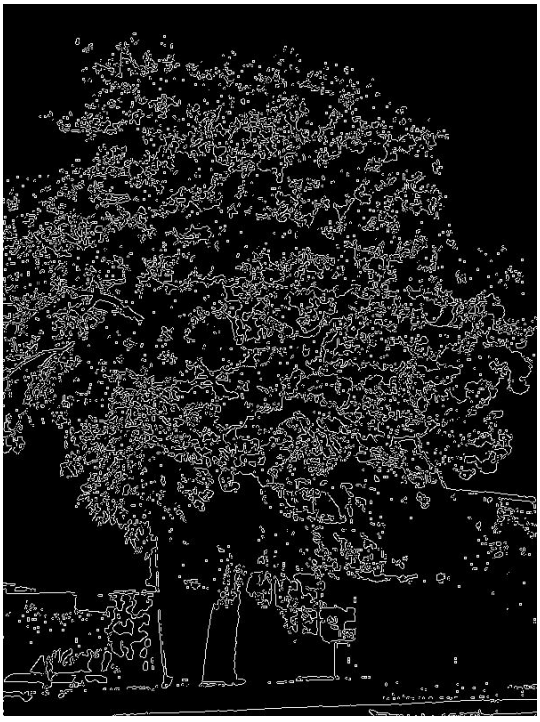


Ilustración 46 - Árbol procesado 2

Como se puede observar, el primer árbol (ilustración 45) si se ha conseguido eliminar prácticamente todo el ruido interno y se distingue perfectamente su silueta exterior, en cambio, en el segundo árbol (ilustración 46) aun ajustando los parámetros específicamente para él, el ruido interior (las hojas) ha sido imposible de eliminar.

En cambio, lo que ocurre con otros árboles con los mismos parámetros impide que más adelante se puedan extraer todas las características que necesitamos para poder distinguirlo.

Las siguientes imágenes (ilustración 46) pertenecen a un ciprés (igual que la ilustración 45), tiene mucho más ruido dentro, y cuando intentamos ajustar los parámetros para que dicho ruido no aparezca, ocurre que la imagen (ilustración 47) empieza a perder los bordes con los que se delimitaban su silueta pero en cambio sigue manteniendo mucho ruido interno.

Ilustración 47 - Ciprés 1

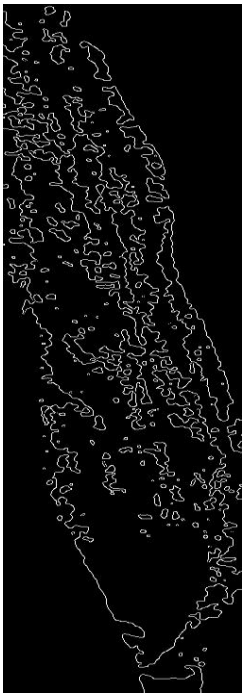


Ilustración 48 Ciprés 2

Para ver si conseguía reducir ruido o captar mejor la forma, he cambiado los distintos pasos a seguir (el orden de estos) que estaba usando (suavizado gaussiano, algoritmo de Canny) pero no había mucha diferencia.

Por lo que se puede suponer que el resultado estará demasiado ligado a la calidad de la foto, del árbol y de la espesura de las hojas de este.

Entonces, ¿Cuál sería el caso perfecto? El caso perfecto seria aquel que con los mismos parámetros para distintos arboles (y no tener que ajustarlo por cada árbol distinto) se consiguiera obtener la silueta del árbol sin ningún tipo de ruido interno o externo, y aunque si se pudiera conseguir la silueta externa de forma general (imágenes) el reconocimiento a través de la foto de un árbol general aun sería posible, pero en ese punto, aun sería muy difícil de conseguir distinguirlo pues habría que enfrentarse aun a la posibilidad de que el árbol estuviera podado o que hubiera crecido de forma extraña, pues la inclinación del suelo, la cantidad de agua, la orientación por donde recibe el sol, y muchos otros factores también afectan a la forma general del árbol.

A continuación se puede observar en las dos imágenes el resultado (dibujado a mano) que se tendría que haber obtenido para seguir con la identificación de árboles a través de una única foto general.

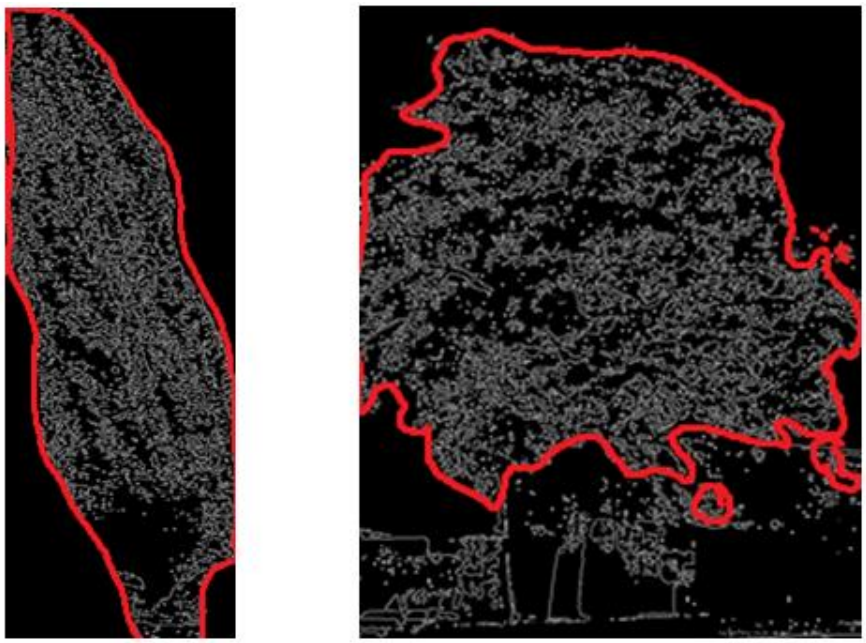


Ilustración 49 - Resultado deseado

Por esta razón, el proyecto se centrara en la identificación de árboles a través de las hojas de un árbol, pues se descarta la posibilidad de conseguirlo con la foto del árbol general. Además, se conseguirá identificar con más exactitud pues la mayor característica por la que se distingue un árbol es por su hoja, más facilidad para obtener un conjunto de datos (de un solo árbol puedes hacerte con una gran cantidad de muestras perfectas).

3.4 Extracción de características

En esta sección se realiza lo más importante de todos, la obtención de las características, para así obtener el vector característica de cada imagen y conseguir clasificar cada una de ellas.

Para que las características nos sirvan, estas tienen que cumplir varios requisitos.

- Tienen que ser características discriminantes entre las distintas clases (arboles)
- Tienen que ser robustas y por tanto ser insensibles al posible ruido que tenga la imagen (pues no se puede asegurar que tras el tratamiento de la imagen, todas estén libres al 100% de ruido).
- Las características obtenidas no pueden depender de un factor aleatorio, pues si se obtuviera una característica que para la misma clase diera resultados poco fiables y pudieran pertenecer a otra clase, al final generaría ruido en el vector característica.
- Tienen que poseer determinadas invariancias respecto a la escala y translación de la imagen, pues una hoja de un mismo árbol puede haber sido fotografiada desde más cerca o más lejos o haberla centrado más o menos en el “cuadro”.

En el caso del proyecto no es necesario que las características sean invariables a la rotación pues en el pretratamiento habíamos puesto todas las imágenes con la misma orientación.

Estas características deberán ser independientes de la escala (tamaño), dado que se puede realizar una foto de una hoja a una distancia u otra.

3.4.1 Momentos invariantes de Hu

Buscando alguna forma de extraer características que fueran invariables a la escala, se encontró la tecnología basada en los momentos invariantes. Los momentos son un promedio ponderado determinado de las intensidades de los píxeles de la imagen. Existen varios tipos (ejemplo: Momentos de Zernike), aunque los más usados son los momentos propuestos por Hu [31] [32], que se explicaran a continuación.

La geometría de una región plana se basa en el tamaño, la posición, la orientación y la forma. Todas estas medidas están relacionadas en los parámetros denominados momentos.

Mediante los momentos invariables de Hu, se consigue una descripción independiente a las posiciones, tamaños y rotación de la imagen. Cuando se desea obtener un descriptor invariante respecto de la posición, se puede utilizar los momentos de Hu centrales los cuales se definen como:

$$\mu\mu_{pq} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}(x - \bar{x})^p(y - \bar{y})^q f(x,y)dx\,dy$$

Donde $\bar{x} = \frac{M_{10}}{M_{00}}$ e $\bar{y} = \frac{M_{01}}{M_{00}}$ son los componentes del centroide

Una vez obtenido los momentos centrales, para obtener los invariantes a la escala, se le dividen a cada momento por el escalado adecuado expuesto en la siguiente formula:

$$\eta_{ij} = \frac{\mu_{ij}}{\mu_{00}^{(1+\frac{i+j}{2})}}$$

De esta forma, se obtendrían varias características (9 aproximadamente) describiendo la imagen sin importar la escala o la posición, pero como las imágenes de las hojas normalmente pueden sufrir pequeñas rotaciones (al no ser "objetos" idénticos dentro de su propia clase y que la captura de la imagen no es siempre perfecta, a veces tienen una rotación de entre uno y cuatro grados) hay que emplear los momentos invariantes los cuales son los siguientes:

$I_1 = \eta_{20} + \eta_{02}$

$I_2 = (\eta_{20} - \eta_{02})^2 + 4\eta^2{}_{11}$

$I_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2$

$I_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2$

$$I_5 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]$$

$$I_6 = (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03})$$

$$I_7 = (3\eta_{21} - 3\eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] - (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]$$

Con estas siete características de la imagen, conseguimos una descripción de la imagen independiente del tamaño, rotación y posición, de esta forma la única parte de la imagen que si expresan estas características es la forma.

Se sabe de otras técnicas parecidas a los momentos invariantes de Hu, como por ejemplo la *Transformada de Fourier-Mellin[37]* desarrollada por Cohen, que a partir de otro método obtiene también varios valores para reconocer patrones en imágenes de forma invariante a la escala, rotación y posición. Más adelante se hablara en trabajos futuros.

Aunque las características obtenidas con los momentos invariantes de Hu podrían dar buenos resultados por si solos, se decidió conseguir más características para ajustar mejor los resultados, las siguientes características deberían de ser también invariantes a ciertos aspectos como la escala, posición y Angulo de rotación. En cuanto a la posición y ángulo, es bastante fácil de salvar, pero la escala, al no tener ningún objeto de referencia en las imágenes limitara mucho las posibilidades.

Para obtener características nuevas, se analizara la imagen final que se había obtenido anteriormente, la cual es una matriz de 1600*1600 que contiene 0 para los espacios en blanco y 255 para los marcados.

3.4.2 Proporción Área & Área

Una de las características que se ha obtenido es la de área, la cual expresara cuantos píxeles hay dentro del contorno de la hoja tal y como se ve en la siguiente imagen.

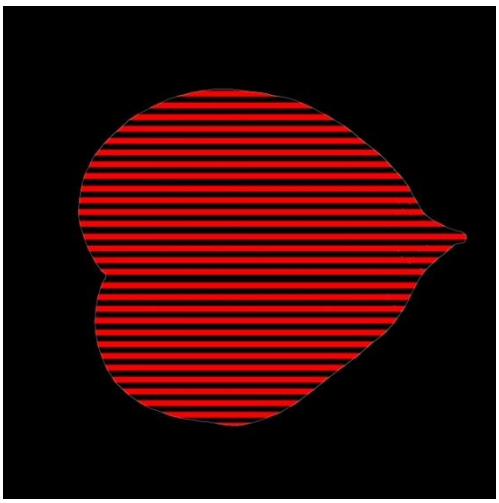


Ilustración 50 - Ejemplo área

Para ello lo que se hace es analizar la imagen por filas, pues realmente se obtiene el área de cada fila que más adelante se suma al área total. Por cada fila, se analizan los cambios de color, se guardan las coordenadas en parejas de dos y tras finalizar la fila, se calcula la distancia entre dichas parejas para agregar su resultado a una variable de área total. La parte difícil de esta solución, son las tangentes o el posible ruido, pues provoca que pueda haber parejas incompletas (que el número de veces que la hoja corta dicha fila sea impares) creando algunos casos especiales:

- Una única pareja incompleta, si esto sucede, lo más seguro que es porque existe un píxel activado antes de llegar a la propia hoja y por lo tanto es ruido o porque es la tangente de inicio o final de la hoja. En estos casos no suma área dicha fila (tal y como debería ser).
- Más de una pareja incompleta, si esto sucede, es porque hay un píxel de ruido y en cuyo caso en vez de calcular el área de esa fila se tomara el área de la fila anterior pues la diferencia entre una fila y otra es mínima.

Ilustración 51 - Error de tangentes

En esta imagen se pueden apreciar ambos "errores", la línea de arriba representa el caso de la tangente superior provocando que solo exista un único cambio de color en esa línea. Y en la segunda línea negra se aprecia la tangente de la punta de otra subhoja provocando que en esa fila existan cinco cambios de color, por lo que hay una pareja incompleta.

Aun si no se trataran estos posibles errores, el error seria mínimo, por lo que no afectaría en gran medida al resultado final (pues el error en cinco o diez filas de un total de mil seiscientas apenas supone un problema en esta característica).

El problema que supone esta característica es que no es invariante a la escala, por lo que se decidió dividir por el perímetro del contorno de la hoja.

$$ProporcionArea = \frac{AreaHoja}{PerimetroHoja}$$

Donde PerímetroHoja es la cantidad de píxeles activados de la imagen, dado que estos solo deberían pertenecer al propio contorno de la hoja. De esta forma, está característica siempre será proporcional a la escala.

3.4.3 Proporción Perímetro

Una buena forma de mantener la proporción escalar en la obtención de características es dividir la imagen en varios sectores y ver la relación entre ellos. El

problema de este método, es que no es invariable a la posición, pero esto es muy fácil de evitar, pudiendo hacerse en el preprocesado (nuestro caso), en el tratamiento de la imagen (buscar los puntos más a la derecha, arriba, izquierda y abajo y centrar la hoja en la imagen) o incluso a la hora de dividir la hoja en sectores (de la misma forma que en el tratamiento de la imagen).

La característica que se ha obtenido de esta forma es la proporción del perímetro. Se ha dividido la hoja en dos secciones, derecha e izquierda, dado que generalmente las hojas presentan una gran diferencia entre la zona más cercana al peciolo (tallo) y la más alejada. Normalmente, las hojas son simétricas si se divide por la nervadura o nervio central, por lo que realizar el mismo proceso pero divido de esta forma solo nos serviría para poder encontrar hojas no simétricas.

Una vez se tiene separada la imagen, se calcula el perímetro de la zona derecha y el perímetro de la zona izquierda y se divide para obtener la característica ProporciónPerímetro.

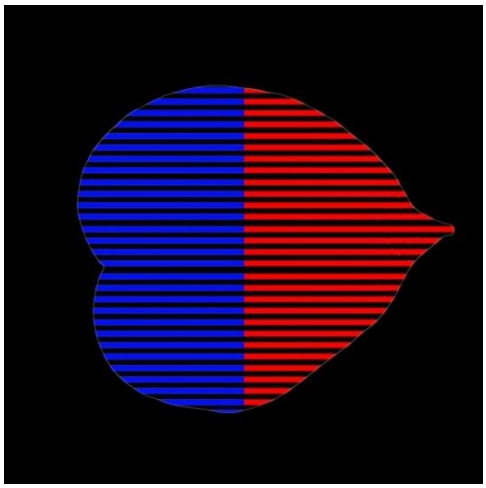


Ilustración 52 - Proporción perímetro

3.4.4 Proporción Altura/Anchura & Proporción Altura/Anchura Media

En este caso, se quiere encontrar una proporción entre las medidas de la propia hoja, la mejor y más fácil que se puede extraer es la proporción entre la altura y la anchura de una hoja, de esta forma, está característica seria invariante a la escala, y posición.

$$Proporcion = \frac{Altura\ de\ la\ hoja}{Anchura\ de\ la\ hoja}$$

Pero para realizar esto se tienen dos posibilidades, para obtener la anchura se puede tomar el punto activado más a la derecha y a la izquierda de la imagen y calcular la distancia entre ambos en el plano horizontal o se puede buscar el punto medio de la hoja y calcular la anchura en dicha posición. Y lo mismo para la altura solo que se calcularía la distancia entre el punto más alto y el más bajo en el plano vertical.

Al final se decidió usar ambos métodos, calcular una proporción entre la altura y la anchura total (proporción normal) y calcular la misma proporción pero en el punto medio de la altura de hoja (allí donde suele encontrarse la nerviación, llamada proporción media).

De la misma forma que en la característica anterior se obvió por dividir la hoja en mitad superior e inferior, en este caso no se calcula la proporción en la anchura media de la hoja pues generalmente se obtendrían los mismos resultados que con la proporción normal. La razón de esto, es porque muchas de las hojas crecen primero hacia atrás para más tarde crecer hacia la punta de la hoja (Ilustración 53), por lo que con estas dos características, aparte de saber cómo de circular es la hoja, se sabrá si la hoja crece de esa forma o no, pues de ser así, la proporción normal tendrá un valor menor que la proporción media.

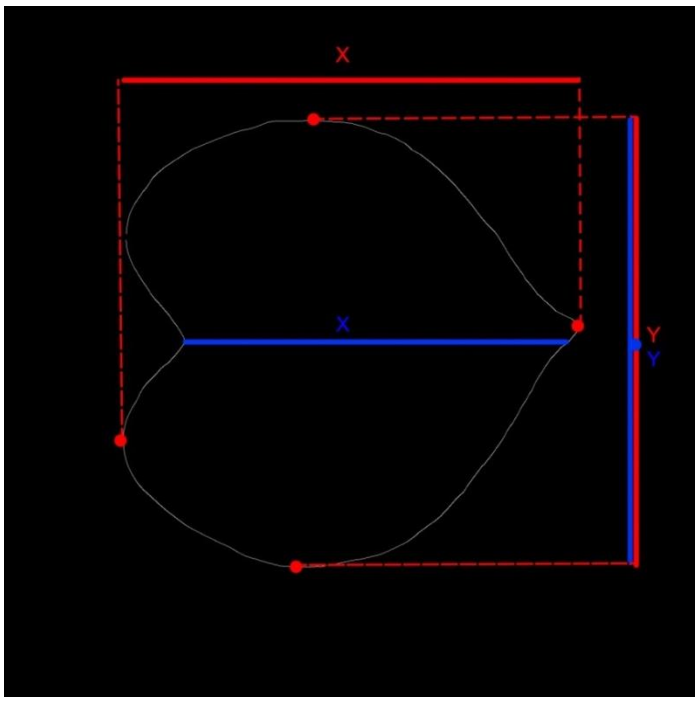


Ilustración 53 - Proporciones altura anchura

La proporción normal dibujada en rojo y la proporción media dibujada en azul. Se puede observar en la imagen que la diferencia es notoria, y que en la proporción normal es necesario calcular la distancia en el plano horizontal y vertical pues los putos no tienen por qué pertenecer a la misma fila de la imagen.

3.5 Dataset

Cualquier proyecto basado en el aprendizaje automático necesitara un conjunto de datos[11] (o banco de datos) para probar el correcto funcionamiento de este. En el caso de este proyecto, el banco de datos estará formado por las imágenes de distintas hojas (clases). Dado que al principio del proyecto existe una preprocesamiento de las imágenes, no es importante que las condiciones de las imágenes no sean exactamente las mismas a las requeridas, aunque cuanto más diste de lo necesitado, más trabajo llevara adaptar las imágenes al proyecto.

Se puede recurrir a dos formas para obtener un banco de datos; La primera es usar un conjunto ya existente, ahorrando así mucho tiempo en la elaboración de los datos, y

[11] Una base de datos o banco de datos es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso.

además ya ha sido usada en otras investigaciones y por tanto es un conjunto de datos "validado". Por otra parte, se puede escoger por realizar una agrupación de datos personalizada, desde cero, pudiendo adaptarla y ajustarla a las necesidades del proyecto, de esta forma, posiblemente se ahorraría tiempo en la fase de preprocesado de la imagen.

Conjuntos de datos ya existentes

En este apartado se expondrán algunos de los conjuntos de datos que se han encontrado, también se analizaran su utilidad.

100 leaves plant species: Traducido como cien especies de hojas de plantas, es un conjunto de datos creada por *James Cope*, *Thibaut Beghin*, *Paolo Remagnino*, *Sarah Barman*, en las que las hojas se han recogido en el Real Jardín Botánico de Kew , Reino Unido y bajo la supervisión de la Kingston University London [33].

Este banco de datos consiste en cien especies distintas de hojas en las que por cada hoja se recogen dieciséis muestras (instancias) diferentes. Este banco de datos no incluye color, pues las muestras están binarizadas. A continuación, una pequeña muestra del conjunto de datos.

