MÁSTER EN DATA SCIENCE

HERRAMIENTA PARA LA CONSTRUCCIÓN DE ÁRBOLES DE DECISIÓN CON AYUDA DE VISUALIZACIÓN DE DATOS

Autor: **Fausto Blasco Pisador**

Director: **Alberto Sánchez Campos**

Madrid, abril de 2020

Resumen

La idea inicial del presente Trabajo Fin de Máster (TFM) surge del trabajo descrito en el artículo de investigación 'Visually guided classification trees for analyzing chronic patients' [1].

El objetivo de la herramienta desarrollada es poder crear manualmente y de manera visual un árbol de decisión [2] para un conjunto de datos de *n* dimensiones. La visualización en la que se apoya el usuario para ir construyendo los diferentes nodos del árbol de decisión se basa en la combinación del uso de Star Coordinates (SC) [4] y Linear Discriminant Analysis (LDA) [3].

La herramienta presenta, en un entorno web, una interfaz para el manejo y carga de datos, visualización de los datos en dos dimensiones usando SC y LDA, creación y visualización del árbol de decisión y, finalmente, comparación de los principales parámetros del resultado de la clasificación efectuada por el árbol creado manualmente y por otro árbol creado automáticamente con la clase **`DecisionTreeClassifier`** [6] del módulo **`tree`** [7] de la librería **`Sklearn`** [5] de **`Python`**. Además, la herramienta permite exportar, en formato PDF, una representación gráfica tanto del árbol generado manualmente en la herramienta como del árbol generado automáticamente por la librería del paquete **`Sklearn`**.

La herramienta se ha desarrollado por completo usando **`Dash`** [8]. Escrito sobre **`Flask`** [9], **`Plotly.js`** [10] y **`React.js`** [11], **`Dash`** es ideal para crear aplicaciones de visualización de datos con interfaces de usuario altamente personalizadas en **`Python`** puro.

En esta memoria se describen los detalles de la herramienta desarrollada, así como el proceso de generación y comparación de un árbol de decisión para el análisis de un conjunto de datos recopilados durante 2012 por el Hospital Universitario de Fuenlabrada (HUF) sobre pacientes sanos y con enfermedades crónicas.

The idea for the current Master Thesis comes from the work described in the investigation paper 'Visually guided classification trees for analyzing chronic patients' [1].

The goal of the designed tool is to manually generate, in a visually guided manner, a decision tree [2] for an n-dimensioned dataset. The user will build the different tree nodes leveraging on a data visualization based on the combination of Star Coordinates (SC) [4] and Linear Discriminant Analysis (LDA) [3] techniques.

The tool has a web interface to process and load data, to visualize it in two dimensions using SC and LDA, to create and visualize a decision tree and, finally, to benchmark some of the main parameters of the classification results executed by the decision tree manually built by the user and another one automatically generated by the **`DecisionTreeClassifier`** [6] class of the **`tree`** [7] module from the **`Sklearn`** [5] **`Python`** library. Besides, the tool has an export module that allows the user to have a PDF-based graphical representation of the two decision trees.

The tool has been entirely designed using **`Dash`** [8]. Written over **`Flask`** [9], **`Plotly.js`** [10] and **`React.js`** [11], **`Dash`** is suitable to create data visualization dashboards with highly customized user interfaces using just **`Python`**.

This Master Thesis describes all the details about the designed tool as well as the generation and comparison process of a decision tree for the analysis of a dataset collected during 2012 by Hospital Universitario de Fuenlabrada (HUF) about two types of patients: healthy and patients with a chronic disease.

Índice

Índice de Figuras

Parte I:

Introducción, descripción del objetivo y solución propuesta

Capítulo 1.

1 Introducción

Los árboles de decisión [2] son un instrumento del campo del aprendizaje máquina ampliamente utilizado debido principalmente a dos motivos: son rápidos de construir y son fácilmente interpretables.

Durante su construcción, el algoritmo selecciona la variable y su valor que se utiliza en cada paso para particionar el espacio. Esta decisión se toma en base a una métrica de manera que se busca la variable y el valor que mejor particionan el espacio de acuerdo con esa métrica. Las dos métricas más utilizadas son la impureza de Gini y la ganancia de información [14].

El algoritmo devuelve el árbol construido en base a las observaciones usadas para su entrenamiento. Ese árbol puede usarse, a partir de ese momento, para clasificar o predecir el valor de la variable objetivo (según se trate de un árbol de clasificación o de regresión) de observaciones nuevas.

El árbol se puede dejar crecer hasta que el algoritmo finalice su cómputo de manera que puede contener muchos nodos, aunque estos no añadan mucho poder de clasificación. Por otro lado, existen métodos para disminuir el tamaño del árbol mediante técnicas de podado. En cualquiera de los casos, estos métodos para dejar crecer el árbol o para reducir su tamaño producen resultados que el usuario del árbol tendrá que interpretar o validar de acuerdo con su criterio del campo de conocimiento en el que se utilice el árbol de decisión.