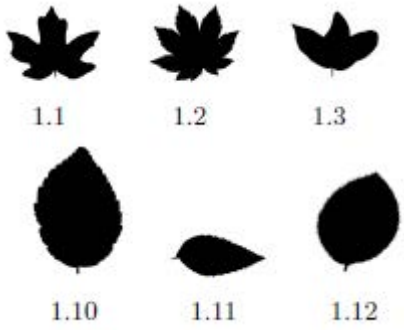


Ilustración 54 - Muestra de datos, 100 leaves plant species

Este conjunto, es muy amplio y ya tiene realizado prácticamente el 90% del procesado que se realiza en este proyecto, por lo que no se podría comprobar el buen funcionamiento de la primera fase del proyecto, pero aparte tiene dos problemas más, principalmente el tamaño de la muestra (16 instancias) es insuficiente para lo que se está buscando, además las imágenes son bastante pequeñas, aunque esto se podría suplir en caso de necesidad.

Leafsnap Dataset Este banco de datos, está creado por la colaboración de tres universidades distintas de Estados Unidos (Columbia University, University of Maryland y Smithsonia Institution) que están realizando un proyecto similar a este.

Este conjunto de datos está dividido en dos tipos, las imágenes capturadas en el laboratorio, con buena calidad, luz, las hojas prensadas y con una escala en el lateral, y por otra parte las capturadas en el campo que consisten en imágenes tomadas en por móviles y que contienen mucho ruido, desenfoque y distintos patrones de iluminación y sombra.

Este banco de datos, contiene más de 185 especies diferentes de hojas del continente americano y más de 31000 muestras distintas siendo un conjunto de datos realmente amplio. Aparte de las hojas en su estado natural, también vienen segmentadas

(realizado el proceso de Binarización y obtención del contorno) y la mayoría de estas tienen mucho ruido, como puede ser agujeros en la hoja, otras directamente no aparecen (dada la iluminación) o aparecen por la mitad, etc. Esto es porque a pesar de haberlas realizado en laboratorio, la calidad de las imágenes no es la necesaria. Por otra parte, el tamaño de las imágenes no es constante y es bastante pequeño (300*400 píxeles). A pesar de estos problemas, con una buena selección y tratamiento se podría usar este conjunto. A continuación, un pequeño ejemplo de muestras que presentan en su página web (siendo claramente de las mejores capturadas)

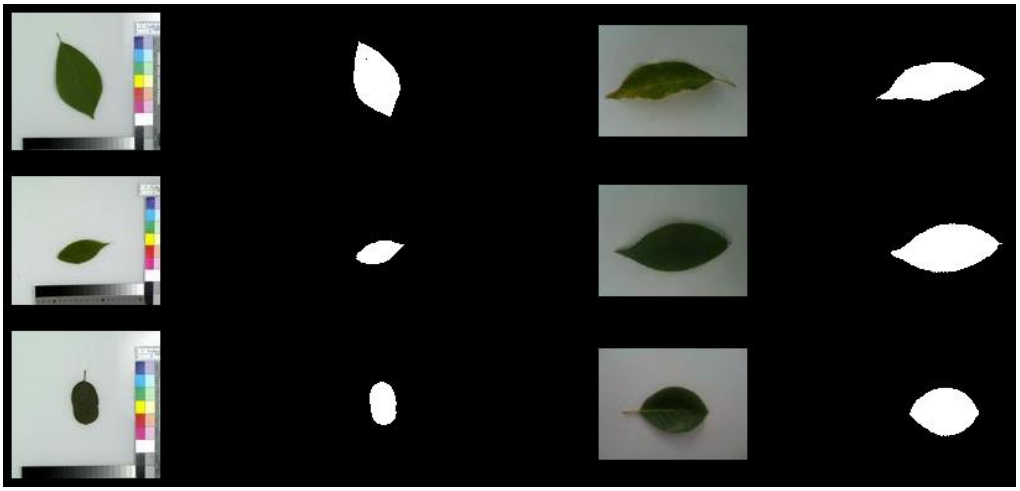


Ilustración 55 - Muestras de datos, Leafsnap

Leaf Recognition: Aunque no se pudo acceder a este banco de datos (no se podía descargar), merece la pena mencionarlo. Es un conjunto de datos sueco usado para el reconocimiento de la hoja, con una cantidad de quince especies diferentes de hojas, además tratada anteriormente para que todas estuvieran alineadas en la misma posición y solo se ha fotografiado el mismo lado de la hoja. Por otra parte, dicho conjunto de datos ya había sido usada en una investigación de reconocimiento de hojas con resultados positivos.

Ilustración 56 - Muestras de datos, Leaf Recognition

Flavia Este banco de datos está creado por *Stephen Gang Wu, Forrest Sheng Bao, Eric You Xu, Yu-Xuan Wang, Yi-Fan Chang* and *Chiao-Liang Shiang*. Está compuesto por treinta y tres hojas distintas con aproximadamente sesenta muestras por hoja, en una calidad difícil de mejorar y sobre un fondo blanco (tal y como se necesita en el proyecto). También se ha usado este conjunto de datos para poder reconocer el árbol al que pertenecen teniendo también resultados positivos, lo que da aún más seguridad a este banco de datos.

Banco de Datos Desarrollado

Además de obtener un conjunto de datos externo, se puede generar uno propio, el punto bueno de hacer el banco de datos de esta forma, es que se elige como realizar la foto pudiendo ajustarla más a las necesidades del proyecto, en cambio, puede llegar a ser difícil encontrar todos los casos necesarios que se necesitan. Por ejemplo al principio cuando aún se estudiaba la posibilidad de las imágenes de los arboles completos, encontrar diez o quince árboles que se ajustaran a las necesidades era relativamente fácil, pero si se hubieran necesitado al menos cien muestras de cada árbol, la tarea hubiera resultado muy difícil y se habrían reducido las muestras necesarias para cada clase de árbol. En el caso de las hojas, eso no pasa, pues de un mismo árbol se pueden obtener muchas hojas, aunque estas tienen que estar al alcance.

Por otra parte, hay varios métodos de conseguir las muestras; Con fotografías a través de un móvil, a través de una cámara profesional, grabando la muestra y recortando del vídeo las muestras necesarias, o también se pueden escanear.

A la hora de fotografiar o grabar, el problema puede ser la mala calidad con las que aparecen las fotos, además de que el brillo puede variar entre las muestras. En cambio, si se usa un escáner, el brillo es constante, la posición de la hoja puede ser siempre la misma, igual que la escala, además es muy simple la tarea de escanear las hojas.

Por otra parte, en este proyecto, se ha decidido escoger el conjunto de datos de *Flavia* al cual se le ha agregado alguna clase capturada a partir del escáner, de esta forma, si se obtienen resultados positivos se demuestra que no solo sirve para un único banco de datos, sino para todas las clases que se adapten a los requisitos del proyecto.

El conjunto de datos que se ha generado contiene nueve clases distintas con una media de cien muestras por cada clase, de esta forma, al tener tantas muestras por cada clase, se evitara que el proyecto sé sobrespecialice en los datos.

Entre las muestras se han introducido clases que no se parecen en nada entre sí y varias que si tienen una gran similitud entre ellas, y así poder comprobar el buen funcionamiento del proyecto.

A continuación, algunas de las muestras que contiene el conjunto de datos.

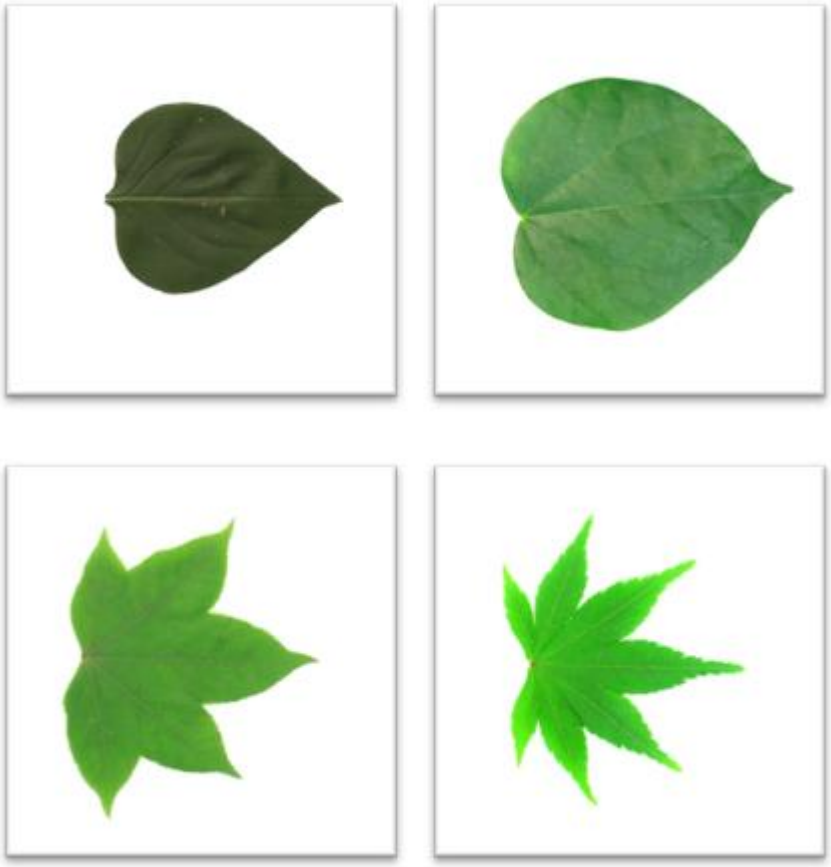


Ilustración 57 - Muestras de las hojas usadas en el conjunto de datos

En la mayoría de las clases, se tuvo que recurrir a transformar geométricamente algunas imágenes para generar nuevos casos pues en alguna de estas no alcanzaban las muestras para llegar a las cien. Las transformaciones geométricas fueron como por ejemplo, alargar, achatar o estirar, algunas de las hojas, generando así casos muy similares pero con distintas características finales.

4 Experimentación y Resultados

En esta sección se realizaran los experimentos llevados a cabo con los vectores característica obtenidos anteriormente, para ello se usaran las técnicas de clasificación y clusterización explicadas en la sección 2.2 de este proyecto.

Pero antes de centrarse en usar dichos algoritmos, se estudiaran y analizaran los diferentes valores de los distintos atributos del vector característica.

El vector característica final está formado por trece características:

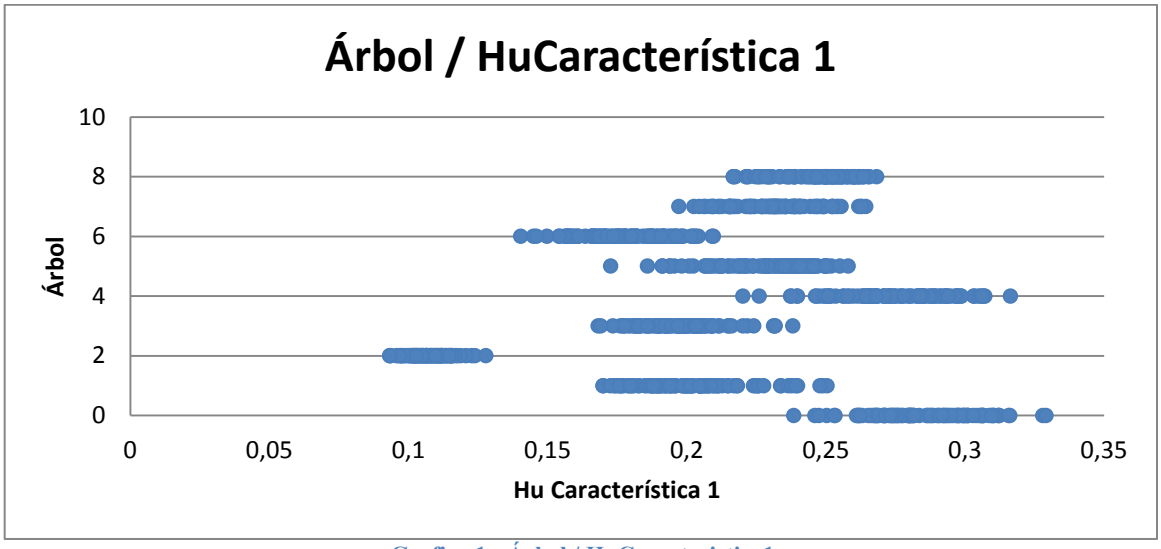
- **HuCaracteristica**: Se componen de siete características y corresponden a los momentos invariables de Hu. Los valores que toman son continuos.
- **Área**: Su valor es continuo y esta entre los siguientes valores [17614 U 1019065] aunque con otras muestras podría aumentar el máximo o disminuir el mínimo, nunca por encima de 1600*1600 ni por debajo de 0.
- **Proporciones**: Las cuatro siguientes características corresponden a la ProporcionArea, ProporcionPerimetro, Proporcion y ProporcionMedia. Los valores que puede tomar son discretos.
- **Árbol**: Esta característica representa el árbol al que corresponde. Al realizar la experimentación con nueve clases distintas los valores que puede tomar el conjunto son los siguientes {0, 1 , 2 , 3, 4, 5 ,6 ,7 ,8} y aunque se les nombra con números es la única clase nominal.

No.	1: HuCaracteristica1 Numeric	2: HuCaracteristica2 Numeric	3: HuCaracteristica3 Numeric	4: HuCaracteristica4 Numeric	5: HuCaracteristica5 Numeric	6: HuCaracteristica6 Numeric
1	0.3056502444913611	0.00122723220286...	0.00106025965789...	7.39954750343281...	-1.6539764371329...	2.38086250180939...
2	0.2937621604202723	4.24514861795675...	5.50638329260112...	3.34014699877724...	-4.4715943149963...	6.14817124468173...
3	0.2770952390090666	2.14099796262952...	2.81110108155389...	1.66335526095515...	4.04601146318725...	2.14237422360426...
4	0.2995419326584483	0.00122825613661...	4.89087910593518...	9.83426420135916...	-2.1526007842603...	3.42962714562003...

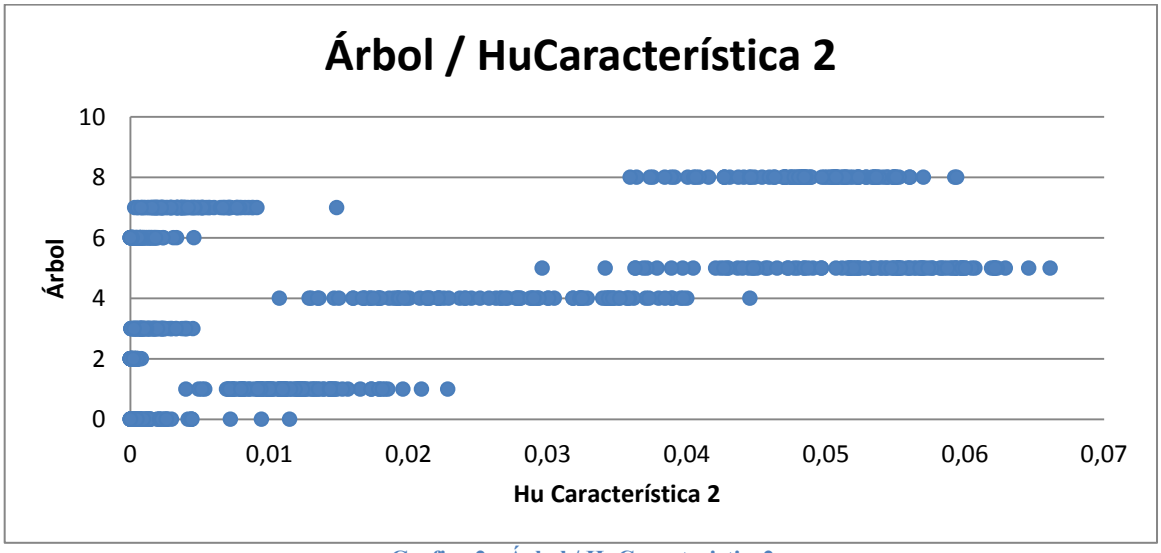
7: HuCaracteristica7 Numeric	8: Area Numeric	9: ProporcionArea Numeric	10: proporcionPerimetro Numeric	11: Proporcion Numeric	12: ProporcionMedia Numeric	13: **Arbol** Nominal
1.24899977327471...	944998.0	240.0	0.9157741	0.86883944	0.9689716	0
7.23998942808386...	924923.0	233.0	0.86723167	0.89689034	1.0579151	0
-1.0630107025403...	862310.0	223.0	1.0741138	0.9882033	1.1684549	0
1.34234531904814...	910633.0	236.0	0.860735	0.8736581	0.9081545	0

Tabla 2 - Vector característica

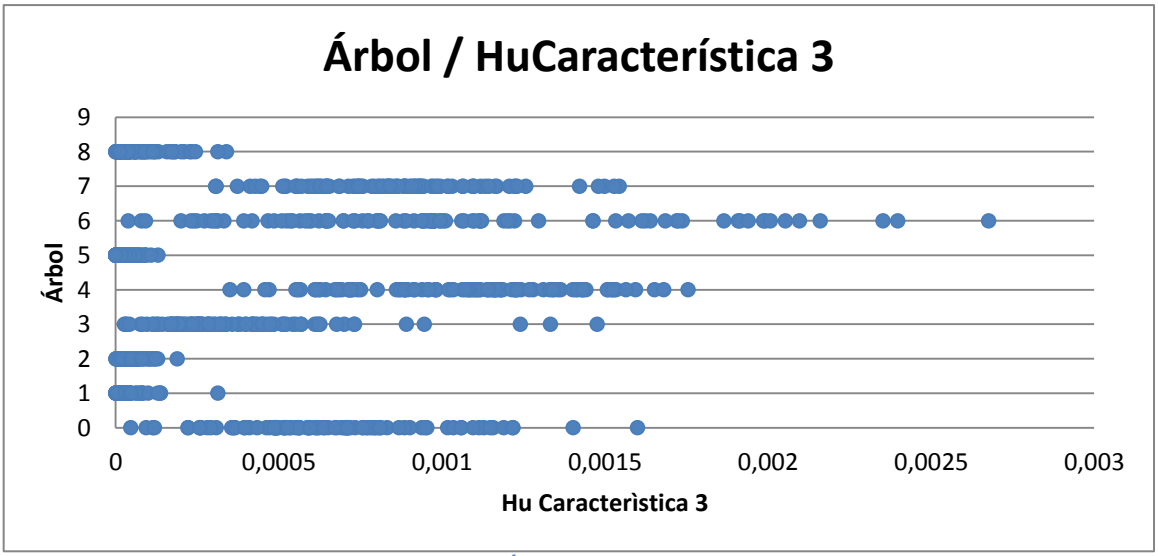
A continuación se mostraran unas gráficas (realizadas con Excel y contrastadas con Weka) en las que se vera la dispersión de los datos respecto al árbol, de esta forma, se podrá determinar si los atributos pueden ser o no discriminantes incluso antes de empezar a realizar la experimentación.