1.1 Objetivos

El principal objetivo de este proyecto es diseñar una herramienta que permita al usuario construir un árbol de decisión en base a su conocimiento del dominio de trabajo de manera interactiva y guiada visualmente con la ayuda de representaciones gráficas del espacio de las variables en cada uno de los nodos del árbol.

Empezando por el nodo raíz del árbol, que representa el conjunto completo de los datos que se usarán para construir el modelo, el usuario podrá ir particionando el espacio de las variables en subespacios disjuntos. En cada paso, se elegirá la variable utilizada para la partición. El valor de dicha variable que particionará el espacio en dos subespacios disjuntos se obtendrá automáticamente por la herramienta en base al algoritmo C4.5 [12].

Cada nodo del árbol se podrá seleccionar de manera que el subconjunto de datos que representa dicho nodo se visualizará a su lado. Es obvio que la visualización elegida debe permitir representar los datos del espacio inicial n-dimensional en el espacio bidimensional del que disponemos en la interfaz.

Hay numerosos métodos de visualización de datos multidimensionales. Algunos, como Principal Component Analysis (PCA) o el Linear Discriminant Analysis (LDA) son métodos de proyección lineales usados normalmente para la reducción de la dimensionalidad de un

problema. Otros métodos, sin embargo, no tratan de reducir la dimensionalidad de los datos para representarlos, sino que tratan de trasladar la dimensionalidad inicial del problema a una representación en la que se reflejen de alguna manera todas las variables de este. Es el caso de métodos como coordenadas paralelas, coordenadas estelares o los gráficos de radar. En todos ellos se pueden comparar las diferentes variables y se pueden observar las relaciones entre ellas.

En el caso de esta herramienta y para visualizar el conjunto de datos se ha elegido un método de reducción de la dimensionalidad de manera que, el espacio n-dimensional de las variables del subconjunto de datos, se represente en un espacio bidimensional o unidimensional. Dicha representación se construye en base a la combinación de la técnica de representación de SC y del método de reducción de dimensionalidad LDA.

LDA nos permite proyectar las observaciones, desde su espacio n-dimensional, en un espacio bidimensional (o unidimensional en algún caso, como ya explicaremos en el capítulo 4.2) de manera que en dicho espacio se maximiza la separación entre clases (también llamada varianza 'entre-clase') y se minimiza la dispersión de las observaciones dentro de una clase (también llamada varianza 'intra-clase').

Por otro lado, SC es una técnica de visualización de datos multidimensionales que genera proyecciones lineales de los datos desde un espacio n-dimensional a un espacio bidimensional. En particular, SC construye los puntos que representan a cada observación haciendo uso de una serie de vectores como ejes donde cada uno de esos vectores está relacionado con cada una de las variables de los datos. La orientación de cada uno de esos ejes determina la dirección en la que crece la variable asociada y su longitud la contribución de dicha variable en la representación final.

SC permite generar un número arbitrario de representaciones para un mismo conjunto de datos modificando los ejes. Esto también puede usarse para hacer coincidir la representación de un conjunto de datos usando SC con la que generaría dicho conjunto de datos en un espacio bidimensional usando cualquier método de proyección lineal como LDA.

Haciendo uso de la combinación de SC y LDA el usuario de la herramienta podrá decidir en base a su conocimiento del campo de trabajo, pero también de la información obtenida en la representación gráfica asociada, qué variable usar para particionar el espacio de datos en cada nodo y construir así, paso a paso y de manera gráfica, el árbol de decisión que mejor conjugue ambos conceptos.

1.2 Organización del proyecto

Este documento se divide en 6 capítulos, en los cuales se explica la solución propuesta y la herramienta desarrollada para llevar a cabo el objetivo del proyecto.

En el primer capítulo se hace una breve introducción y se describen los objetivos del proyecto.

El segundo capítulo se dedica a profundizar en los principales conceptos teóricos en los que se apoya la idea en la que se sustenta el proyecto. Se detallan sus principales características y se sientan las bases para comprender el por qué y el cómo de este proyecto.

El tercer capítulo se dedica a describir la motivación de la herramienta desarrollada y la tecnología en la que se basa.

En el cuatro capítulo se describe en detalle la herramienta, su interfaz, el proceso de construcción de un árbol y la información que la herramienta entrega sobre el resultado del proceso.

En el quinto capítulo se describen los pasos y los resultados del uso de la herramienta y el método propuesto en este proyecto al ser aplicados a un caso real en el campo de la medicina.

Por último, en el sexto capítulo se reflexiona sobre todo lo expuesto hasta el momento, obteniendo algunas conclusiones al respecto y proponiendo mejoras y futuros desarrollos de la herramienta diseñada.