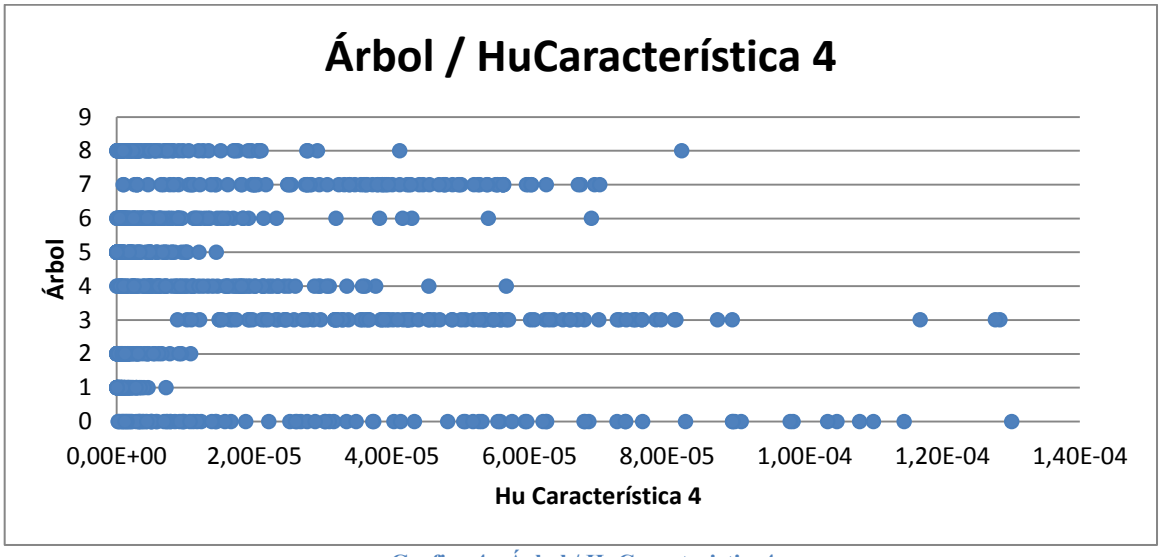
Grafica 1 - Árbol / HuCaracteristica1



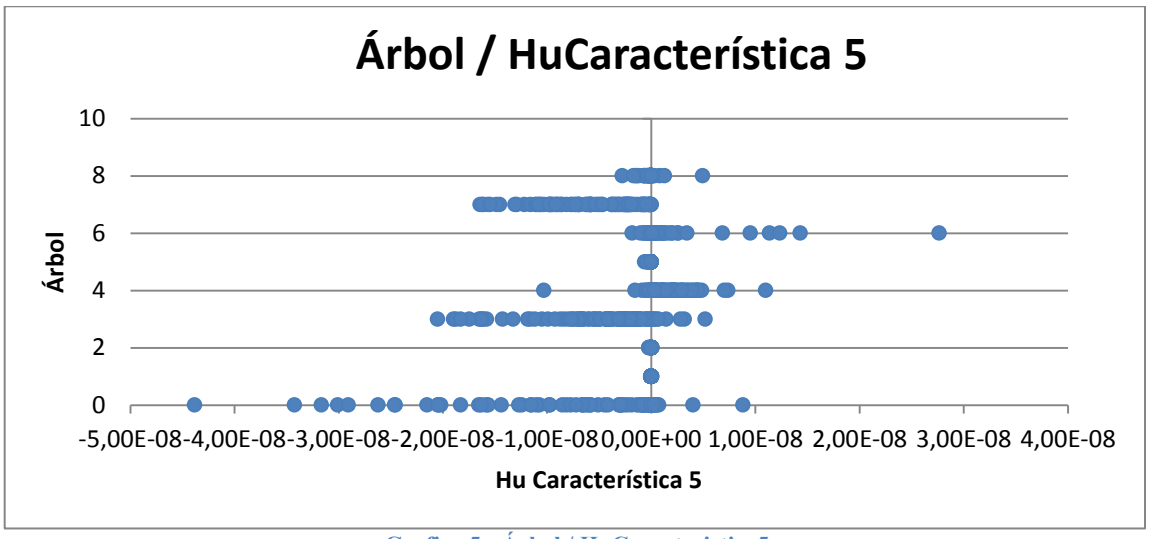
Grafica 2 - Árbol / HuCaracteristica2



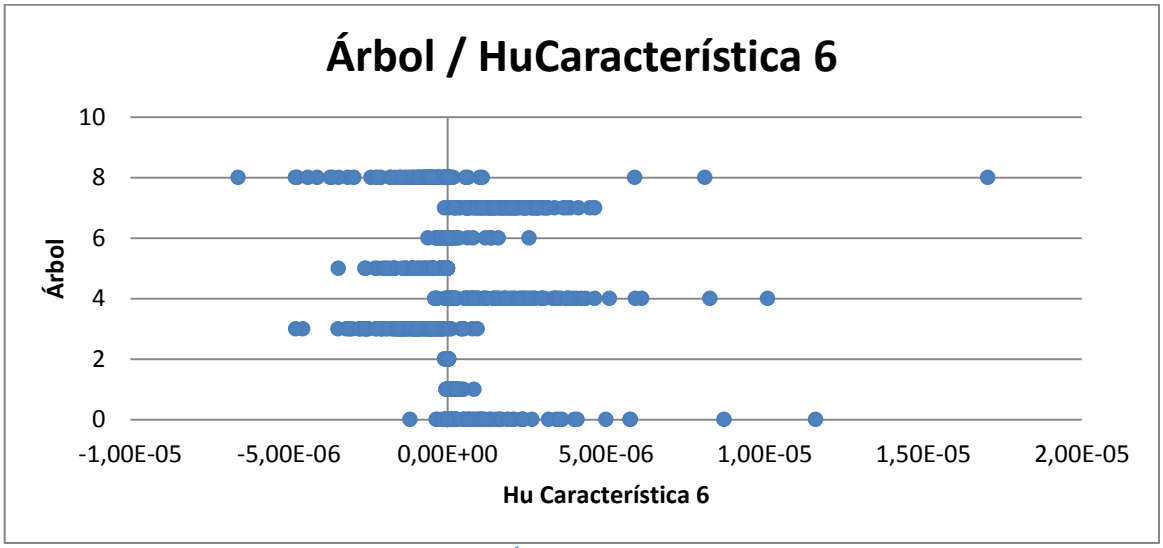
Grafica 3 - Árbol / HuCaracteristica3



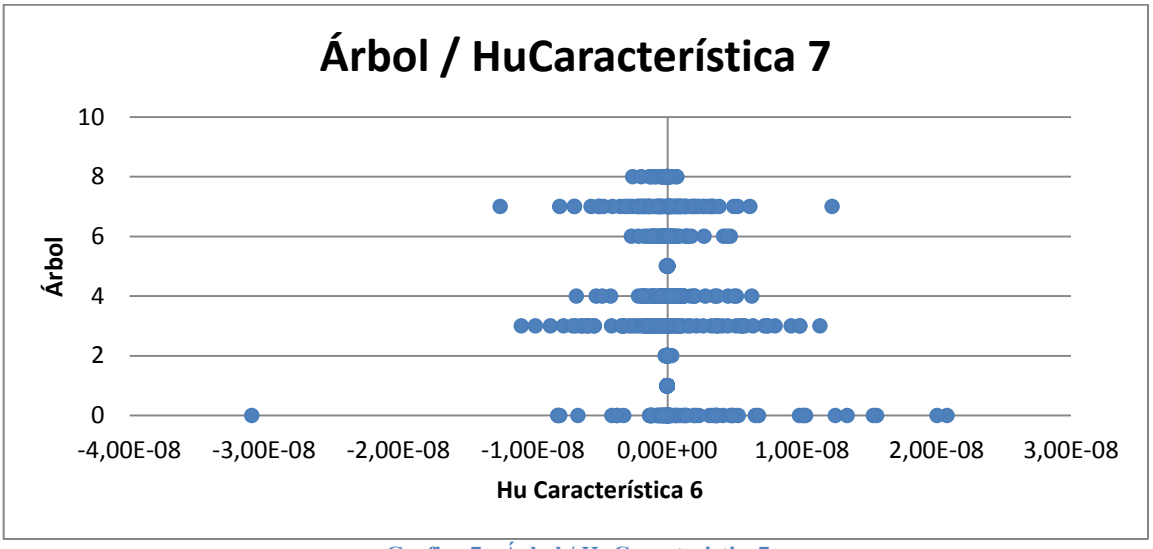
Grafica 4 - Árbol / HuCaracteristica4



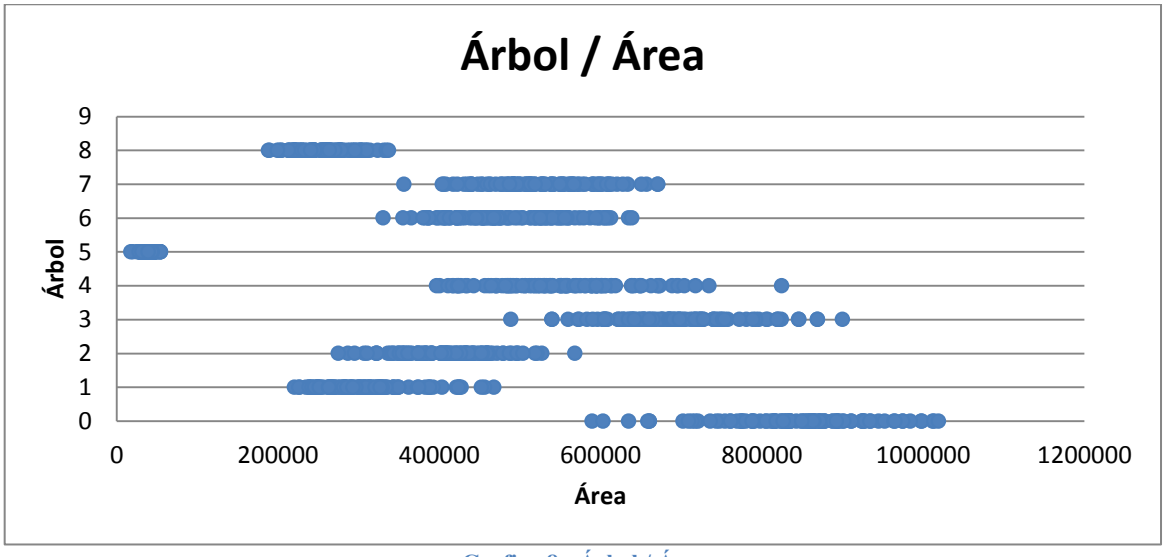
Grafica 5 - Árbol / HuCaracteristica5



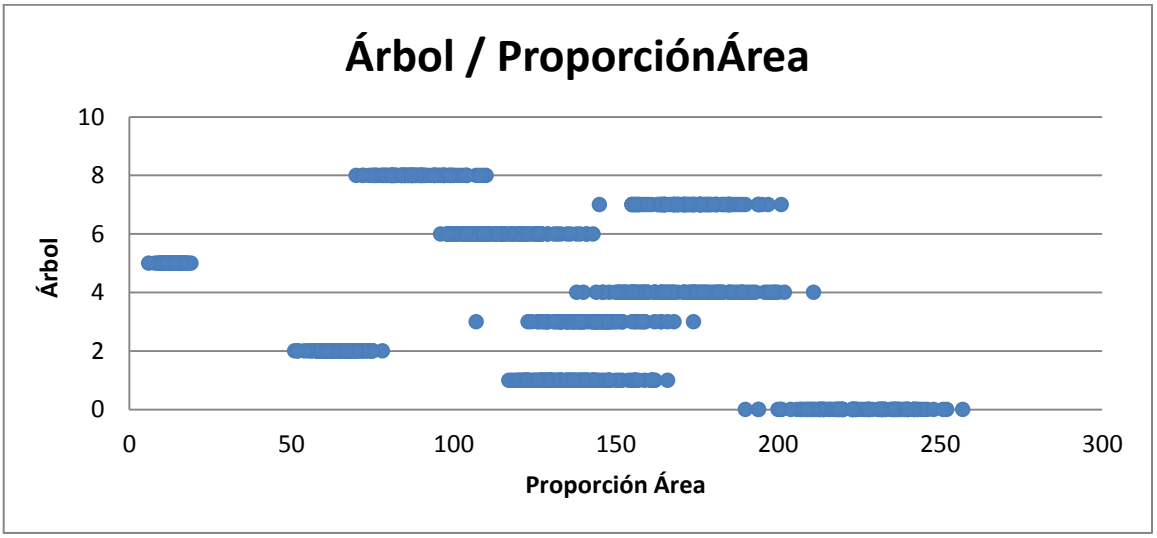
Grafica 6 - Árbol / HuCaracteristica6



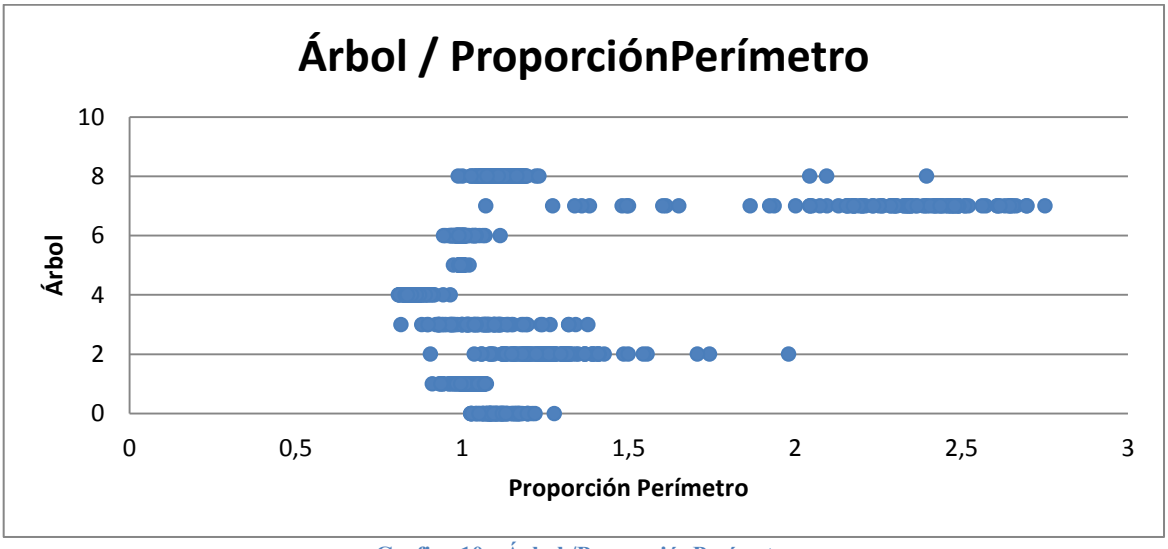
Grafica 7 - Árbol / HuCaracteristica7



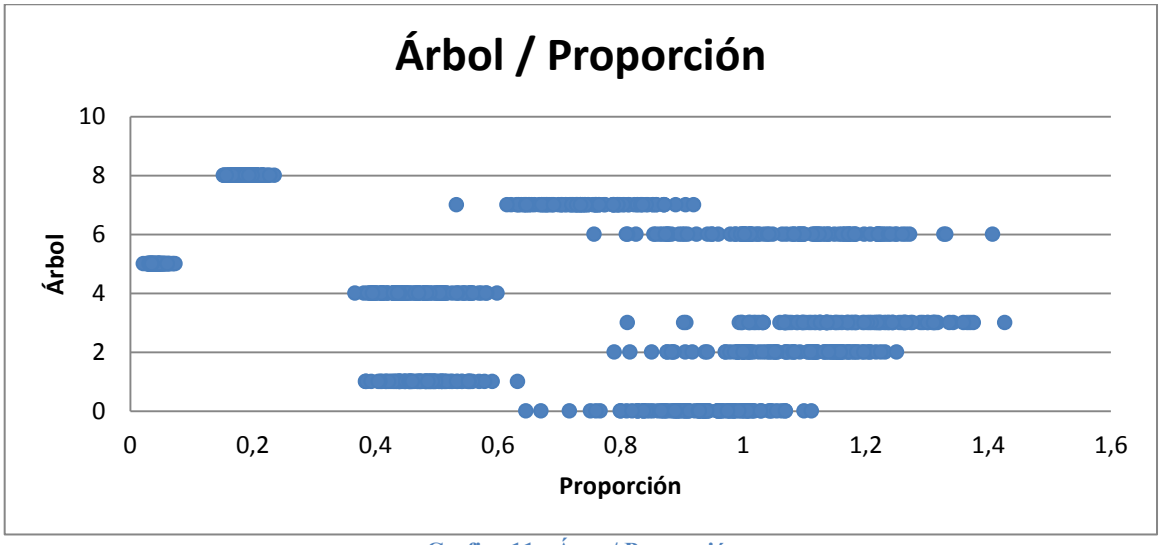
Grafica 8 - Árbol / Área



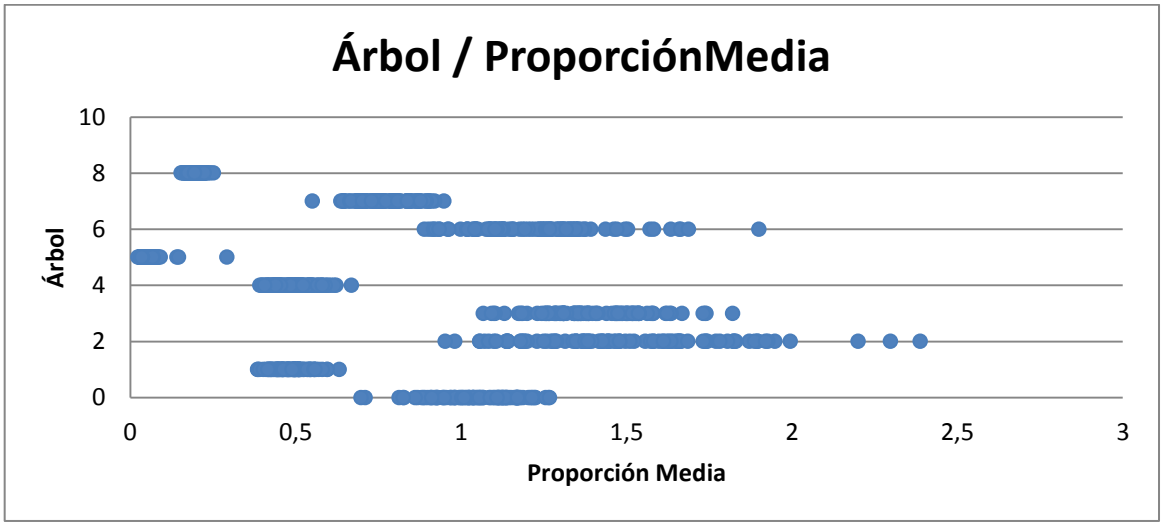
Grafica 9 - Área / ProporciónÁrea



Grafica 10 - Árbol /ProporciónPerímetro



Grafica 11 - Área / Proporción



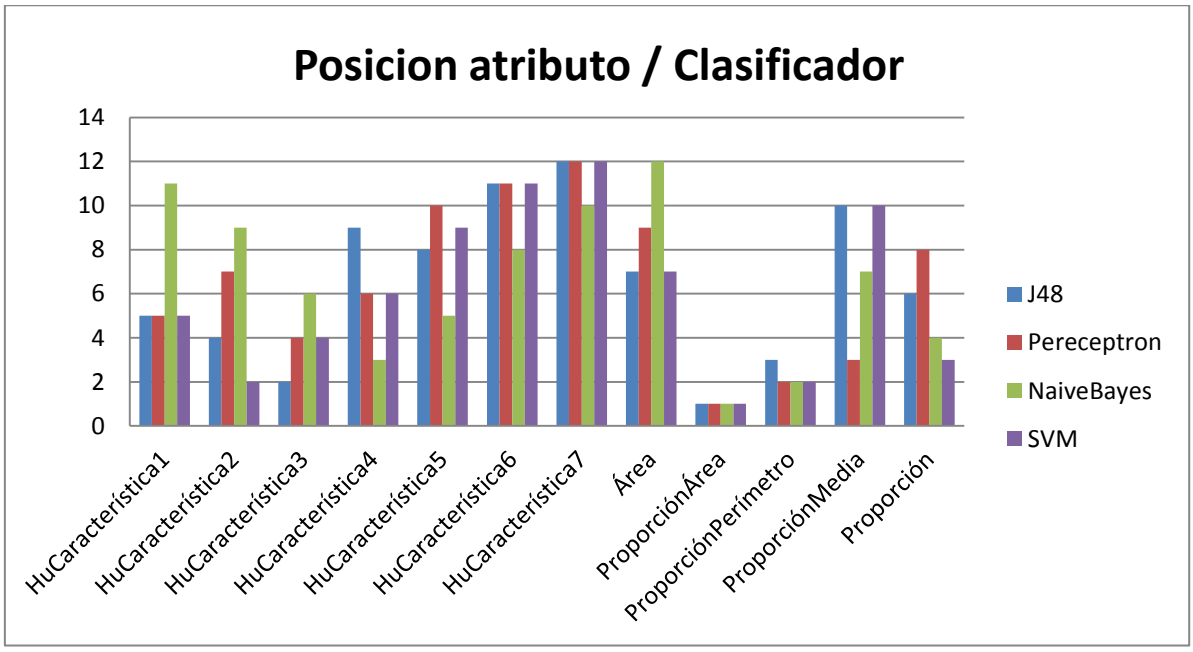
Grafica 12 - Árbol / ProporciónMedia

En las anteriores gráficas se puede observar como hay ciertas clases que solo mirando un atributo (Proporción Área, Proporción Media, Característica Hu 2) se pueden clasificar directamente pues los puntos de dispersión están acotados en unos valores que no comparte con ninguna otra clase. Esto se debe a que esas características son muy discriminatorias entre distintas hojas, lo que significa que se han escogido/extraído bien.

Además, con la ayuda de Weka y la selección de atributos se analizaran a través de distintos algoritmos, como de discriminantes son las características y cuáles son las mejores, de esta forma, si a la hora de realizar las distintas técnicas de aprendizaje alguna lleva más tiempo del estimado (segundos) se puede optar por ignorar los atributos peor valorados y acortar el tiempo de respuesta.