Capítulo 2

2 Conceptos principales

2.1 Árboles de decisión

Los árboles de decisión son un tipo de algoritmo supervisado de aprendizaje máquina que se puede usar tanto en problemas de regresión como de clasificación. En el caso de usarse para problemas de clasificación se denominan árboles de clasificación. En tal caso, la variable objetivo es una variable de tipo categórica.

Los árboles de decisión son un algoritmo usado ampliamente por su relativa rapidez de construcción además de que es fácilmente interpretable (si no son demasiado grandes).

Los árboles de decisión particionan el espacio de las variables, de manera binaria y recursiva, en una serie de espacios lineales disjuntos y en cada uno de ellos aplican un modelo simple, como asignar un valor constante a la variable objetivo.

La partición del espacio de las variables se realiza de una manera jerárquica y esa partición se representa gráficamente como una colección de nodos conectados a través de ramas también de una manera jerárquica. Cada nodo está asociado con un subconjunto o región del espacio de datos original. Cada rama representa una frontera, dada por un valor concreto de la variable usada para particionar el nodo desde el cual parte dicha rama, que divide el espacio de los datos de ese nodo en dos subconjuntos disjuntos dando lugar a otros dos nuevos nodos.

En la Figura 1 puede verse un ejemplo de partición de un espacio bidimensional y su representación en forma de árbol.

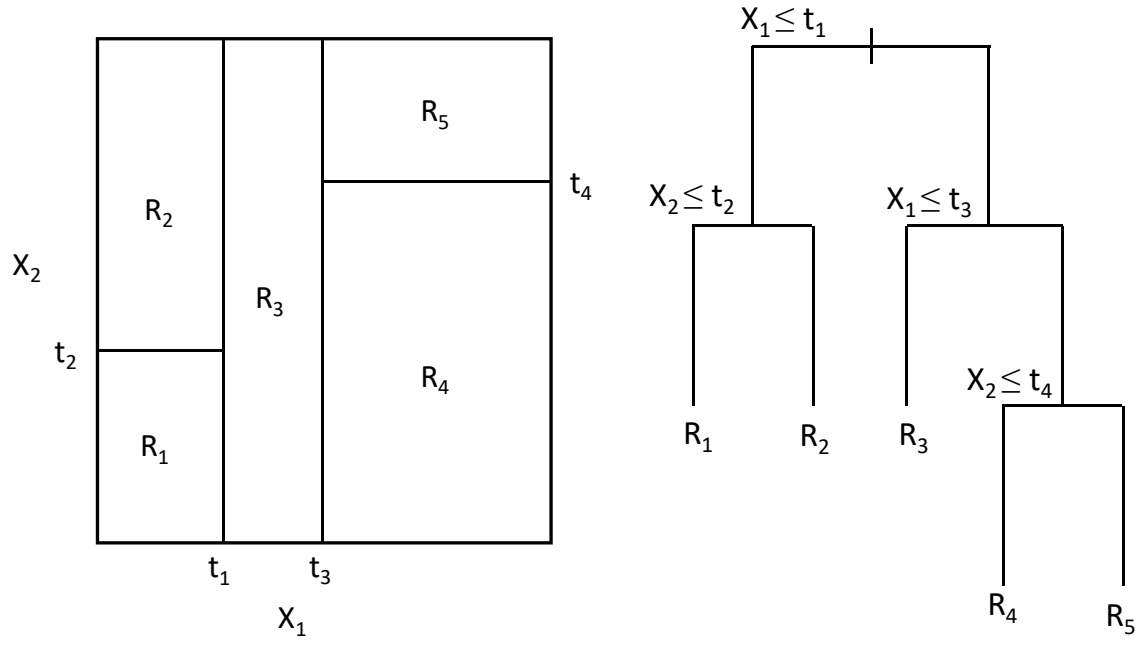


Figura 1: Partición de un espacio de variables bidimensional en regiones disjuntas y su correspondiente representación en forma de árbol

El criterio por el cual se particiona el espacio en base a una variable se puede basar en diferentes funciones y métricas. En este proyecto nos hemos fijado en la función de ganancia de información, basada en la métrica de la entropía.

La **entropía** [13] (según la definición del concepto en la teoría de información) de una variable aleatoria se interpreta como el nivel medio de "información", "sorpresa" o "incertidumbre" inherente en los posibles valores que pueda tomar dicha variable.

Para una variable aleatoria discreta, *X*, la información de un valor $X=x$ es

(2.1) $$i(x) = -\log\big(p(x)\big)$$

La entropía de una variable aleatoria discreta se define por

(2.2) $$H(X) = -E[\log p(X)] = -\sum_{x\in X} p(x)\ \log\big(p(x)\big)$$

Por lo que la entropía se puede definir como información media de la variable aleatoria *X*

(2.3) $$H(X) = E[i(X)]$$

La entropía mide por tanto y de cierta manera la cantidad de "certidumbre" acerca del valor de una variable. Si una variable puede tener dos valores con igual probabilidad, la entropía es máxima, ya que no tenemos información acerca del valor que la variable puede tomar. Sin embargo, si la probabilidad de uno de los valores es mucho mayor que la del contrario, la entropía decrece hacia su mínimo ya que tenemos información acerca del valor que la variable puede tomar.