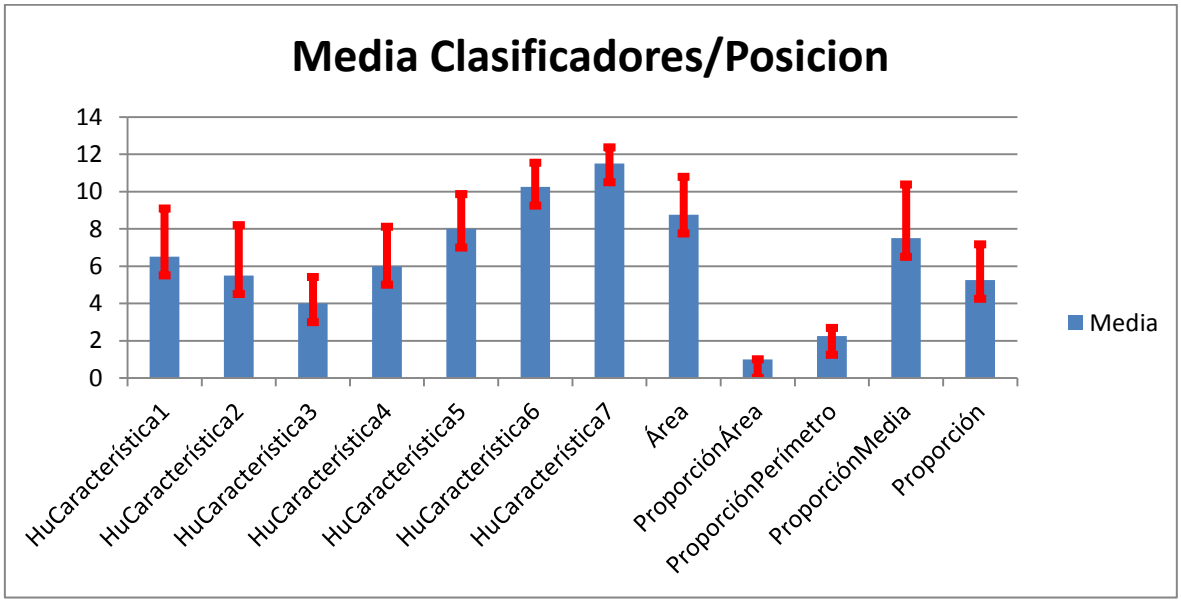
Una de las pruebas se ha realizado con el evaluador ClassifierSubsetEval en el que hay que elegir un clasificador para evaluar, es por esta razón por la que se han escogido los clasificadores que más adelante se usaran (J48, perceptrón multicapa, Naïve Bayes y SVM). Como solución se obtendrá un ranking de las mejores características.

En la siguiente tabla se muestra la posición del 1 al 12 que ha obtenido cada atributo para los distintos clasificadores. Es por eso, que los mejores atributos serán aquellos que tengan menor valor.



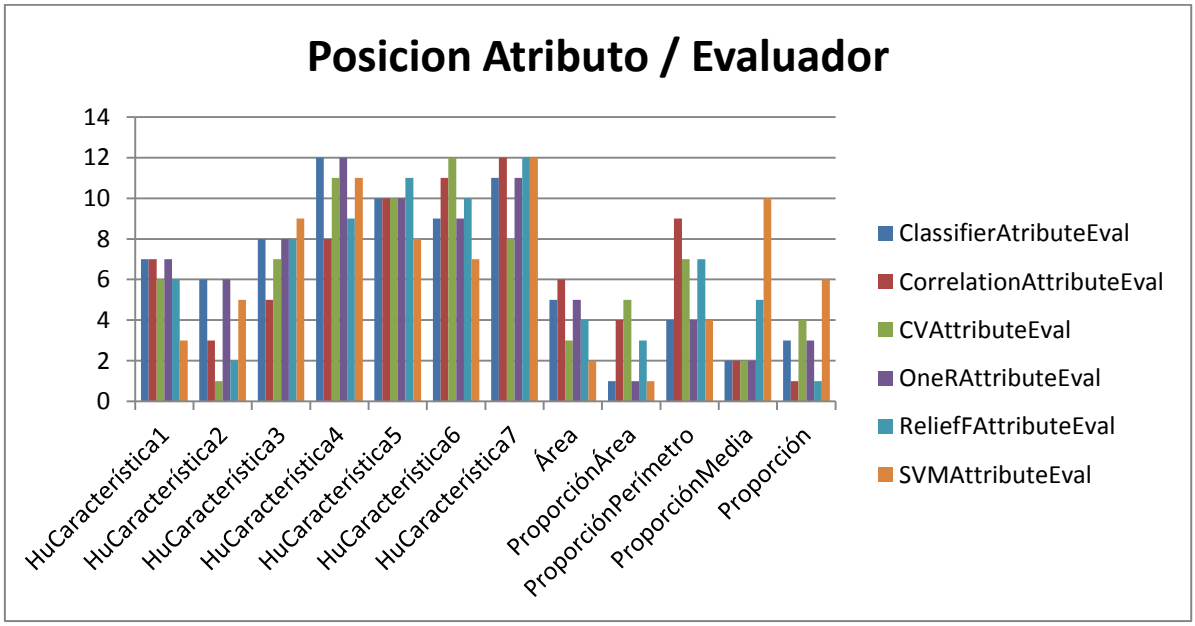
Grafica 13 - Posición atributo / Clasificador

Se observa que los mejores atributos han sido ProporciónÁrea, ProporciónPerímetro y después ya depende del clasificador a usar, aunque si se hace la media de las posiciones (gráfica X) se ve que el siguiente mejor valorado es la HuCaracterística3 seguida de proporción, además estos cuatro atributos son los que tienen la desviación estándar más pequeña.



Grafica 14 - Media Clasificador / Posición

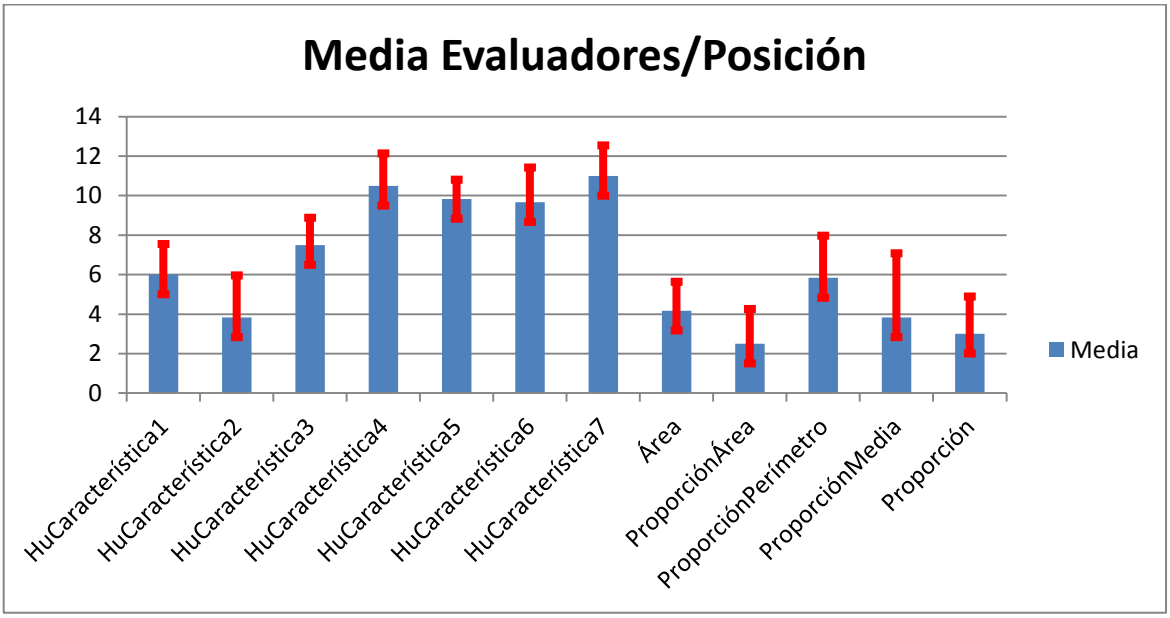
Usando otros evaluadores se han obtenido los siguientes resultados (gráfica X), aunque hay que tener en cuenta que no siempre el ranking va de uno a doce en estos casos, a veces puede producirse empate entre varios atributos o que el algoritmo considere que deberían estar muy cerca en el ranking, pero para poder crear una media más adelante, se ha decidido normalizar los puestos dejándolos como se tenía previsto (del puesto uno al doce).



Grafica 15 - Posición Atributo / Evaluador

Se puede observar que los resultados más o menos se mantienen respecto a la anterior evaluación, en este caso si observamos la gráfica de medias (gráfica X), en primera posición se mantiene ProporciónÁrea, pero Proporción ha mejorado mucho su resultado y ProporciónPerímetro ha empeorado, pero en cambio si se observa la desviación

estándar, proporción media es la que más desviación tiene, por lo que para algún problema en especial podría dar malos resultados así que en consecuencia habría que hacer un seguimiento a este atributo para comprobar cómo se comporta con las distintas técnicas para poder eliminarlo o no. En conclusión, la HuCaracterística 7, 6, 5 y 4 son la que menos aportan y en caso de tener que reducir características esas son las que habría que ignorar.



Grafica 16 - Media Evaluadores / Posición

Por último, hay que destacar que se volverá a usar el programa Weka para realizar las distintas experimentaciones y en general en los modelos de aprendizaje supervisado se usara el 66% de separación (Split) en los datos, lo cual significa que del total de las muestras se dividirán en un 66% para entrenar los distintos modelos y un 44% para comprobar los resultados.

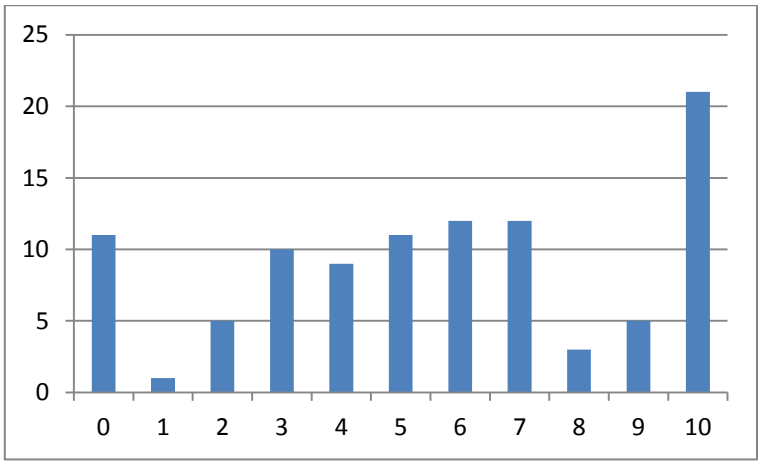
4.1 Técnicas de aprendizaje no supervisado

EM: Con este algoritmo se obtendrá un número de agrupaciones de forma automática comprobando así si los datos están bien generados. Los resultados deberían aproximarse al número de clases incluidas en el conjunto de datos y se debe eliminar (ignorar) el atributo árbol, dado que es la clasificación de cada muestra.

A continuación, algunos de los resultados obtenidos con EM

```
Clustered Instances

0       33 ( 11%)
1        3 (  1%)
2       16 (  5%)
3       31 ( 10%)
4       28 (  9%)
5       33 ( 11%)
6       37 ( 12%)
7       37 ( 12%)
8        8 (  3%)
9       15 (  5%)
10      64 ( 21%)
```



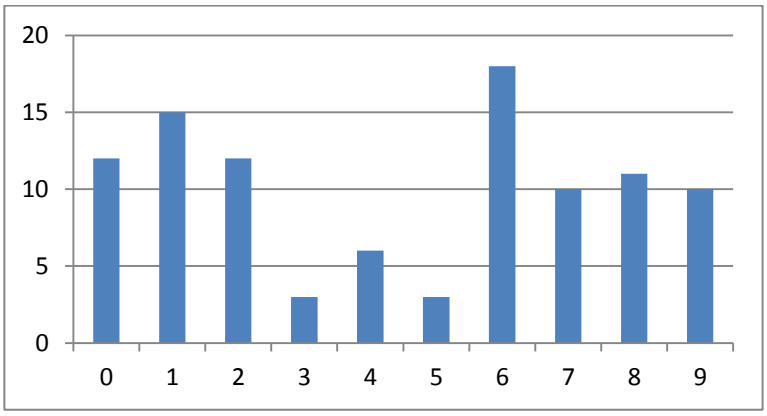
Grafica 17 - Resultados EM

Como se puede ver, esta técnica genera un total de once clusters pero entre ellos hay dos que tienen un porcentaje muy bajo (1% y 3%), si se ignoran, el resultado sería de nueve clusters (el número de clases distintas en el conjunto de datos) aunque hay un cluster con un porcentaje realmente alto (21%) y otros dos un poco bajo (5%)

K-medias: Se usara el algoritmo SimpleKMeans de Weka. En este caso, hay que elegir el número de cluster en los que se dividirá el conjunto de muestras, por lo tanto una buena solución será cuando los clusters importantes se acerquen a nueve y el porcentaje de instancias serán parecidas.

```
Clustered Instances

0       38 ( 12%)
1       45 ( 15%)
2       38 ( 12%)
3        8 (  3%)
4       19 (  6%)
5        9 (  3%)
6       55 ( 18%)
7       29 ( 10%)
8       35 ( 11%)
9       29 ( 10%)
```

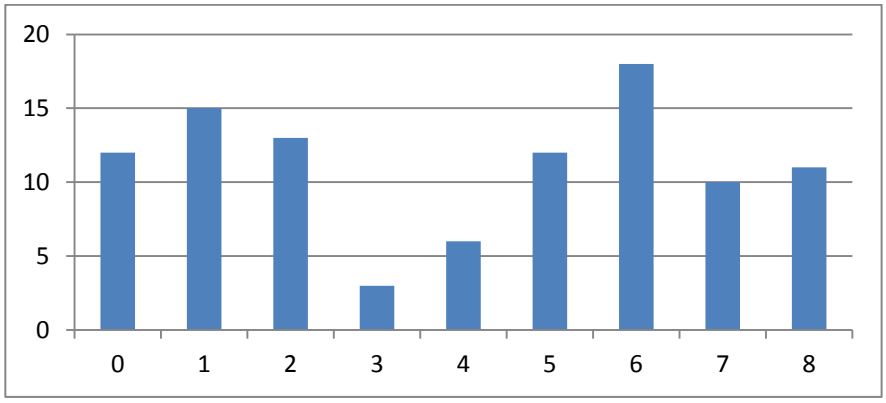


Grafica 18 - Resultados K-medias (K=10)

En este caso se eligió K=10, se puede observar en la gráfica 18 que hay dos clusters con menos porcentaje que el resto, cuando solo debería existir uno que destacara por abajo o por arriba. En cambio, si se elige K=9, en la gráfica 19 se puede ver que lo que ocurre es que hay un conjunto con un porcentaje algo escaso, quedando un total de ocho clusters, pero la diferencia (en porcentaje) entre los cluster es menor que con K=10 por lo que aun con ocho cluster es posible que sea mejor solución.

```
Clustered Instances

0       38 ( 12%)
1       45 ( 15%)
2       39 ( 13%)
3        8 (  3%)
4       19 (  6%)
5       37 ( 12%)
6       55 ( 18%)
7       29 ( 10%)
8       35 ( 11%)
```

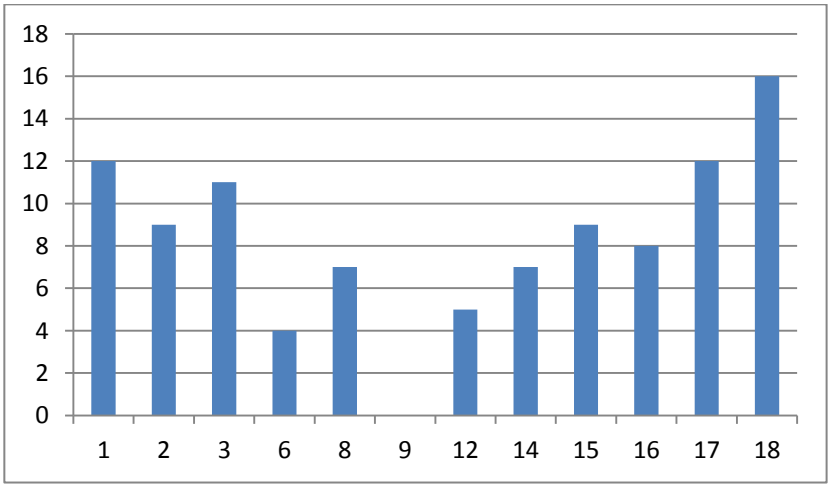


Grafica 19 - Resultados K-medias (K=9)

Cobweb: Este algoritmo generara un árbol con muchos clusters y al final agrupara en un número final de cluster las instancias, siendo esa la razón por la que los clusters no están nombrados de forma consecutiva.

```
Clustered Instances

1       36 ( 12%)
2       27 (  9%)
3       35 ( 11%)
6       12 (  4%)
8       21 (  7%)
9        1 (  0%)
12      16 (  5%)
14      20 (  7%)
15      28 (  9%)
16      23 (  8%)
17      36 ( 12%)
18      50 ( 16%)
```



Grafica 20 - Resultados Cobweb

En este caso se generan un total de doce clusters, aunque uno se puede ignorar por qué contiene un valor de 0%, quedando un total de once clusters, de entre los cuales hay dos con un porcentaje algo más bajo de la media y otro con un porcentaje muy por encima de la media. Esto quiere decir que no ha dividido bien los datos pues se han tenido más clusters que clases y con porcentajes de instancias muy diferentes entre ellos. Por otra parte, ha sido necesario ajustar mucho los parámetros de entrada de esta técnica y este ha sido el caso que mejor resultado ha dado.

4.2 Técnicas de aprendizaje supervisado

Clustering

Se va a contrastar los resultados de los algoritmos de clustering no supervisado con los mismos algoritmos pero esta vez supervisados y ver si los resultados son buenos o no.

EM: Se han introducido los mismos parámetros pero esta vez se han evaluado los resultados con el atributo solución árbol.

```
Class attribute: Arbol
Classes to Clusters:

  0   1   2   3   4   5   6   7   8   9  10  <-- assigned to cluster
  0   0   2  39   0  59   0   0   0   0   0 | 0
  0 100   0   0   0   0   0   1   0   0   0 | 1
  0   0   0   0   0   0   0   0   0  98   0 | 2
 99   0   0   0   0   0   0   0   1   0   0 | 3
  0   0   0   0   0   0   0  99   0   0   0 | 4
  0   0   0   0   0   0 101   0   0   0   0 | 5
  0   0   0   0 100   0   0   0   0   0   0 | 6
  0   0   0   0   0   0   0   0 100   0   0 | 7
  0   0  63   0   0   0   0   0   0   0  34 | 8

Incorrectly clustered instances :       77.0      8.5938 %
```

Ilustración 58 - Resultados Supervisado EM

Como se puede observar a tenido un 91,4062% (100 – 8.593) de acierto, lo que es un muy buen resultado.

K-medias: En este caso con los mismos parámetros que en el caso no supervisado ha dado los siguientes resultados.

```
Class attribute: Arbol
Classes to Clusters:

  0   1   2   3   4   5   6   7   8  <-- assigned to cluster
  0   0   0   0  32   0   0   0  68 | 0
  0   0   0   0   0   0 101   0   0 | 1
  0   0  98   0   0   0   0   0   0 | 2
 97   0   0   0   3   0   0   0   0 | 3
  0   0   0   0   0  94   5   0   0 | 4
  0 101   0   0   0   0   0   0   0 | 5
  0 100   0   0   0   0   0   0   0 | 6
  3   0   7  88   0   0   1   1   0 | 7
  0   0   0   0   1   1   2  93   0 | 8

Incorrectly clustered instances :      156.0     17.4107 %
```

Ilustración 59 - Resultados Supervisado K-medias (K= 9)

Estos resultados son más bajos de lo que se esperaba (82,6%) por lo que se decidió a usar el mismo número de clusters que con el algoritmo EM, once clusters distintos y la solución mejoro notablemente, incluso casi idéntica que con el algoritmo EM con un 91,074% de acierto.

```
Class attribute: Arbol
Classes to Clusters:

   0   1   2   3   4   5   6   7   8   9  10  <-- assigned to cluster
   0   0   0   0  32   0   0   0  68   0   0 | 0
   0   0   0   0   0   0 101   0   0   0   0 | 1
   0   0  98   0   0   0   0   0   0   0   0 | 2
  97   0   0   0   3   0   0   0   0   0   0 | 3
   0   0   0   0   0  94   5   0   0   0   0 | 4
   0   0   0   0   0   0   0   0   0   0 101 | 5
   0 100   0   0   0   0   0   0   0   0   0 | 6
   3   0   7  24   0   0   1   1   0  64   0 | 7
   0   0   0   0   1   1   2  93   0   0   0 | 8

Incorrectly clustered instances :       80.0      8.9286 %
```

Ilustración 60 - Resultados Supervisado K-medias (K=11)

CobWeb: Si usamos la misma configuración que en caso no supervisado se obtiene un 59,033% de acierto, pero no es el mejor resultado, ajustando un poco los parámetros de entrada se obtiene el siguiente resultado.

```
Class attribute: Arbol
Classes to Clusters:

   1   2  11  12  13  14  16  17  18  <-- assigned to cluster
   0   0  40  54   6   0   0   0   0 | 0
   0  68   0   0   0   8   0   0  25 | 1
   0   0   0   0   0   0   0  98   0 | 2
   0  84   0   0   0  10   0   0   6 | 3
   0  27   0   1  25  46   0   0   0 | 4
 101   0   0   0   0   0   0   0   0 | 5
   0   0   0   0   0   0  75   0  25 | 6
   0  12   0   0   0   0   0   0  88 | 7
   0  10   0   0  20  67   0   0   0 | 8

Incorrectly clustered instances :       304.0     33.9286 %
```

Ilustración 61 - Resultados Supervisado Cobweb

El acierto mejora hasta el 66,0714%, pero aun así es un resultado mucho peor que los obtenidos anteriormente.

Clasificación

Se espera que los mejores resultados se obtengan con estas técnicas pues el problema de reconocimiento de árboles a través de la foto de una hoja es un problema de clasificación más que de agrupación.

Naïve Bayes: Para este clasificador solo ha hecho falta cambiar el parámetro de KernelEstimator al valor true para obtener los mejores resultados.

```
Correctly Classified Instances         303               99.3443 %
Incorrectly Classified Instances         2                0.6557 %
Kappa statistic                          0.9926
Mean absolute error                      0.0016
Root mean squared error                  0.0364
Relative absolute error                  0.8268 %
Root relative squared error             11.567  %
Coverage of cases (0.95 level)          99.6721 %
Mean rel. region size (0.95 level)      11.2204 %
Total Number of Instances              305
=== Confusion Matrix ===
  a  b  c  d  e  f  g  h  i   <-- classified as
 32  0  0  0  0  0  0  0  0 |  a = 0
  0 33  0  0  1  0  0  0  0 |  b = 1
  0  0 37  0  0  0  0  0  0 |  c = 2
  0  0  0 30  0  0  0  1  0 |  d = 3
  0  0  0  0 29  0  0  0  0 |  e = 4
  0  0  0  0  0 27  0  0  0 |  f = 5
  0  0  0  0  0  0 38  0  0 |  g = 6
  0  0  0  0  0  0  0 41  0 |  h = 7
  0  0  0  0  0  0  0  0 36 |  i = 8
```

Ilustración 62 - Resultados Naïve Bayes

Como se puede observador, el porcentaje de acierto ha sido muy alto, tal vez demasiado alto por lo que se realizaran varias pruebas más con diferentes técnicas para comprobar si estos valores han podido ser por una sobreadecuación en los datos, un error, o un problema en la conjunto de datos.

SVM: En este caso, se han tenido que normalizar los datos, y usar una función de kernel de tipo lineal (u*v) para obtener buenos resultados, además el parámetro cost (coste) se incrementó a 3 y se activó la opción de probabilityEstimates (probabilidad estimada).

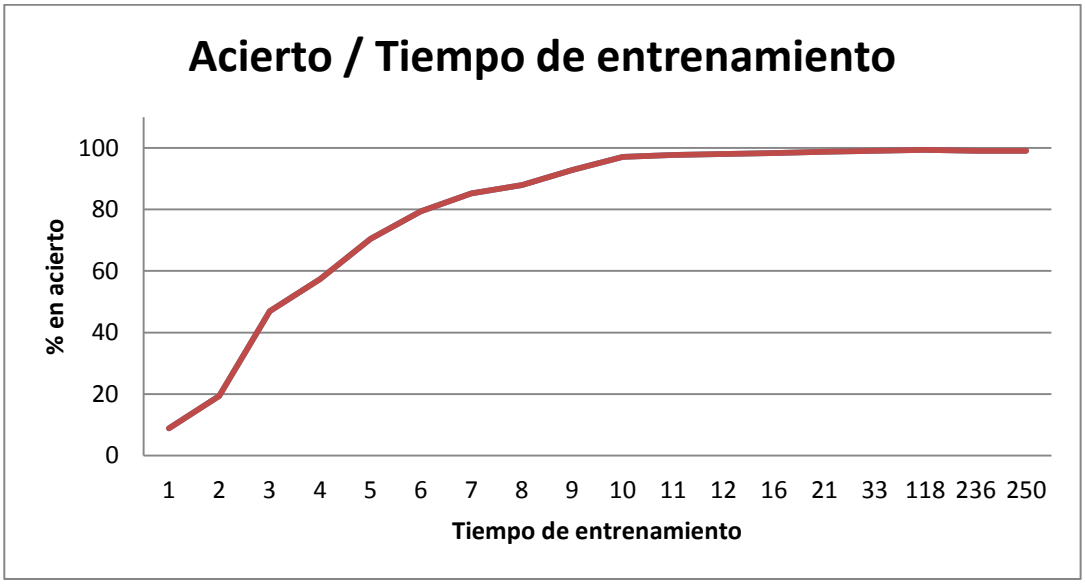
```
Correctly Classified Instances         304               99.6721 %
Incorrectly Classified Instances         1                0.3279 %
Kappa statistic                          0.9963
Mean absolute error                      0.0189
Root mean squared error                  0.0523
Relative absolute error                  9.5355 %
Root relative squared error             16.6215 %
Coverage of cases (0.95 level)         100      %
Mean rel. region size (0.95 level)      23.0601 %
Total Number of Instances              305
=== Confusion Matrix ===

  a  b  c  d  e  f  g  h  i   <-- classified as
 31  0  0  0  0  0  0  0  1 |  a = 0
  0 34  0  0  0  0  0  0  0 |  b = 1
  0  0 37  0  0  0  0  0  0 |  c = 2
  0  0  0 31  0  0  0  0  0 |  d = 3
  0  0  0  0 29  0  0  0  0 |  e = 4
  0  0  0  0  0 27  0  0  0 |  f = 5
  0  0  0  0  0  0 38  0  0 |  g = 6
  0  0  0  0  0  0  0 41  0 |  h = 7
  0  0  0  0  0  0  0  0 36 |  i = 8
```

Ilustración 63 - Resultados SVM

El resultado se aproxima mucho a la solución obtenida con el NaiveBayes, produciendo las mismas inquietudes.

Red de neuronas multicapa: Con esta técnica hay que tener cuidado a sobre especializar la red a los datos de prueba, por lo que se ha entrenado con distintos tiempos la red para ver la evolución en el entrenamiento.



Grafica 21 - Acierto / Tiempo de entrenamiento

Se puede observar como a medida que se aumenta el tiempo de entrenamiento el porcentaje de acierto se eleva, hasta las 236 iteraciones, que aunque en la gráfica no se aprecie, empeora un 0,32% (a partir de ese punto ya no empeora más)

```
Correctly Classified Instances         303               99.3443 %
Incorrectly Classified Instances         2                0.6557 %
Kappa statistic                          0.9926
Mean absolute error                      0.0082
Root mean squared error                  0.043
Relative absolute error                  4.1451 %
Root relative squared error             13.6477 %
Coverage of cases (0.95 level)          99.6721 %
Mean rel. region size (0.95 level)      12.0947 %
Total Number of Instances              305
=== Confusion Matrix ===
  a  b  c  d  e  f  g  h  i   <-- classified as
 31  0  0  0  0  0  0  0  1 |  a = 0
  0 34  0  0  0  0  0  0  0 |  b = 1
  0  0 37  0  0  0  0  0  0 |  c = 2
  0  0  0 31  0  0  0  0  0 |  d = 3
  0  0  0  0 29  0  0  0  0 |  e = 4
  0  0  0  0  0 27  0  0  0 |  f = 5
  0  0  0  0  0  0 38  0  0 |  g = 6
  0  0  0  0  0  0  0 41  0 |  h = 7
  0  0  0  0  1  0  0  0 35 |  i = 8
```

Ilustración 64 - Resultados Red de Neuronas

Por otra parte, al fijarse a que clase pertenece la instancia que ha dado error, se puede observar que es el mismo fallo que con el algoritmo de SVM y mirando también la clasificación del sobreentrenamiento, el error era parecido al del NaiveBayes.

Árbol de Decisión J48: Con este modelo se consiguen los mismos resultados que con NaiveBayes, es por eso por lo que en esta ocasión se mostraran los resultados con un entrenamiento basado en la validación cruzada, únicamente para comprobar si los resultados se mantienen. Esto significa que dividirá las muestras en varios subconjuntos, entrenara con una parte de los subconjuntos y comprobara con otra parte, realizando esto varias rondas usando así todos los datos para entrenar y comprobar.

```
Correctly Classified Instances         882               98.4375 %
Incorrectly Classified Instances        14                1.5625 %
Kappa statistic                          0.9824
Mean absolute error                      0.0043
Root mean squared error                  0.0588
Relative absolute error                  2.1649 %
Root relative squared error             18.7216 %
Coverage of cases (0.95 level)          98.4375 %
Mean rel. region size (0.95 level)      11.1483 %
Total Number of Instances              896
=== Confusion Matrix ===
   a   b   c   d   e   f   g   h   i   <-- classified as
  99   0   0   0   0   0   0   0   1 |   a = 0
   0  99   0   0   1   0   0   1   0 |   b = 1
   0   0  98   0   0   0   0   0   0 |   c = 2
   0   0   0  97   0   0   0   3   0 |   d = 3
   1   0   0   0  98   0   0   0   0 |   e = 4
   0   0   0   0   0 100   1   0   0 |   f = 5
   0   0   0   0   0   0 100   0   0 |   g = 6
   0   0   0   2   0   0   0  98   0 |   h = 7
   1   0   0   1   1   0   0   1  93 |   i = 8
```

Ilustración 65 - Resultados Árbol de decisión

Se puede observar que los datos siguen manteniéndose cerca del 100%. El árbol está incluido en el anexo X.

Por último quedan los metaclasificadores, que se han dejado para el final dado que se usaran algunos de los modelos usados anteriormente. Aunque con los resultados obtenidos es muy difícil poder mejorarlos dado que estas técnicas suelen usarse para casos en la que los resultados son más cercanos a porcentajes bajos (60/70/80%).

Bagging: Esta técnica ha dado los mismos resultados exceptuando para la red de neuronas que ha aumentado su acierto hasta el 99,442%.

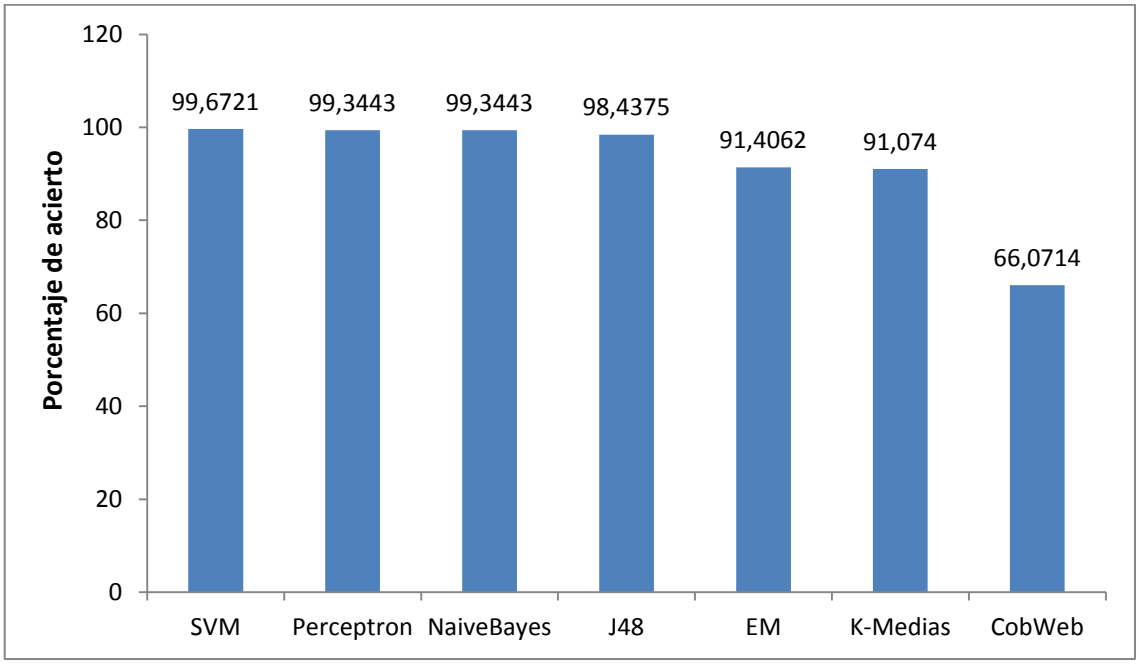
Bosting: Los resultados obtenidos son iguales que con la técnica de Bagging, como mucho mejoran o empeoran en un 1% los resultados obtenidos.

Stacking: Se introdujeron los anteriores modelos vistos, como metaclasificador el propio SVM y los resultados, previsiblemente igualaron el mejor resultado obtenido del 99.6721%.

4.3 Discusión

En este apartado se discutirán los resultados obtenidos con las diferentes técnicas, se compararan y se compararan con los resultados de otras investigaciones.

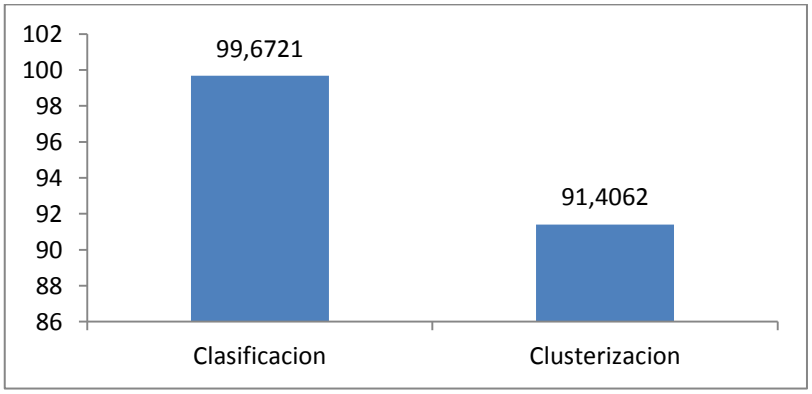
A continuación se muestra una gráfica con los resultados obtenidos de los clasificadores y clusterizaciones supervisadas, ordenándolos de mejor a peor resultado.



Grafica 22 - Resultados finales

Se observa en la gráfica que los mejores resultados se han obtenido con las técnicas de clasificación, estando muy cerca del 100%, y entre estas técnicas el mejor ha sido el SVM, técnica que generalmente está dando muy buenos resultados en los problemas de clasificación, siendo la más utilizada, aunque la diferencia entre estas cuatro técnicas es despreciable (1,2%), pudiendo deberse a que los datos usados en las pruebas se adaptan mejor para el SVM.

Por otra parte, los algoritmos de agrupación, claramente han tenido peores resultados, aunque EM y K-medias han tenido resultados muy óptimos (superando el 90%). Esto se debe a que el problema era principalmente de clasificación como ya había expuesto en apartados anteriores. Aun así, si se compara el mejor resultado de clasificación con el de agrupamiento, la diferencia es escasa (8%), otorgando mayor validez a los resultados del proyecto.



Grafica 23 - Comparación Clasificación vs Clusterizacion

A pesar de tener tan buenos resultados en la parte supervisada, la parte no supervisada no resulto tan buena, en ninguno de los casos consiguieron los datos ajustarse a nueve clases exactas con porcentajes parecidos. Es muy posible que esto se deba a que hay más características de las verdaderamente necesarias (aunque no generar ruido en la parte supervisada, además de que el tiempo de ejecución de todas las pruebas no superaba los dos segundos) tal y como se había visto en las pruebas de selección de datos.

Por otra parte, a la hora de agregar más clases al proyecto es muy posible que su resultado vaya descendiendo poco a poco. En este proyecto al principio se empezaron a hacer las pruebas con siete clases distintas y se obtenían resultados parecidos, así que se introdujo una nueva clase parecida a una que ya existía para ver la solidez de los resultados, los cuales se mantuvieron en niveles. Por último, se agregó una clase nueva pero distinta a las ya existentes y los resultados se mantuvieron, lo cual, significa que las características extraídas no son únicamente validadas para las clases escogidas, sino que posiblemente se extiendan hasta un número amplio. Aun así, habrá un momento o alguna clase que comience a generar cierto error, y el porcentaje de acierto disminuya, pero hay un margen de casi un 9% hasta el 90% el cual seguiría siendo un porcentaje de acierto muy elevado.

Comparando resultados con otros proyectos

A continuación se va a comprobar los resultados obtenidos con los obtenidos en otros proyectos (apartado 2.3). Entre estos proyectos destacaban el *Flavia* y un proyecto desconocido, donde ambos obtenían resultados muy altos, en el primer caso llegaba al 93% de acierto con más de treinta clases distintas y el segundo con un resultado del 99,3% con un número de quince clases distintas. Ambos resultados son muy parecidos a los obtenidos en este proyecto, aunque ambos tienen más clases de hojas distintas que este proyecto. Por otra parte, el proyecto desconocido con resultados del 99,3% también utiliza la técnica del SVM y el proyecto de Flavia usa únicamente una red de neuronas hecha con Matlab**.** Los otros proyectos parecidos a este, tenían resultados más bajos, en el caso del Neuronal Network based recognition system of leaf images, posiblemente se corresponda a que intenta reconocer las hojas a través de otras técnicas y en el caso de *LeafSnap* su mayor problema es que abarca más de 185 clases distintas de árboles, y al buscar valoraciones de dicha aplicación en internet, en casi todos los casos el resultado que presentaba no se parecía al resultado que tenía que dar, exceptuando en algunos casos en los que la iluminación era muy particular, por lo que probablemente el problema venga con la calidad de las imágenes que han usado para el conjunto de datos.

5 Conclusiones y trabajos futuros

En este proyecto se han alcanzado los objetivos de poder identificar un árbol a través de una fotografía, además con unos resultados muy óptimos, por encima de los esperados con más de un 99% de acierto con las distintas técnicas de clasificación. El mejor resultado ha sido de un 99,6721% con el algoritmo de máquinas de soporte vectorial (SVM), además los modelos de clasificación han dado mejores resultados que los modelos de clusterización, con los cuales se han obtenido resultados cercanos al 91% de acierto. De este resultado se concluye que los mejores modelos de aprendizaje automático para este problema son los clasificatorios.

Casi todos los algoritmos (de modelos de clasificación) han tenido resultados parecidos y muy buenos, por lo que se puede deducir que en este caso (reconocimiento de árboles) lo importante no es el modelo escogido a la hora de evaluar, sino el vector característica que se obtiene al final, es decir, lo importante para poder reconocer un árbol a partir de su hoja es poder distinguir bien su morfología (aspecto).

Por otro lado, se tuvo que descartar el intentar alcanzar los objetivos con la fotografía de un árbol general (entero), y centrarse solo en identificarlo con la hoja, donde ya existían varios antecedentes de proyectos que alcanzaban el objetivo de identificar arboles a partir de solo la hoja. Esto se debe a la gran complejidad de extraer las características del árbol entero y de la gran cantidad de causas que pueden provocar que un árbol de la misma especie crezca con formas completamente distintas.

Por último, con el estudio de atributos se ha podido observar que las mejores características han resultado ser las más simples (las características geométricas como el área, perímetro, etc.), y aunque no han dado malos resultados, las que peor resultado han dado han sido las características ligadas a los momentos de Hu, los cuales, aunque no han sido los que más discriminación han otorgado en el proyecto, si se observa el anexo C, se ve cómo dichas características han sido usadas para discernir entre varias clases y por tanto han sido útiles.

Trabajos Futuros

Al final, el número de clases distintas que contiene el conjunto de datos utilizado es de nueve, quedándose un poco escaso si se quisiera poder hacer una aplicación o un proyecto más completo sobre este tema, por ello habría que introducir nuevas especies de árboles, lo que podría suponer una perdida en el porcentaje de acierto, el cual al ser tan elevado tiene bastante margen de error, pero si se quisiera mantener ese porcentaje y agregar clases nuevas se debería agregar algunas características nuevas. A continuación muestran algunas características que se podrían incluir en trabajos futuros.

Hay algunas características nuevas de las que ya se han hablado como por ejemplo la *Transformada de Fourier-Mellin*, con la que se obtendrían varias características invariables a la escala, posición y tamaño, pues esta técnica es muy parecida a la de los momentos invariantes de Hu.

Por otro lado, también se ha hablado en el proyecto de dividir la hoja en horizontal para obtener la proporción entre una mitad y otra, pudiendo así saber si la hoja en concreto

es simétrica o no. Además también se podría realizar la proporción no solo entre esos dos sectores sino entre los cuatro que se generarían, obteniendo nuevas características.

Una característica que se intentó llevar a cabo durante la implementación del proyecto fue la de obtener el número de subhojas que tenía la hoja y la de identificar la cantidad de picos que tenía el margen de esta. Solo agregando esas dos características es muy posible que el porcentaje de acierto en la identificación de árboles se mantuviera alrededor del 95% aun si se agregaran muchas clases nuevas (30 o 40 clases nuevas). El problema por el cual no se terminó implementando en este proyecto, era por que la solución obtenida para cada hoja dependía mucho de los parámetros que se le introducían al método creado.

Por último en cuanto al vector característica, una vez obtenido el contorno de la hoja (eliminando todo el ruido externo e interno), se podría usar dicho contorno para generar una imagen máscara[12] e intentar obtener la venación de la hoja, con la cual se podría realizar otra clasificación para identificar el tipo de venación y sumar el resultado como una nueva característica del proyecto base.