En el caso extremo, cuando una variable sólo puede tomar un único valor, la entropía es cero, ya que la "certidumbre" que tenemos del valor que tomará dicha variable es total.

La **ganancia de información** [14], por su parte, mide la reducción de la entropía de una variable aleatoria al particionar un conjunto de datos por un valor concreto de otra de sus variables aleatorias. La ganancia de información mide por lo tanto la información mutua de dos variables aleatorias *X* e *Y*, es decir, la reducción de la entropía de *X* conseguido al conocer el estado de la variable aleatoria *Y*.

Si tenemos por tanto un conjunto de datos T, la ganancia de información de T con respecto a un atributo α es la diferencia entre la entropía $H(\mathrm{T})$ del conjunto de datos y la entropía condicional $H(\mathrm{T}|\alpha)$

(2.4) $$IG(\mathrm{T},\alpha) = \mathrm{H}(\mathrm{T}) - \mathrm{H}(\mathrm{T}|\alpha)$$

Donde $H(\mathrm{T}|\alpha)$ es la entropía condicional de T dado un atributo α

(2.5) $$\mathrm{H}(\mathrm{T}|\alpha) = \sum_{v\in vals(\alpha)} \frac{|S_\alpha(v)|}{|\mathrm{T}|} \cdot H\big(S_\alpha(v)\big)$$

Siendo $S_\alpha(v)$ el subconjunto de datos de T para los cuales el atributo α toma el valor v y, por lo tanto, $\frac{|S_\alpha(v)|}{|T|}$ es la ratio de las observaciones del conjunto de datos en las que la variable α toma el valor v con respecto al total de observaciones del conjunto de datos.

Combinando ambos conceptos (ganancia de información y entropía) tenemos que, a mayor ganancia de información al particionar un conjunto de datos por una variable, menor será la entropía de los subconjuntos resultantes de tal partición. Además, sabemos que, a menor entropía, menor es la "incertidumbre" en el valor de la variable aleatoria en ese subconjunto de datos.

Es decir, la ganancia de información es una manera de usar la entropía para calcular como un cambio en el conjunto de datos impacta en la pureza de este, esto es, en la distribución de las clases de la variable objetivo. Una menor entropía sugiere más pureza o menos "incertidumbre".

No perdamos de vista que el objetivo final de un árbol de clasificación es el de crear nodos en los cuales la mayoría de las observaciones pertenezcan a una única clase. De esta manera estaremos construyendo nodos con una mayor "certidumbre" con respecto a la clase que en ellos tomará la variable objetivo de la clasificación. Por lo tanto, al construir un árbol de clasificación, los nodos se particionan de manera que la ganancia de información sea máxima y, por lo tanto, la entropía resultante en los nodos resultado sea mínima, lo que nos llevará a una "certidumbre" máxima acerca de la clase de la variable objetivo en dicho nodo.

Todos estos conceptos se aplicarán en el diseño de los mecanismos de los que hace uso la herramienta desarrollada y veremos cómo se han construido algoritmos que los implementan.

2.2 Star Coordinates

SC es un sistema de representación de datos multidimensionales en dos dimensiones (plano). La motivación principal del sistema SC no es el de proveer medios para análisis numéricos con los datos representados, sino principalmente el de ganar información acerca de los datos en base a una representación gráfica de los mismos. Ya que la representación gráfica de datos multidimensionales es compleja, cuando no imposible en muchos casos, el hecho de poder obtener una representación bidimensional de los mismos puede ayudar a los usuarios de los datos en su interpretación en las primeras fases de un proyecto basado en datos, esto es, durante la fase de exploración de éstos.

La idea inicial y primera del sistema de SC es la siguiente:

- Se distribuyen los ejes del sistema dentro de un círculo en un plano bidimensional con (inicialmente) ángulos iguales entre ellos y con un origen común en el centro del círculo.

- Inicialmente todos los ejes tienen la misma longitud.

- Los datos se escalan a la longitud de los ejes, haciendo coincidir el mínimo valor con el origen de los ejes y el máximo con el extremo opuesto.

- Los vectores unitarios se calculan de acuerdo con ese escalado.

En la Figura 2 puede verse como se calcula la posición de un punto en el espacio bidimensional usando SC para un dato de un conjunto de datos de 8 dimensiones.

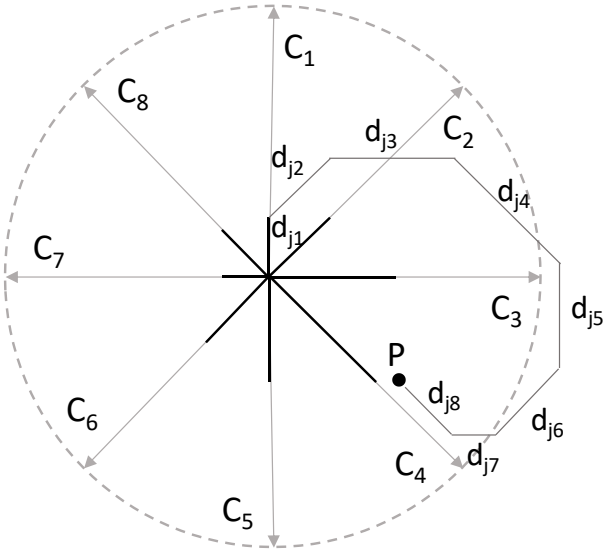


Figura 2: Posición de un punto para un conjunto de datos 8-dimensional

Los usuarios pueden aplicar ciertas transformaciones a la visualización. Es posible cambiar la longitud o la rotación de cada uno de los ejes. Con estas transformaciones el usuario puede modular la contribución de cada uno de los atributos del conjunto de datos en la visualización resultante (en el caso de cambiar la longitud de un eje) o la correlación de cada uno de ellos con el resto (en el caso de cambiar la rotación de un eje).

Ambas modificaciones, escalado y rotación, se pueden aplicar a múltiples ejes para examinar los efectos combinados de múltiples atributos de una sola vez. En la Figura 3 puede verse un ejemplo de posicionado de un punto de un conjunto de datos de 5 dimensiones en el plano bidimensional en el que los ejes se han escalado y rotado.

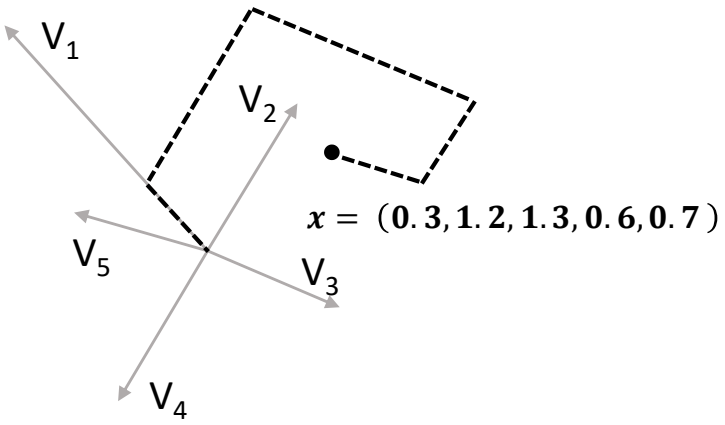


Figura 3: Posición de un punto para un conjunto de datos 5-dimensional con ejes escalados y rotados

2.3 Linear Discriminant Analysis

LDA es un método usado en estadística, reconocimiento de patrones y aprendizaje máquina, para encontrar la combinación lineal de las variables de un conjunto de datos de manera que separe al máximo dos o más clases. La combinación resultante puede usarse tanto para construir un clasificador lineal o como método de reducción de la dimensionalidad para exploración de datos o paso previo a la ejecución de otro algoritmo de clasificación.

Ronald A. Fisher formuló los discriminantes lineales en su trabajo "The Use of Multiple Measurements in Taxonomic Problems" [2] de 1936. Los discriminantes lineales se describieron inicialmente para problemas con dos únicas clases de la variable objetivo de la clasificación, pero pronto fue generalizado a problemas multi-clase.

La idea principal de LDA es muy similar a la del análisis de componentes principales (PCA por sus siglas en inglés) pero en vez de encontrar la combinación lineal de variables que producen una proyección de los datos en un espacio que maximiza la varianza de estos, LDA se centra en encontrar los ejes que maximizan la separación entre las múltiples clases. Puede observarse en la Figura 4 la diferencia entre ambas técnicas a la hora de encontrar los ejes sobre lo que proyectar los datos.