Este proyecto se podría convertir fácilmente en una aplicación móvil dado que se ha utilizado tecnología disponible en Android (OpenCV), con lo que además se podría tener en cuenta la situación geográfica y la estación (invierno, verano, etc.) en el momento en el que se realiza la imagen y así poder delimitar los árboles que podría obtener como solución o incluso poder usar el color de la hoja dado que al saber la estación en la que se realiza la foto, el cambio de color no sería un problema.

Como se puede ver, este proyecto tiene muchas posibilidades de mejoras y ampliaciones.

[12]**Mascara** en informática, es un conjunto de datos que, junto con una operación, permiten extraer selectivamente ciertos datos almacenados.

Bibliografía y Referencias

Se ha utilizado el estándar ISO 690:2010 por norma general exceptuando para aquellos documentos que especifican como quieren ser referenciados.

[1] Clark, P.; Boswell, R. *Practical Machine Learning Tools and Techniques with Java Implementations*, Morgan Kaufmann Publishers. USA. 2000.

[2] Ian H. Witten; Eibe Frank. *Practical Machine Learning Tools and Techniques*. Morgan Kaufmann. USA. 2005.

[3] Molero, Guillermo. *Desarrollo de modelo de técnicas de Minería de Datos*. Maestría en Ciencia e Ingeniería de la Computación, 2008 UNAM, México.

[4] Molero Castillo, Guillermo; Meda Campaña, María Elena. *Integración de Minería de Datos y Sistemas Multiagente: un campo de investigación y desarrollo*. Ciencias de la Información, 2010, (Septiembre-Diciembre)

[5] Quinlan, J.R; *C4.5 Programs for Machine Learning. San Mateo,* CA: Morgan Kaufmann. 1992

[6] S. Haykin*, Neural Networks and Learning Machines (3rd Edition)*, Prentice Hall, 2009

[7] S. Haykin, *Neural Networks*: A Comprehensive Foundation, 2nd Edition, Prentice-Hall, 1999.

[8] *Redes Neuronales*. Universidad de Salamanca. [en línea] Disponible en <http://avellano.usal.es/~lalonso/RNA>

[9] *Estudio de Redes Neuronales de DARPA.* AFCEAInternational Press, p. 60

[10] Bayes, Thomas. *Essay Towards Solving a Problem in the Doctrine of Chances.* 1763; reimpreso en Biometrika,45 (1958) 296-315.

[11] Cortes, C.; Vapnik, V. *Support-vector networks*. Machine Learning 20: 273. 1995.

[12] Cristianini, Nello; Shawe-Taylor, John. *An Introduction to Support Vector Machines and other kernel-based learning methods*, Cambridge University Press, 2000.

[13] MacQueen, J. B *Some methods for classification and analysis of multivariate observations. Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and*

Probability, Volume 1: Statistics, 281--297, University of California Press, Berkeley, Calif., 1967.

[14]Llanos Alfonzo, David; Miguel Romero, Ibis. *Redes de aprendizaje supervisado y no supervisado*. Publicado por Libox, 5 junio de 2013.

[15] Fisher, Douglas. *Knowledge acquisition via incremental conceptual clustering*. Machine Learning 2: 139–172. Septiembre 1987.

[16] *Introduccion a la estimacion con datos faltantes. Algoritmo EM para poblaciones normales.* [PDF][en linea] 2012-2013 Disponible en <http://www.ugr.es/~fdeasis/Material/MultivarianteGrado/Trabajo_C.pdf>

[17] Dempster, A.P.; Laird, N.M.; Rubin, D.B. *Maximum Likelihood from Incomplete Data via the EM Algorithm*. Journal of the Royal Statistical Society, Series B 39: 1–38. 8 de diciembre de 1977.

[18] Borrajo, Daniel; Fernández, Fernando. *Aprendizaje Automático* UC3M 2012-2013 [Transparencias]

[19] Neeraj Kumar, Peter N. Belhumeur, Arijit Biswas, David W. Jacobs, W. John Kress, Ida Lopez, João V. B. Soares, "*Leafsnap: A Computer Vision System for Automatic Plant Species Identification*," Proceedings of the 12th European Conference on Computer Vision (ECCV), October 2012.

[20] Stephen Gang Wu, y otros, *A Leaf Recognition Algorithm for Plant classification Using Probabilistic Neural Network*, IEEE 7th International Symposium on Signal Processing and Information Technology, Dec. 2007, Cario, Egypt Disponible en <http://flavia.sourceforge.net/>

[21] Anónimo. [Página web] Disponible en <http://qixianbiao.github.io/Leaf.html>

[22] Jens, Langner. *Neuronal Network based recognition system of leaf images*. LigthSpeed Communications GbR. Germany, 18 de Abril de 2006

[23] Colaboradores de Wikipedia. *Morfología foliar* [en línea]. Wikipedia, La enciclopedia libre, 2014. Disponible en <http://es.wikipedia.org/w/index.php?title=Morfolog%C3%ADa_foliar&oldid=72978418>

[24] Vaclav Vetvicka. *El gran libro de árboles y arbustos*. Susaeta ediciones, S. A, Madrid. 1984 Aventium Praha.

[25] Tanner Helland. *"Seven grayscale conversion algorithms (with pseudocode and VB6 source code)"* [en línea]. (4 de Junio, 2012) <http://www.tannerhelland.com/3643/grayscale-image-algorithm-vb6/>

[26] Opencv dev team. *Basic Thresholding Operations.* [en linea]. (Abril de 2014) Disponible en:
<http://docs.opencv.org/doc/tutorials/imgproc/threshold/threshold.html>

[27] Sobel, I.; Feldman, G. *A 3x3 isotropic gradient operator for image processing.* Presented at a talk the Stanford Artificial Proyect. 1968

[28] J. Canny . *A computational approach to edge detection*, IEEE Trans. Pattern Analysis and Machine Intelligence, vol 8, pages 679-714. (1986)

[29] Curras Martinez, Manuel; Traba Martinez, Lola. *Detección de Bordes,* Visión Artificial. [PDF] 13 de marzo de 2012.

[30] Zolquernie Othman; Habibollah Haron; Mohammed Rafiq Abdul Kadir. *Comparison of Canny and Sobel Edge Detection in MRI Images.* Universidad tecnológica de Malasia

[31] M-K. Hu. *Pattern recognition by moment invariants*. Proc. IRE (Correspondence), 49:pp. 1428, 1961.

[32] M-K. Hu. *Visual pattern recognition by moment invariants*. IRE Trans. on Information Theory, IT-8:pp. 179-187, 1962.

[33] James Orwell Charles Mallah, James Cope. *Plant leaf classification using probabilistic integration of shape, texture and margin features*. *Signal Processing, Pattern Recognition and Applications, in press*. Febrero 2013. Innsbruck, Austria.

[34] J. Ross Quinlan. *Combining instance-based and model-based learning.* In Proceedings of the Tenth International Conference on Machine Learning, pages 236–243, Amherst, MA, June 1993. Morgan Kaufmann.

[35] Mitchell, T. M. *Machine Learning*. McGraw–Hill, 1997. Capıtulo II.

[36] Bullock, T.H., Bennett, M.V.L., Johnston, D., Josephson, R., Marder, E., Fields R.D. 2005. "The Neuron Doctrine, Redux", Science, Volume 310, Issue 5749, Pages 791-793. PMID 16272104. Retrieved on March 19, 2007.

[37] D. Casasent and D. Psaltis. *Scale invariant optical transform*. Optical Engineering, 15(3):258,261, Junio de 1976.

Anexos

Anexo A: Marco regulador

Este proyecto no se ve afectado por ningún marco regulador.

Anexo B: Planificación y presupuesto

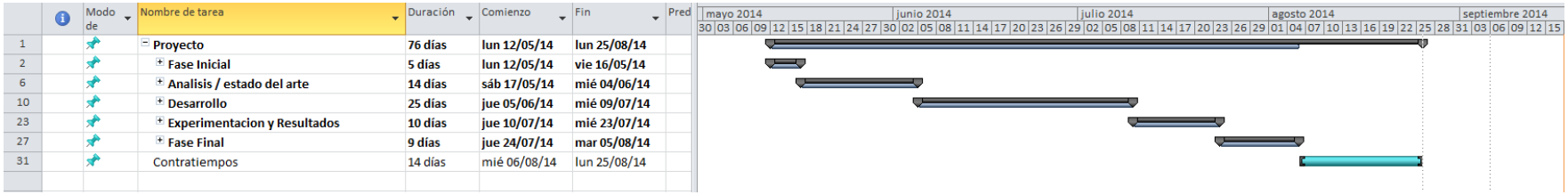
Planificación Inicial

En esta sección se muestra la planificación que se realizó de lo que se tardaría en cada fase del proyecto. Solo se han incluido las fases principales del proyecto. Como se puede observar en la ilustración, la planificación estaba pensada inicialmente en 62 días más 14 de contratiempo haciendo un total de setenta y seis días de trabajo a una media de cuatro horas diarias hace un total de 304 horas de proyecto. No se han tenido en cuenta los festivos (fines de semana incluidos) dado que esos días se suplirán con más horas los días lectivos.

La *fase inicial*, comprende desde el momento en el que se plantea el proyecto hasta que se termina la planificación de este. En la fase de *análisis* se estudia el problema y se documenta sobre este. La fase de *desarrollo* comprende todas las tareas que impliquen la programación o utilización de herramientas (Photoshop, escáner, entorno de programación, etc.) para hacer una tarea. La fase de *experimentación y resultado*, también contiene la discusión de estos. Por último, la *fase final* comprende las conclusiones, presupuesto y la documentación del proyecto.

Por otra parte, la fase de contratiempos incluye posibles desvíos en el tiempo de realización de alguna tarea específica como también posibles incapacidades para realizar el proyecto como posibles días de descanso.

La fase más larga es "Desarrollo", pero se debe a la preparación del entorno, la preparación y obtención del conjunto de datos (con un total de 12 días) y el planteamiento del vector característica, dejando así a la fase de análisis (documentación) como la más larga del proyecto.



	Modo de	Nombre de tarea	Duración	Comienzo	Fin	Pred
1		Proyecto	76 días	lun 12/05/14	lun 25/08/14	
2		Fase Inicial	5 días	lun 12/05/14	vie 16/05/14	
6		Analisis / estado del arte	14 días	sáb 17/05/14	mié 04/06/14	
10		Desarrollo	25 días	jue 05/06/14	mié 09/07/14	
23		Experimentacion y Resultados	10 días	jue 10/07/14	mié 23/07/14	
27		Fase Final	9 días	jue 24/07/14	mar 05/08/14	
31		Contratiempos	14 días	mié 06/08/14	lun 25/08/14	

Ilustración 66 - Planificación inicial

Planificación Final

En esta sección se muestra los tiempos que se han tardado al final al realizar este proyecto (de forma ampliada).

Al final el proyecto se alargó 8 días más, provocado sobre todo por la fase de desarrollo y la fase final con la documentación. Esto es debido a que se tardó más de lo esperado en la parte de preprocesador de las imágenes (una semana más) y la agregación de una sección llamada *ajuste de parámetros* que llevo dos días extras. El resto de fases o se mantuvieron o se alargaron entre uno y cuatro días, tal y como estaba previsto con los contratiempos, aunque se tuvo que aumentar el tiempo de dedicación de cada día, llegando a un total de 433 horas (5 horas de media en la primera parte del proyecto y 6 horas en la fase final).

	Modo de	Nombre de tarea	Duración	Comienzo	Fin	Predeces
1		**Proyecto**	**84 días**	**lun 12/05/14**	**jue 04/09/14**	
2		**Fase Inicial**	**5 días**	**lun 12/05/14**	**vie 16/05/14**	
3		Proposicion del proyecto	1 día	lun 12/05/14	lun 12/05/14	
4		Definición de objetivos	1 día	mar 13/05/14	mar 13/05/14	
5		Planificacion del proyecto	3 días	mié 14/05/14	vie 16/05/14	
6		**Analisis / estado del arte**	**15 días**	**sáb 17/05/14**	**jue 05/06/14**	
7		Documentacion tecnologica	8 días	sáb 17/05/14	mar 27/05/14	
8		Documentacion otros proyectos	3 días	mié 28/05/14	vie 30/05/14	
9		Documentacion Biologica	5 días	sáb 31/05/14	jue 05/06/14	
10		**Desarrollo**	**40 días**	**vie 06/06/14**	**jue 31/07/14**	
11		Preparacion del entorno	2 días	vie 06/06/14	dom 08/06/14	
12		Diseño de la estructura del proyecto	2 días	dom 08/06/14	lun 09/06/14	
13		Planteamiento del vector caracteristica	9 días	lun 09/06/14	jue 19/06/14	
14		Adquisicion del conjunto de datos	5 días	vie 20/06/14	jue 26/06/14	
15		Preprocesado de las imágenes	14 días	vie 27/06/14	mié 16/07/14	
16		**Tratamiento de imagen**	**7 días**	**jue 17/07/14**	**vie 25/07/14**	
17		Escala de grises	1 día	jue 17/07/14	jue 17/07/14	
18		Binarizacion	2 días	vie 18/07/14	lun 21/07/14	
19		Deteccion de bordes	3 días	mar 22/07/14	jue 24/07/14	
20		Ajuste de parametros	2 días	jue 24/07/14	vie 25/07/14	
21		**Extraccion de caracteristicas**	**5 días**	**sáb 26/07/14**	**jue 31/07/14**	
22		Momentos invariantes de Hu	2 días	sáb 26/07/14	lun 28/07/14	
23		Caracteristicas geometricas	3 días	mar 29/07/14	jue 31/07/14	
24		**Experimentacion y Resultados**	**12 días**	**vie 01/08/14**	**lun 18/08/14**	
25		Aprendizaje no supervisado	2 días	vie 01/08/14	lun 04/08/14	
26		Aprendizaje supervisado	10 días	lun 04/08/14	vie 15/08/14	
27		Discusion de resultados	2 días	sáb 16/08/14	lun 18/08/14	
28		**Fase Final**	**13 días**	**mar 19/08/14**	**jue 04/09/14**	
29		Conclusiones	1 día	mar 19/08/14	mar 19/08/14	
30		Presupuesto	1 día	mié 20/08/14	mié 20/08/14	
31		Documentacion	11 días	jue 21/08/14	jue 04/09/14	

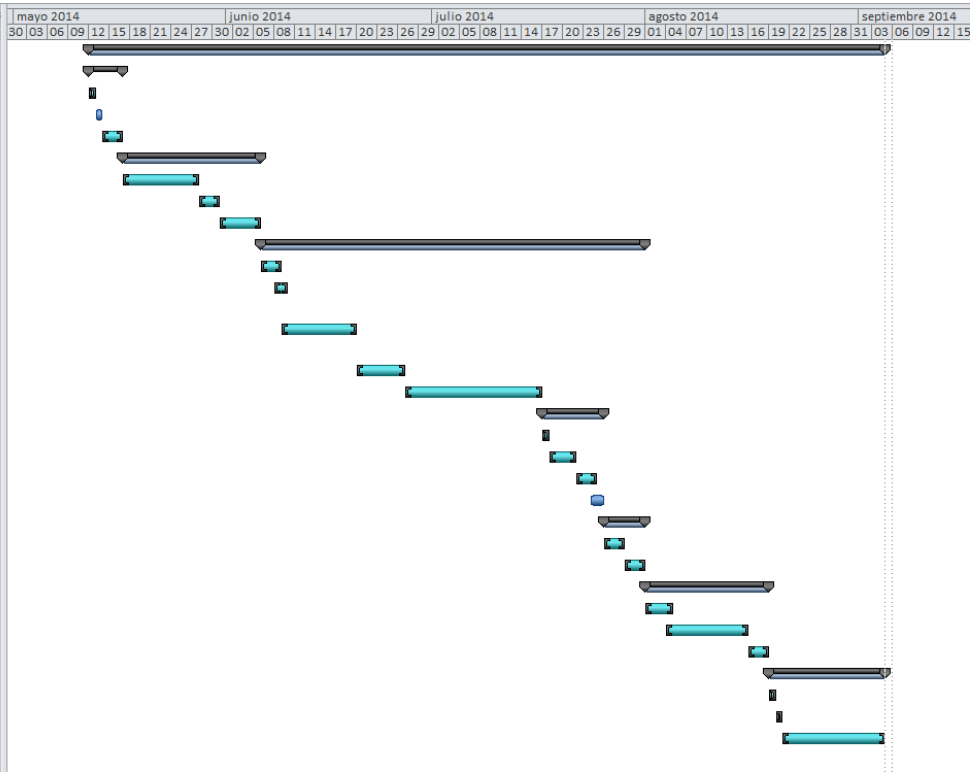


Ilustración 67 - Planificación Final extendida

Presupuesto

1. **Autor:** Mario Carmona Benítez.
2. **Departamento:** Departamento de informática.
3. **Descripción del proyecto**
   a. **Título:** Clasificación Automática del tipo de árbol a partir de las características de sus hojas.
   b. **Duración:** 4 meses (se aproxima a lo alto).
4. **Presupuesto total del proyecto:** 10051,47€
5. **Desglose presupuestario:**

PERSONAL

Apellidos, Nombre	Categoría	Hora de trabajo	Coste por Hora(€)	Coste Total (€)
Carmona Benítez, Mario	Investigador Principal	338	20,00	6760
Carmona Benítez, Mario	Encargado de conjunto de datos	95	18,00	1710
			Total (€)	8470

Tabla 3 - Presupuesto - Personal

EQUIPOS

Descripción	Coste (€)	Uso dedicado al proyecto (%)	Periodo de depreciación (meses)	Coste Imputable*
PC - Sobremesa	500	100	36	55
PC – Portátil	900	10	30	3
Alquiler de Escáner	6€*3Dias = 18	100	0	18
Alquiler cámara fotográfica	35	100	0	35
*En el coste imputable se ha tenido en cuenta que dichos equipos no se habían comprado exclusivamente para el proyecto.			Total (€)	111

Tabla 4 - Presupuesto - Equipos

SOFTWARE

Descripción	Coste (€)	Uso dedicado al proyecto (%)	Periodo de depreciación (meses)	Coste Imputable*
Licencia Windows 7	115	100	30	4
Microsoft office 2010 (Word, Excel)	80	100	36	3

Microsoft office 2010 Project	680	100	36	19
Adobe Photoshop CS4	70	100	48	2
Weka	Free	100	0	0
OpenCV	Free	100	0	0
Eclipse	Free	100	0	0
*En el coste imputable se ha tenido en cuenta que dichos equipos no se habían comprado exclusivamente para el proyecto.			Total (€)	28

Tabla 5 - Presupuesto - Sofware

OTROS COSTES DEL PROYECTO

Descripción	Empresa	Coste por mes (4 meses en total)	Coste Imputable*
Internet	Jazztel	24*4 = 96	40
Luz	Iberdrola	100*4 = 400	125
*En el coste imputable se ha tenido en cuenta que dichos equipos no se habían comprado exclusivamente para el proyecto.		Total (€)	**165**

Tabla 6 - Presupuesto - Otros costes del proyecto

RESUMEN DE COSTES

Concepto	Coste (€)
Personal	8470
Equipos	111
Software	28
Otros	165
TOTAL SIN IVA	8674
IVA (21%)	1821,54
TOTAL CON IVA	10291,54

Tabla 7 - Presupuesto - Resumen de costes

El coste total ha sido de 10291,54€, a una media de 2570€ al mes.

Anexo C: Datos Finales y Modelos

EM

Attribute	Cluster 0 (0.11)	1 (0.11)	2 (0.07)	3 (0.04)	4 (0.11)	5 (0.07)	6 (0.11)	7 (0.11)	8 (0.11)	9 (0.11)	10 (0.04)
HuCaracteristica1											
mean	0.1976	0.201	0.2327	0.2968	0.2471	0.2793	0.2287	0.2746	0.1773	0.1074	0.2307
std. dev.	0.0132	0.018	0.0156	0.0155	0.0124	0.0158	0.0171	0.0196	0.0152	0.0067	0.0125
HuCaracteristica2											
mean	0.0014	0.0114	0.0029	0.0023	0.0488	0.0002	0.0522	0.0265	0.0008	0.0002	0.0063
std. dev.	0.001	0.0035	0.0017	0.0024	0.0051	0.0002	0.0076	0.0079	0.0008	0.0002	0.0024
HuCaracteristica3											
mean	0.0004	0	0.0008	0.0008	0.0001	0.0005	0	0.0011	0.001	0	0.0009
std. dev.	0.0003	0	0.0003	0.0003	0.0001	0.0003	0	0.0003	0.0006	0	0.0003
HuCaracteristica4											
mean	0	0	0	0.0001	0	0	0	0	0	0	0
std. dev.	0	0	0	0	0	0	0	0	0	0	0
HuCaracteristica5											
mean	0	0	0	0	0	0	0	0	0	0	0
std. dev.	0	0	0	0	0	0	0	0	0	0	0
HuCaracteristica6											
mean	0	0	0	0	0	0	0	0	0	0	0
std. dev.	0	0	0	0	0	0	0	0	0	0	0
HuCaracteristica7											
mean	0	0	0	0	0	0	0	0	0	0	0
std. dev.	0	0	0	0	0	0	0	0	0	0	0
Area											
mean	692759.8188	308509.9077	549519.9698	884238.3717	257234.77	819103.3361	38417.4356	547950.5968	487113.5006	410885.3671	484112.9517
std. dev.	76001.7637	55286.363	63204.2577	82610.9667	33801.1513	75503.0864	6970.5945	84940.1866	69302.7085	56428.7377	47857.388
ProporcionArea											
mean	142.9071	137.01	176.5442	231.1338	88.32	221.752	13.2574	169.84	114.9151	64.3878	167.4296
std. dev.	10.2905	12.152	11.244	12.608	8.9408	12.3561	2.3321	15.8276	11.3322	5.4989	8.1379
proporcionPerimetro											
mean	1.0533	1.0231	1.0945	1.0876	0.9987	1.1239	0.9979	0.8545	2.2364	1.2649	1.1478
std. dev.	0.0989	0.0322	0.0432	0.0363	0.0241	0.0514	0.0049	0.0268	0.3622	0.1481	0.0373
Proporcion											
mean	1.1671	0.4784	0.7821	0.847	0.1878	0.9831	0.0433	0.4704	1.0709	1.0749	0.6755
std. dev.	0.1128	0.0475	0.0617	0.0667	0.0187	0.0476	0.0109	0.0539	0.1345	0.1014	0.0472
ProporcionMedia											
mean	1.3981	0.4847	0.8129	0.9359	0.1973	1.1264	0.0514	0.492	1.2265	1.4937	0.6975
std. dev.	0.1511	0.0484	0.0681	0.0831	0.0203	0.0666	0.031	0.0577	0.191	0.2883	0.0493

Ilustración 68 - Modelo EM

Simple K-Means

```
Initial staring points (canopy):

T2 radius: 1,069
T1 radius: 1,336

Cluster 0: 0.216483,0.006919,0.000594,0.00002,0,0.000001,0,552684.961598,151.250384,1.22716,0.833474,0.953364,{651} <0,1,2,3,4,5,6,7,8,9,10>
Cluster 1: 0.237861,0.05051,0.000039,0.000004,0,-0.000001,0,147281.781095,50.60199,0.998295,0.115169,0.124004,{201} <0,1,5,6,7,8,10>
Cluster 2: 0.106668,0.000369,0.000087,0.000002,0,0,0,365686.108108,62.486486,1.425446,1.084768,1.566039,{37} <0,2,3,6,7,8,9>
Cluster 3: 0.184173,0.001195,0.002375,0.000055,0,0.000002,0,443518,110.5,2.168524,0.886578,1.093927,{2} <0,2,3,5,6,7,8,9>
Cluster 4: 0.267885,0.003147,0.000553,0.000122,0,0.000004,0,904190.8,205,1.004084,0.870006,1.027375,{5} <0,4,5,7,8>
Cluster 5: 0.316357,0.030482,0.001331,0.000005,0,0.000001,0,734343,202,0.862352,0.48136,0.495289 <0,1,3,4,5,6,7,8,9,10>
Cluster 6: 0.201874,0.014654,0.000001,0.000001,0,0,0,299087,130,1,0.411205,0.416043 <0,1,2,3,5,6,7,8,9,10>
Cluster 7: 0.224039,0.012451,0.000025,0.000005,0,0.000001,0,388115,155,1.051071,0.490276,0.499479 <0,1,2,3,4,5,6,7,8,9,10>
Cluster 8: 0.247184,0.007703,0.000808,0.000032,0,0.000003,0,560242,179,1.087743,0.651024,0.657192 <0,1,2,3,4,5,6,7,8,9,10>
Cluster 9: 0.195541,0.000326,0.001624,0.000005,0,0,0,523193,126,2.326629,1.091486,1.244835 <0,2,3,5,6,7,8,9>
Cluster 10: 0.244476,0.059381,0.000025,0.000003,0,-0.000001,0,37460,12,1.002679,0.041077,0.04354 <0,1,5,6,7,8,10>


Missing values globally replaced with mean/mode

Final cluster centroids:
                           Cluster#
Attribute               Full Data          0          1          2          3          4          5          6          7          8          9         10
                           (896)      (100)      (100)      (105)       (24)       (36)       (95)      (109)       (94)       (68)       (64)      (101)
=============================================================================================================================================================
HuCaracteristica1           0.217     0.1968     0.2471     0.1109     0.1891     0.2887     0.2769     0.2025     0.2313     0.2802     0.1744     0.2287
HuCaracteristica2          0.0164     0.0014     0.0488     0.0003     0.0007     0.0016     0.0271     0.0114     0.0039     0.0008     0.0008     0.0522
HuCaracteristica3          0.0005     0.0004     0.0001          0     0.0019     0.0008     0.0011     0.0001     0.0009     0.0006     0.0008          0
HuCaracteristica4               0          0          0          0          0     0.0001          0          0          0          0          0          0
HuCaracteristica5               0          0          0          0          0          0          0          0          0          0          0          0
HuCaracteristica6               0          0          0          0          0          0          0          0          0          0          0          0
HuCaracteristica7               0          0          0          0          0          0          0          0          0          0          0          0
Area                   455736.5223  681529.01  257234.77 409873.5619 517612.125 884111.3333 555309.9158 316897.3578 527934.6915 816502.4853 482057.4688 38417.4356
ProporcionArea             125.2154     141.87      88.32    67.3333    120.1667    225.1667     170.8316    137.9358    173.0106    221.4706    112.3125    13.2574
proporcionPerimetro          1.1849     1.0723     0.9987     1.2787     2.2742      1.064     0.8565     1.0202     1.1141      1.126     2.3866     0.9979
Proporcion                    0.683     1.1667     0.1878     1.0823     0.9987     0.8908     0.4693      0.486     0.7496     0.9485     1.0935     0.0433
ProporcionMedia              0.7933     1.3956     0.1973     1.4875     1.1093     0.9922      0.491     0.4937     0.7778     1.0854      1.257     0.0514
```

Ilustración 69 - Modelo Simple K-Means

CobWeb

Árbol generado por Weka

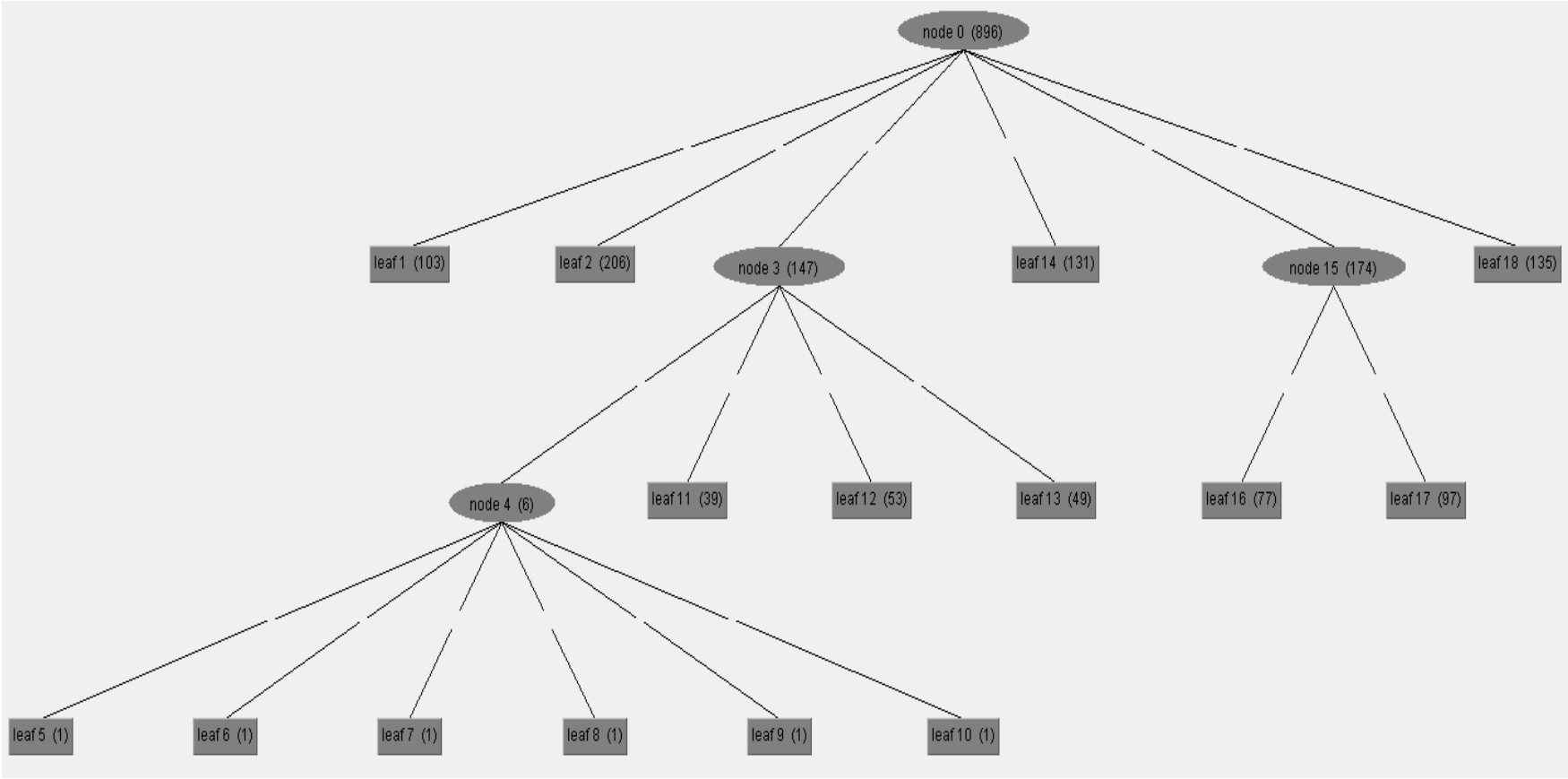


Ilustración 70 - Árbol CobWeb

```
=== Clustering model (full training set) ===

Number of merges: 54
Number of splits: 46
Number of clusters: 19

node 0 [896]
|   leaf 1 [103]
node 0 [896]
|   leaf 2 [206]
node 0 [896]
|   node 3 [147]
|   |   node 4 [6]
|   |   |   leaf 5 [1]
|   |   node 4 [6]
|   |   |   leaf 6 [1]
|   |   node 4 [6]
|   |   |   leaf 7 [1]
|   |   node 4 [6]
|   |   |   leaf 8 [1]
|   |   node 4 [6]
|   |   |   leaf 9 [1]
|   |   node 4 [6]
|   |   |   leaf 10 [1]
|   node 3 [147]
|   |   leaf 11 [39]
|   node 3 [147]
|   |   leaf 12 [53]
|   node 3 [147]
|   |   leaf 13 [49]
node 0 [896]
|   leaf 14 [131]
node 0 [896]
|   node 15 [174]
|   |   leaf 16 [77]
|   node 15 [174]
|   |   leaf 17 [97]
node 0 [896]
|   leaf 18 [135]
```

Ilustración 71 - Modelo CobWeb (Árbol)

Red de Neuronas

Conexiones neuronales

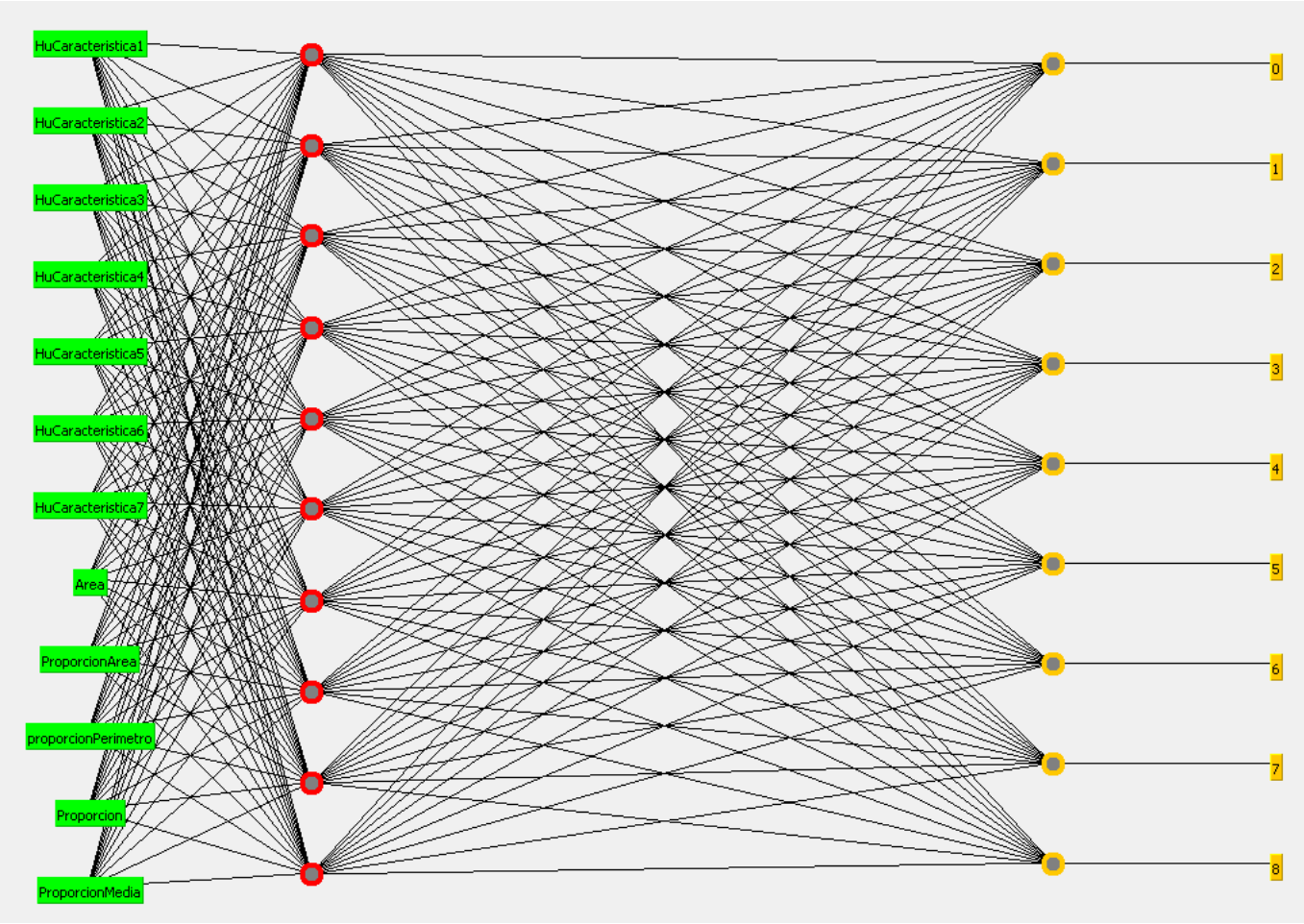


Ilustración 72 - Red Neuronal

Pesos de la red de neuronas

```
=== Classifier model (full training set) ===

Sigmoid Node 0
    Inputs    Weights
    Threshold    -5.362338301971532
    Node 9    -2.065987490062591
    Node 10    4.973971420624647
    Node 11    -2.164029417353713
    Node 12    7.27201743809949
    Node 13    2.292596731806897
    Node 14    -3.517566549367945
    Node 15    -2.6240281180222054
    Node 16    -6.256778822502756
    Node 17    -2.38827392931563
    Node 18    -3.033682534382569
Sigmoid Node 1
    Inputs    Weights
    Threshold    -2.314674238300475
    Node 9    -8.63789073194985
    Node 10    -2.064741521917422
    Node 11    2.150015041790171
    Node 12    -4.587762526576746
    Node 13    -3.7298518430251173
    Node 14    -2.2890272265467537
    Node 15    1.634112286321375
    Node 16    -3.2652340008279555
    Node 17    -3.1736292191016937
    Node 18    7.90947889552235
Sigmoid Node 2
    Inputs    Weights
    Threshold    -4.252134713031264
    Node 9    -4.142456391344822
    Node 10    -3.1114806946408002
    Node 11    -2.449843248939745
    Node 12    0.15984874061241802
    Node 13    -5.91585126655741
    Node 14    0.02834517372653785
    Node 15    -4.315059046723438
    Node 16    1.5316458605960583
    Node 17    4.254469519693991
    Node 18    4.050701120207472
Sigmoid Node 3
    Inputs    Weights
    Threshold    -3.0943624311345315
    Node 9    -2.295236889506624
    Node 10    -2.6575576834113552
    Node 11    2.460096429095911
    Node 12    3.245884966974238
    Node 13    -6.50914352808151
    Node 14    -4.675910344305315
    Node 15    -2.489393711945209
    Node 16    4.184113480496645
    Node 17    -2.714304411715867
    Node 18    -4.911412595097762

Sigmoid Node 4
    Inputs    Weights
    Threshold    -2.8163846945979 42
    Node 9    4.5573008473431855
    Node 10    1.6142219409865968
    Node 11    2.7675955889289954
    Node 12    -5.314099801371839
    Node 13    -5.8424424368668975
    Node 14    -5.484190316565404
    Node 15    3.754821771232275
    Node 16    -2.72823312989874
    Node 17    -2.47546607387175
    Node 18    -6.45160644528003
Sigmoid Node 5
    Inputs    Weights
    Threshold    -3.566673330135022
    Node 9    1.6453172473833684
    Node 10    -3.5353504167884666
    Node 11    -5.421987011445115
    Node 12    -3.862177866483689
    Node 13    -2.29494411497399
    Node 14    0.27635540318533525
    Node 15    0.7156831284026361
    Node 16    -4.896882066593088
    Node 17    7.461823631645072
    Node 18    -1.1566198640322964
Sigmoid Node 6
    Inputs    Weights
    Threshold    -4.767992168882844
    Node 9    3.4533772152301823
    Node 10    3.240672485566235
    Node 11    -5.261822767671217
    Node 12    -2.221766797054003
    Node 13    -5.259730138802811
    Node 14    3.217048995941494
    Node 15    -1.332133434731635
    Node 16    -2.5213701733008276
    Node 17    -8.790418558771316
    Node 18    2.5516517722621295
Sigmoid Node 7
    Inputs    Weights
    Threshold    -1.459987399208699
    Node 9    -1.5003724932880558
    Node 10    -3.289364765305597
    Node 11    -7.6671440038223855
    Node 12    0.4056362352613939
    Node 13    5.13864501190766
    Node 14    2.307107821016763
    Node 15    -2.0945559395332105
    Node 16    2.353542463277347
    Node 17    -6.063137269933895
    Node 18    -2.2322121627528917

Sigmoid Node 8
    Inputs    Weights
    Threshold    0.09551553448075792
    Node 9    -3.203184258046011
    Node 10    -5.21339934214645
    Node 11    2.375475068496363
    Node 12    -9.40660740770065
    Node 13    6.057356037121424
    Node 14    -2.6764272360897583
    Node 15    -3.4876039713603717
    Node 16    -1.5545900016282066
    Node 17    -1.4094370162055776
    Node 18    -5.157477612775506
Sigmoid Node 9
    Inputs    Weights
    Threshold    -0.4943045216264784
    Attrib HuCaracteristica1    1.3446765311370203
    Attrib HuCaracteristica2    4.388478067000973
    Attrib HuCaracteristica3    4.7816784201332005
    Attrib HuCaracteristica4    1.2373330569630363
    Attrib HuCaracteristica5    0.8245889420418301
    Attrib HuCaracteristica6    0.3779110109569577
    Attrib HuCaracteristica7    -0.0739584825672138
    Attrib Area    -0.4319460830746261
    Attrib ProporcionArea    -2.0542847820495655
    Attrib proporcionPerimetro    -6.56987337497646
    Attrib Proporcion    -1.5615283841473675
    Attrib ProporcionMedia    -0.47697722436575984
Sigmoid Node 10
    Inputs    Weights
    Threshold    -2.4674203598622624
    Attrib HuCaracteristica1    1.5612454625321612
    Attrib HuCaracteristica2    4.213129379650791
    Attrib HuCaracteristica3    -2.1405493944942906
    Attrib HuCaracteristica4    -0.3836292693883231
    Attrib HuCaracteristica5    1.754994841842235
    Attrib HuCaracteristica6    -0.7208624368378388
    Attrib HuCaracteristica7    -0.48141724210418047
    Attrib Area    2.7556517086749808
    Attrib ProporcionArea    4.484212926369383
    Attrib proporcionPerimetro    -2.2483656624895416
    Attrib Proporcion    -0.9005094951422491
    Attrib ProporcionMedia    1.4007540131760767
```