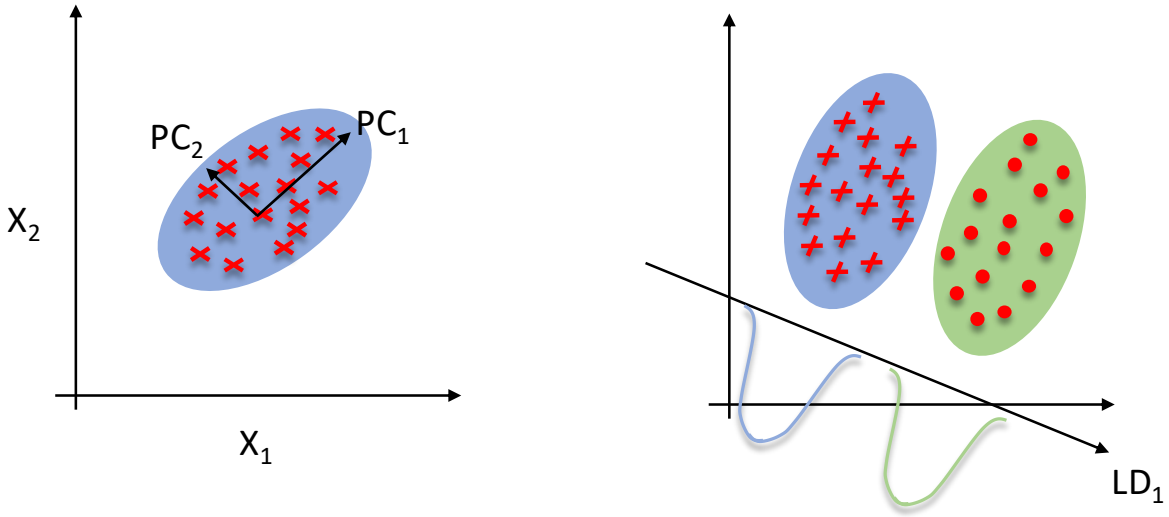


Figura 4: Diferencia entre PCA y LDA en la proyección de datos a un subespacio que maximiza la varianza o la separación entre clases

Es por ello por lo que, a diferencia de la técnica de PCA que es no supervisada, LDA es una técnica supervisada ya que toma en cuenta las clases de las observaciones de manera que encuentra los ejes en los cuales la proyección del conjunto de datos maximiza la separación entre las diferentes clases.

A alto nivel, los pasos que hay que dar para ejecutar LDA sobre un conjunto de datos con n dimensiones son:

1. Calcular los vectores n-dimensionales de la media de cada variable para los datos pertenecientes a cada una de las clases de la variable objetivo del conjunto.

2. Calcular las matrices de dispersión o varianza inter-clases ($\boldsymbol{S}_W$) e intra-clase ($\boldsymbol{S}_B$).

3. Calcular los autovectores $(\boldsymbol{e}_1, \boldsymbol{e}_2, \dots, \boldsymbol{e}_n)$ y sus correspondientes autovalores $(\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \dots, \boldsymbol{\lambda}_n)$ para la matriz $\boldsymbol{S}_W^{-1} \times \boldsymbol{S}_B$.

4. Ordenar los autovectores por orden descendiente de sus autovalores y elegir los m autovectores con mayor valor de sus autovectores (donde $m \leq n$). Formar una matriz $\boldsymbol{W}$ de dimensión $n \times m$ donde cada columna representa un autovector.

5. Transformar los datos del espacio n-dimensional inicial al espacio m-dimensional haciendo uso de la matriz $\boldsymbol{W}$ de dimensión $n \times m$ por medio de una multiplicación matricial $\boldsymbol{Y} = \boldsymbol{X} \times \boldsymbol{W}$ (donde $\boldsymbol{X}$ es una matriz de $p \times n$ dimensiones que representa las p observaciones e $\boldsymbol{y}$ son las observaciones transformadas de dimensión $p \times m$ en el nuevo subespacio).

Es importante señalar que, en el algoritmo de LDA, el número máximo de discriminantes lineales que se puede conseguir es el menor de las cantidades $N_c - 1$ o n. Donde N_c es el número de clases diferentes de la variable respuesta y n es el número de variables predictoras. Este hecho nos lleva a plantear una alternativa a la representación en dos dimensiones de los datos de un nodo cuando en éste únicamente hay observaciones de dos o una clase.

En el caso de dos clases, sólo podremos obtener un discriminante lineal, lo que nos llevará a tener una proyección unidimensional de los datos. En el caso de una única clase, no tiene sentido aplicar LDA ya que no tenemos clases que separar.

Ambos casos se han resuelto de una manera que ya explicaremos en el capítulo 4 cuando describamos en detalle la interfaz de la herramienta desarrollada.

Por último, es clave también destacar que LDA asume variables independientes, normalmente distribuidas y con idéntica matriz de covarianza para cada una de las clases. En cualquier caso, se ha comprobado que LDA es bastante robusto ante violaciones de esas asunciones.

2.4 Combinando SC y LDA

Como hemos comentado anteriormente, SC es un método de visualización de datos multidimensionales que genera proyecciones de un espacio n-dimensional a un espacio m-dimensional menor (normalmente $m = 2$) para poder representar datos gráficamente.