Ilustración 73 - Pesos Red de Neuronas

```
Sigmoid Node 12
    Inputs    Weights
    Threshold    -0.9535360418526967
    Attrib HuCaracteristica1    0.09296168274337513
    Attrib HuCaracteristica2    -1.6656796023524234
    Attrib HuCaracteristica3    -2.571302953935699
    Attrib HuCaracteristica4    -0.9545460688360696
    Attrib HuCaracteristica5    1.2778220096202264
    Attrib HuCaracteristica6    0.9786278591680816
    Attrib HuCaracteristica7    1.2926408837100982
    Attrib Area    7.0216063879459165
    Attrib ProporcionArea    0.6127747681494792
    Attrib proporcionPerimetro    4.686485568909117
    Attrib Proporcion    3.658104003952408
    Attrib ProporcionMedia    5.442868267785094
Sigmoid Node 13
    Inputs    Weights
    Threshold    0.9042252941856413
    Attrib HuCaracteristica1    2.2129209055059196
    Attrib HuCaracteristica2    -4.347171129122515
    Attrib HuCaracteristica3    1.507950729055097
    Attrib HuCaracteristica4    -0.8560931764085691
    Attrib HuCaracteristica5    -0.2856700589987346
    Attrib HuCaracteristica6    3.2822160078279823
    Attrib HuCaracteristica7    0.029606092906908528
    Attrib Area    -3.3969866809292673
    Attrib ProporcionArea    5.2377462448630565
    Attrib proporcionPerimetro    6.968400425324615
    Attrib Proporcion    1.884622896276795
    Attrib ProporcionMedia    -3.5063154032580894
Sigmoid Node 14
    Inputs    Weights
    Threshold    0.5357986301845267
    Attrib HuCaracteristica1    -1.4671925945018958
    Attrib HuCaracteristica2    2.445381961214256
    Attrib HuCaracteristica3    -1.6961850214445466
    Attrib HuCaracteristica4    -1.873480049757511
    Attrib HuCaracteristica5    0.4419977020641842
    Attrib HuCaracteristica6    0.43096581004292855
    Attrib HuCaracteristica7    -0.30374463565763565
    Attrib Area    -2.8213233641682542
    Attrib ProporcionArea    -2.910082792357999
    Attrib proporcionPerimetro    4.01036141278459
    Attrib Proporcion    1.35073318311898
    Attrib ProporcionMedia    -1.1302412723531137
Sigmoid Node 15
    Inputs    Weights
    Threshold    -1.7716673645968155
    Attrib HuCaracteristica1    0.6019045580573533
    Attrib HuCaracteristica2    4.028312874739063
    Attrib HuCaracteristica3    -1.5088715003527884
    Attrib HuCaracteristica4    -0.28230226830575517
    Attrib HuCaracteristica5    0.4839096475669734
    Attrib HuCaracteristica6    -0.15271243289972813
    Attrib HuCaracteristica7    -0.519285480729619
    Attrib Area    -1.8115975593682452
    Attrib ProporcionArea    -0.4641668149117412
    Attrib proporcionPerimetro    -3.441834106636631
    Attrib Proporcion    -2.567547167383217
    Attrib ProporcionMedia    -1.1528251985506435
Sigmoid Node 16
    Inputs    Weights
    Threshold    0.498778340715623
    Attrib HuCaracteristica1    -4.005868009763355
    Attrib HuCaracteristica2    -1.242624136409327
    Attrib HuCaracteristica3    0.6671904487536481
    Attrib HuCaracteristica4    1.1614067224672864
    Attrib HuCaracteristica5    -0.4282658302236936
    Attrib HuCaracteristica6    -0.4072906637503845
    Attrib HuCaracteristica7    0.2352879437207532
    Attrib Area    1.041676667548363
    Attrib ProporcionArea    -3.5692253066276867
    Attrib proporcionPerimetro    0.6257012840008811
    Attrib Proporcion    2.4784039190590406
    Attrib ProporcionMedia    2.024946722226238
Sigmoid Node 17
    Inputs    Weights
    Threshold    -2.3904618841604948
    Attrib HuCaracteristica1    -4.28196680453346
    Attrib HuCaracteristica2    2.3835323308580607
    Attrib HuCaracteristica3    1.6230083808201559
    Attrib HuCaracteristica4    -0.180419829572400O5
    Attrib HuCaracteristica5    -0.725012356390744
    Attrib HuCaracteristica6    0.7152227815205292
    Attrib HuCaracteristica7    -0.3055202902067623
    Attrib Area    -1.6020316669449526
    Attrib ProporcionArea    -9.31175626949432
    Attrib proporcionPerimetro    0.6827276546444888
    Attrib Proporcion    0.8148596717617029
    Attrib ProporcionMedia    2.6705653358184844
Sigmoid Node 18
    Inputs    Weights
    Threshold    -4.4770359800204655
    Attrib HuCaracteristica1    -3.714385379820455
    Attrib HuCaracteristica2    3.8082952516617787
    Attrib HuCaracteristica3    -8.59452154115312
    Attrib HuCaracteristica4    -1.134679912366156
    Attrib HuCaracteristica5    -0.8813690160016656
    Attrib HuCaracteristica6    0.42704407933886673
    Attrib HuCaracteristica7    -0.8087448906544646
    Attrib Area    -2.3964549598628175
    Attrib ProporcionArea    -2.9988842560856024
    Attrib proporcionPerimetro    1.7430880686624737
    Attrib Proporcion    -4.277870744135296
    Attrib ProporcionMedia    -0.8196305803433072
Class 0
    Input
    Node 0
Class 1
    Input
    Node 1
Class 2
    Input
    Node 2
Class 3
    Input
    Node 3
Class 4
    Input
    Node 4
Class 5
    Input
    Node 5
Class 6
    Input
    Node 6
Class 7
    Input
    Node 7
Class 8
    Input
    Node 8
```

Ilustración 74 - Pesos Red de Neuronas 2

Árbol de decisión J48

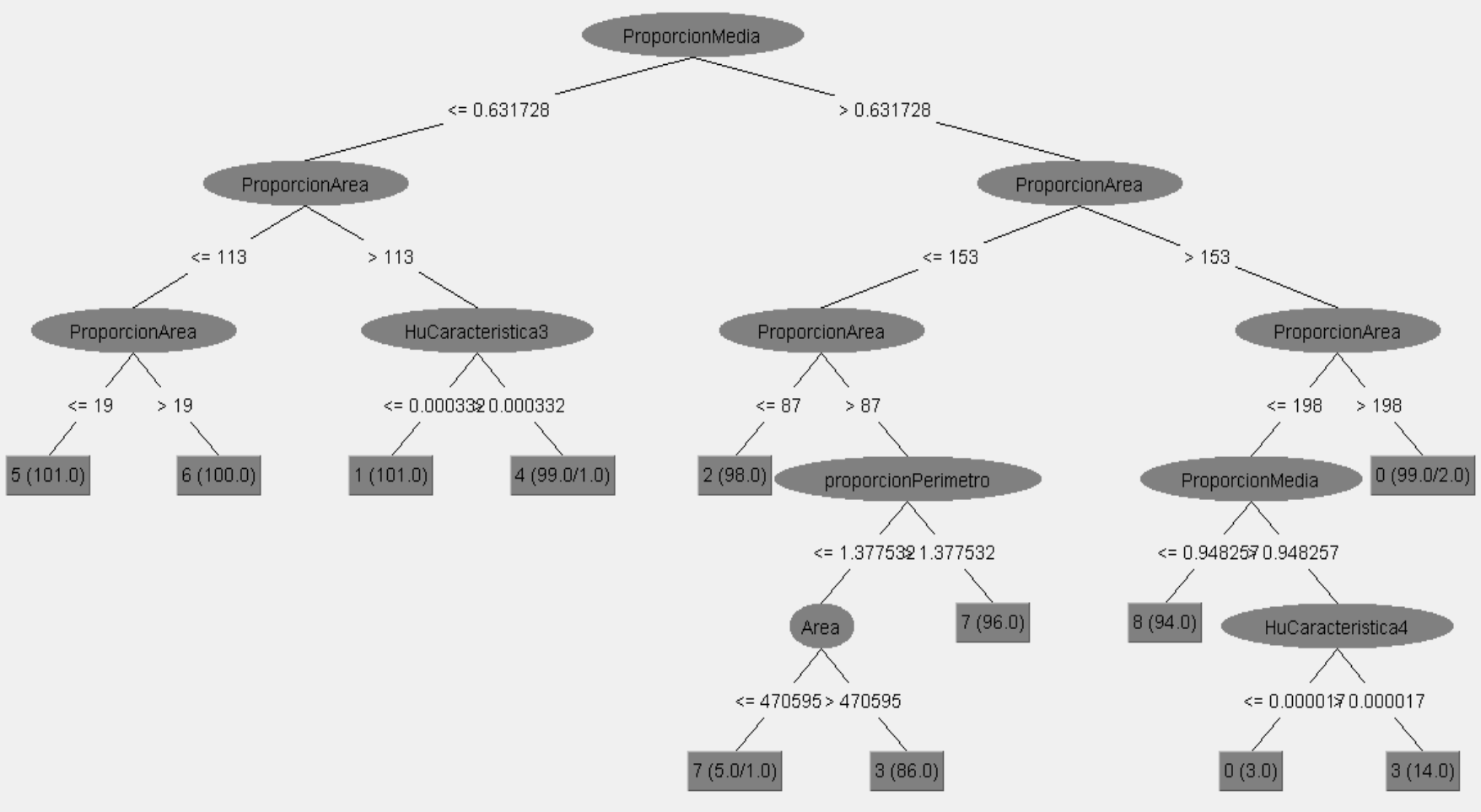


Ilustración 75 - Árbol de decisión - Realizado con WEKA

Anexo D: Imágenes en 3D realizado por WEKA

Imágenes de Weka de las clases en el plano 3D, se ha intentado capturar las imágenes desde distintas posiciones para que se pueda observar la agrupación de datos cuando se usan las características de Hu 1, 2 y 3. Se puede observar claramente la buena distribución de los datos con solo tres características.

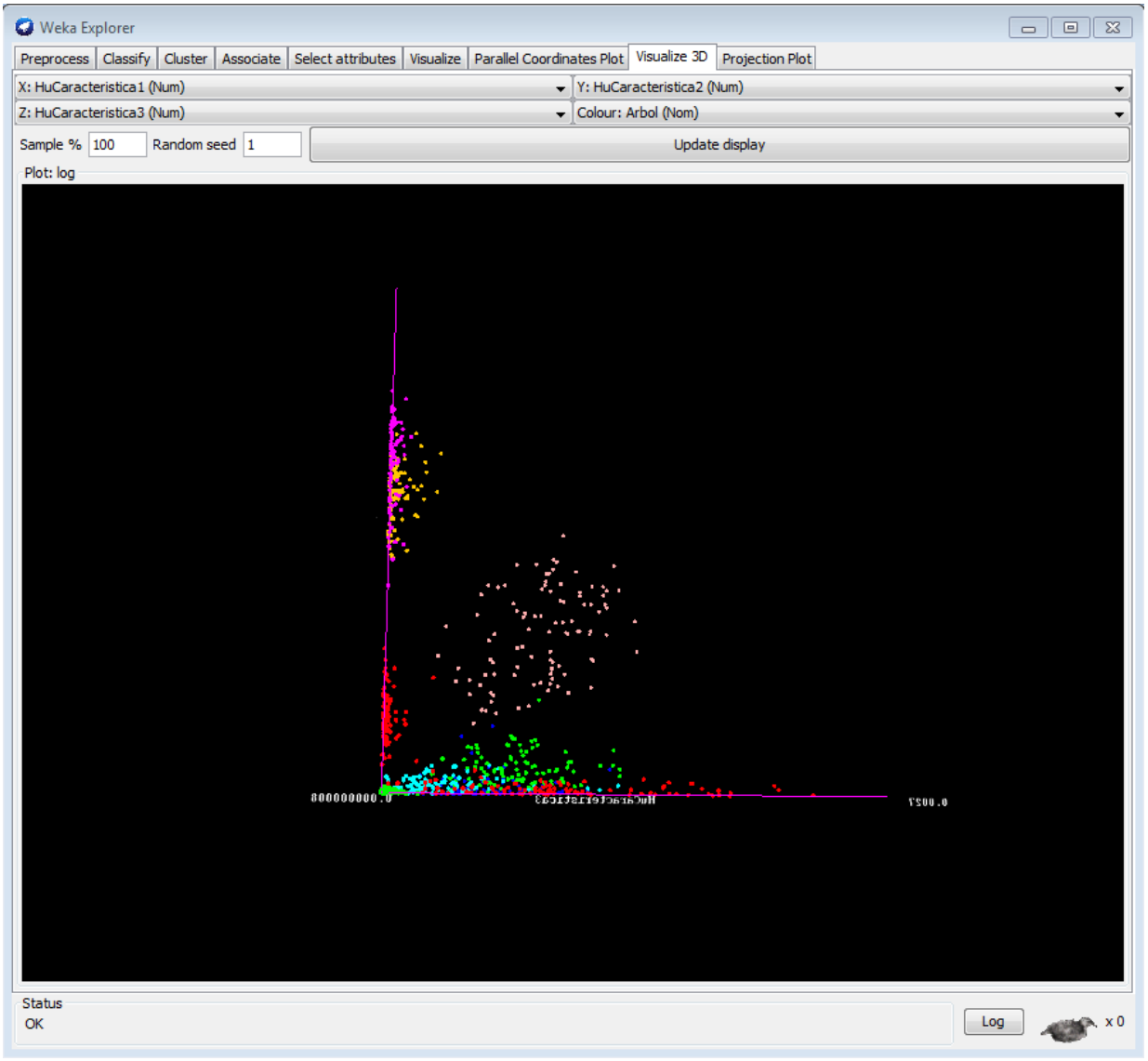


Ilustración 76 - Captura 1 de WEKA en 3D

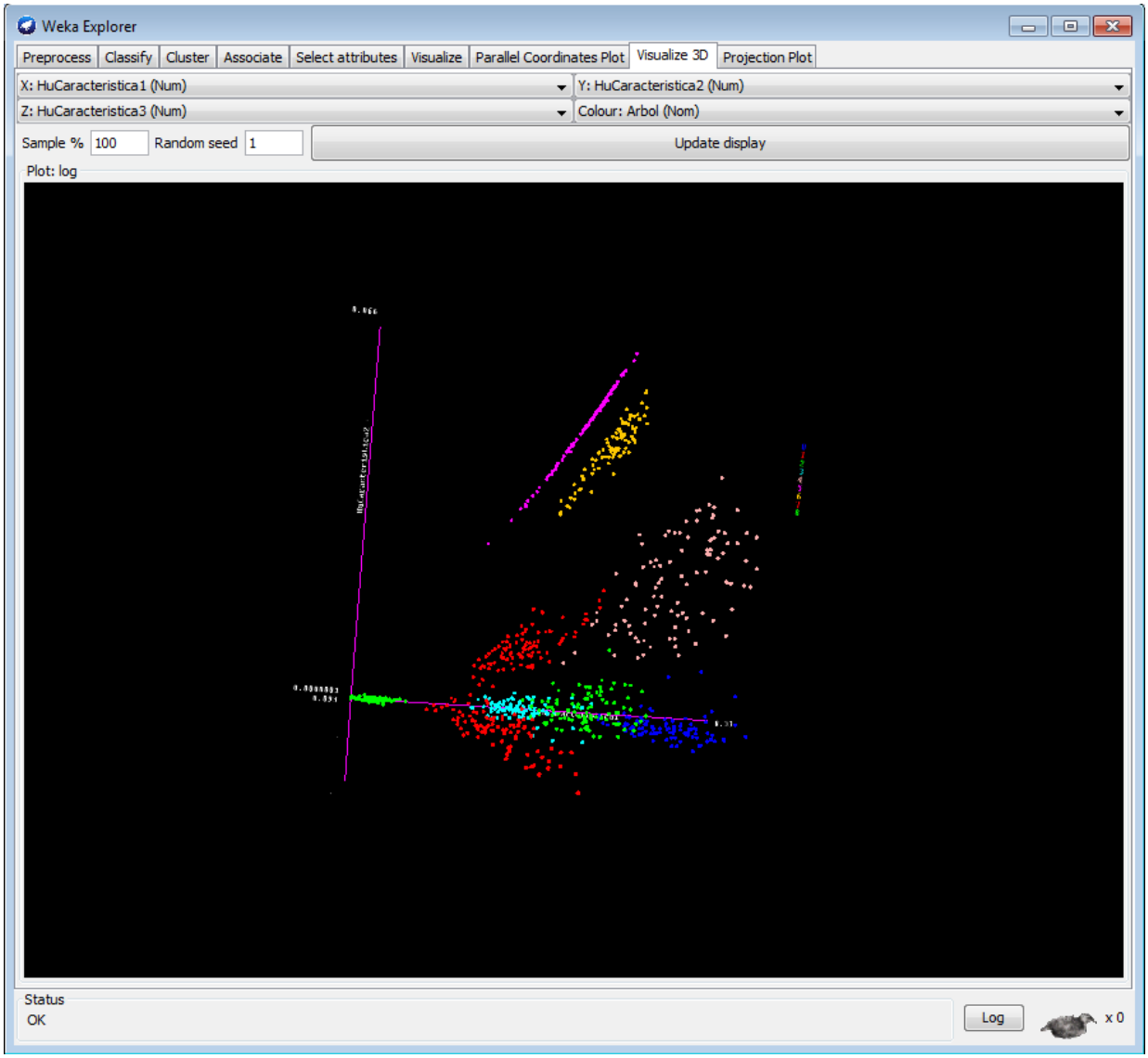


Ilustración 77 - Captura 2 de WEKA en 3D

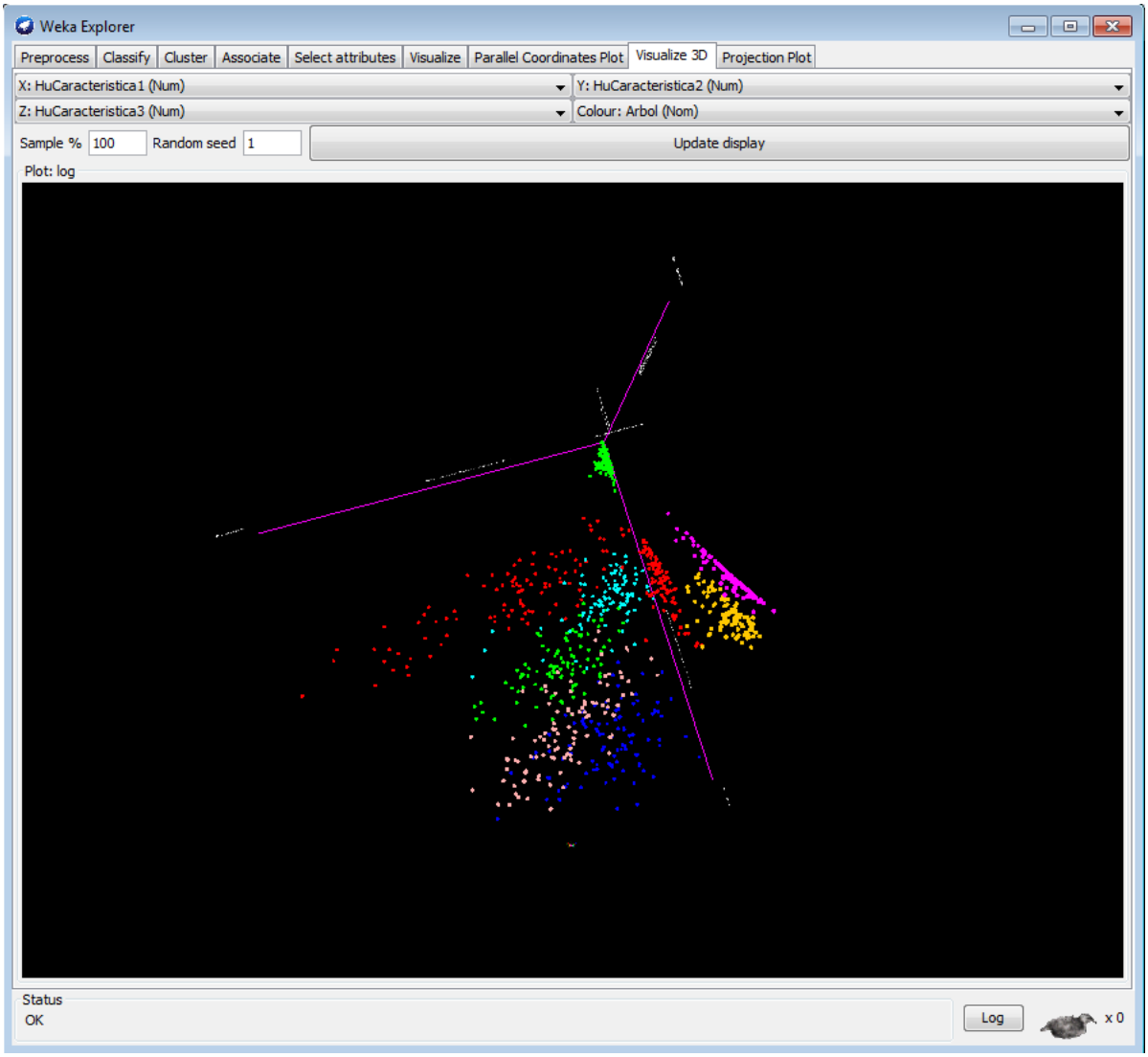


Ilustración 78- Captura 3 de WEKA en 3D