La proyección queda definida por un conjunto de n vectores m-dimensionales $\boldsymbol{v_i}$ con un origen común, donde $\boldsymbol{v_i}$ está asociado con el atributo i-ésimo de los datos. La proyección $\boldsymbol{p} \in \mathbb{R}^m$ de una muestra $\boldsymbol{x} \in \mathbb{R}^n$ de los datos viene dada por la combinación lineal de los vectores $\boldsymbol{v_i}$, siendo los coeficientes los valores de los atributos de $\boldsymbol{x}$. Es decir,

$$\boldsymbol{p} = x_1 \boldsymbol{v_1} + x_2 \boldsymbol{v_2} + \cdots + x_n \boldsymbol{v_n} = \boldsymbol{V}^{\mathrm{T}} \boldsymbol{x} \tag{2.6}$$

Donde $\boldsymbol{V}$ es la matriz de dimensiones $n \times m$ cuyas filas son los vectores $\boldsymbol{v_i}$.

Por otro lado, los distintos métodos lineales de reducción de la dimensionalidad (PCA, LDA, …) tienen una matriz asociada que transforma cada muestra original $\boldsymbol{x}$ en su proyección $\boldsymbol{p}$ del espacio de llegada. Es decir,

(2.7) $$\boldsymbol{p} = \boldsymbol{A}\,\boldsymbol{x}$$

Se puede construir un modelo de SC que genere un gráfico en el que la representación de cada muestra original $\boldsymbol{x}$ venga dada por la proyección (2.7) de un modelo lineal (LDA en nuestro caso) y, por tanto, como consecuencia de (2.6), $\boldsymbol{V} = \boldsymbol{A}^{\mathrm{T}}$.

Si recordamos que $\boldsymbol{A}$ es la matriz que tiene en sus filas los autovectores que se obtienen de resolver el problema de autovectores que plantea el algoritmo LDA (ver capítulo 2.3), los ejes de un gráfico SC ($\boldsymbol{V}$) que haga coincidir su proyección de los datos con la que genera el algoritmo LDA se obtienen como $\boldsymbol{V} = \boldsymbol{A}^{\mathrm{T}}$.

La combinación de ambas técnicas puede ayudar al usuario en el proceso de selección de variables previo a la construcción de un clasificador. Se puede simplificar la visualización descartando variables que no tienen mucha influencia en la separación de las clases. En SC los ejes de menor longitud son candidatos para ser eliminados, ya que el efecto de cada variable en la proyección depende de la longitud de este [15]. En el caso, por tanto, de haber obtenido los ejes de la proyección SC a partir de la proyección de un método como LDA, se pueden eliminar las variables que corresponden a los ejes más cortos ya que estos tienen poco efecto en la separación de las clases.

Lo mismo ocurre con la dirección de los ejes, que hay que tener también en cuenta ya que, cuando hay cierto solapamiento de las clases, los ejes que se encuentran en la dirección que mejor separa las clases, aun siendo cortos, son relevantes [15].

Capítulo 3

3 Solución propuesta

3.1 Planteamiento de la solución

Inicialmente este trabajo iba a consistir en añadir un módulo a una herramienta ya existente. La herramienta viene descrita en el artículo de investigación "A Visual Interface for Feature Subset Selection Using Machine Learning Methods" [16] del cuál es coautor el director de este TFM.

El objetivo de la herramienta descrita en dicho artículo es proporcionar una interfaz visual e interactiva para permitir la selección de atributos sobre conjuntos de datos multidimensionales. Con el fin de que la visualización sea efectiva y aporte el mayor conocimiento posible para su exploración, la interfaz guía a los analistas en la selección de un conjunto de atributos significativo. Una vez indicados los atributos con los que trabajar, la herramienta permite la elección de métodos automáticos de reducción de dimensionalidad lineales y no lineales. Tras el preprocesamiento de los datos, éstos se utilizan para crear un modelo de predicción basado en algún algoritmo de clasificación supervisada.

Sobre dicha herramienta, como primer paso de este TFM, se diseñó un módulo nuevo que permitía construir y visualizar un árbol de clasificación para el conjunto de datos de entrada. Una vez dado ese paso, y para poder trabajar en la construcción de una herramienta desde cero y que fuera independiente de lo ya existente, se construyó la herramienta que se describe en este trabajo.

Las principales diferencias, a alto nivel, entre la herramienta original en la que incluyó el módulo de árbol de clasificación y la herramienta final que se presenta en este TFM son:

1. La herramienta original se enfoca sobre todo en la visualización de los datos, incluyendo varios métodos de reducción de la dimensionalidad (tanto lineales como no lineales), así como información varia sobre estadística descriptiva del conjunto de datos usado.

2. La herramienta presentada en este TFM se centra en la construcción del árbol de clasificación y utiliza únicamente LDA como método de visualización de los datos para la ayuda a la construcción de este.

3. La herramienta original presenta además funcionalidad para clustering que la herramienta presentada en este TFM no tiene.

4. La herramienta presentada en este TFM extiende la funcionalidad de construcción del árbol de clasificación con respecto a la funcionalidad que se ha diseñado para la herramienta original en los siguientes aspectos:

a. Una vez construido el árbol, la herramienta construye un clasificador basado en dicho árbol y ofrece la posibilidad de ejecutar una clasificación usando dicho clasificador.

b. También ejecuta una clasificación usando un árbol construido automáticamente por la librería **Sklearn** de **Python**.

c. Se presenta además una comparación de las principales métricas de bondad de ambos árboles para poder comparar ambos clasificadores.

d. Finalmente, es posible exportar, en formato PDF, ambos árboles.

3.2 Tecnología usada

Para el desarrollo de la herramienta se ha utilizado el paquete **Dash** para **Python**. Este paquete permite la creación de aplicaciones de analíticas de datos reactivas basadas en una interfaz web. No requiere de programación en **JavaScript** [17] ni otras tecnologías asociadas al diseño web.

Dash nace de la empresa que creó el paquete de visualización **Plotly**. **Plotly** fue construido usando **Python** y el marco de referencia de **Django**. **Plotly** tiene librerías con interfaz de programación de aplicaciones (API por sus siglas en inglés) para **Python**, **R**, **MATLAB**, **Node.js**, **Julia** y **Arduino**.

Dash por su parte es un producto de código abierto con API para **Python** y **R**. Las aplicaciones generadas por este paquete se representan en un navegador web. Se puede desarrollar una aplicación y compartirla vía un localizador uniforme de recursos (URL por sus siglas en inglés). Como la aplicación se visualiza en un navegador web es inherentemente multiplataforma y preparada para ser visualizada en dispositivos móviles.

Las aplicaciones **Dash** son en realidad un servidor web que ejecuta **Flask** y cuya interfaz es representada usando **React.js**, la librería de **JavaScript** para interfaz de usuario escrita y mantenida por Facebook. **Dash** se apoya en toda la funcionalidad que ofrecen **Flask** y **React.js** para ponerlas al alcance de científicos de datos que programen en **Python** pero que no tengan conocimientos en programación web.

Capítulo 4

4 Descripción de la aplicación desarrollada

4.1 Partes principales de la interfaz

Como hemos comentado anteriormente, la herramienta presenta una interfaz web en la que se pueden realizar todas las tareas de visualización y construcción del árbol de clasificación. Dicha interfaz consta de 4 partes principales como puede observase en la Figura 5.

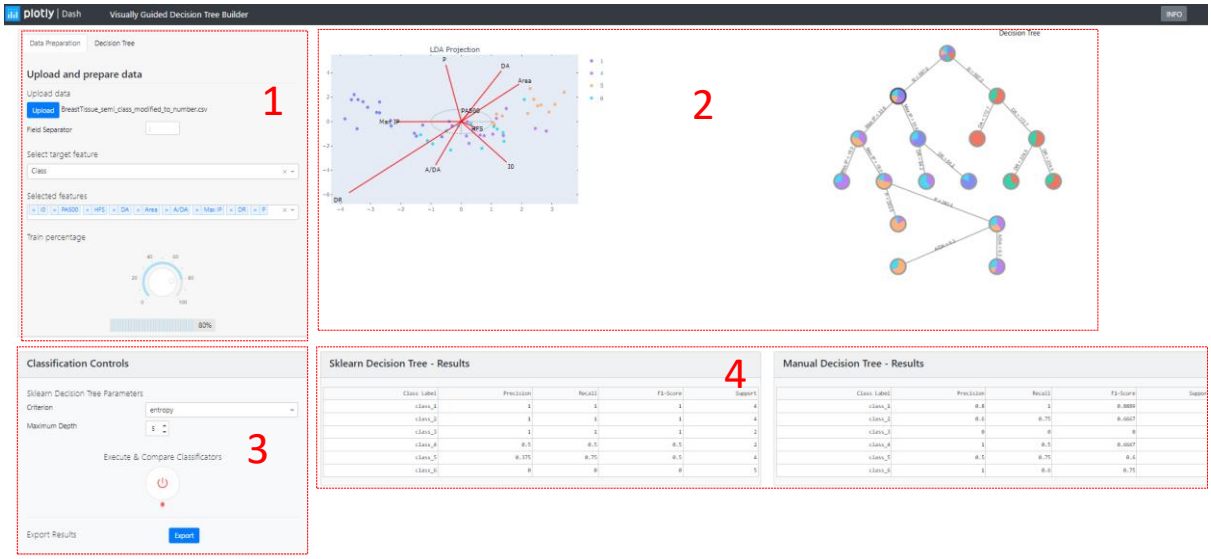


Figura 5: Interfaz principal propuesta

Estas 4 zonas principales son:

1. Controles de carga de datos y construcción el árbol
2. Zona de visualización
3. Controles de los clasificadores y exportado
4. Tablas comparativas de los clasificadores

En la primera zona, controles de carga de datos y construcción del árbol, existen dos pestañas. La primera se destina a los controles de carga de datos y la segunda a los controles para la construcción del árbol. En la Figura 6 pueden verse en detalle los elementos de cada una de ellas.

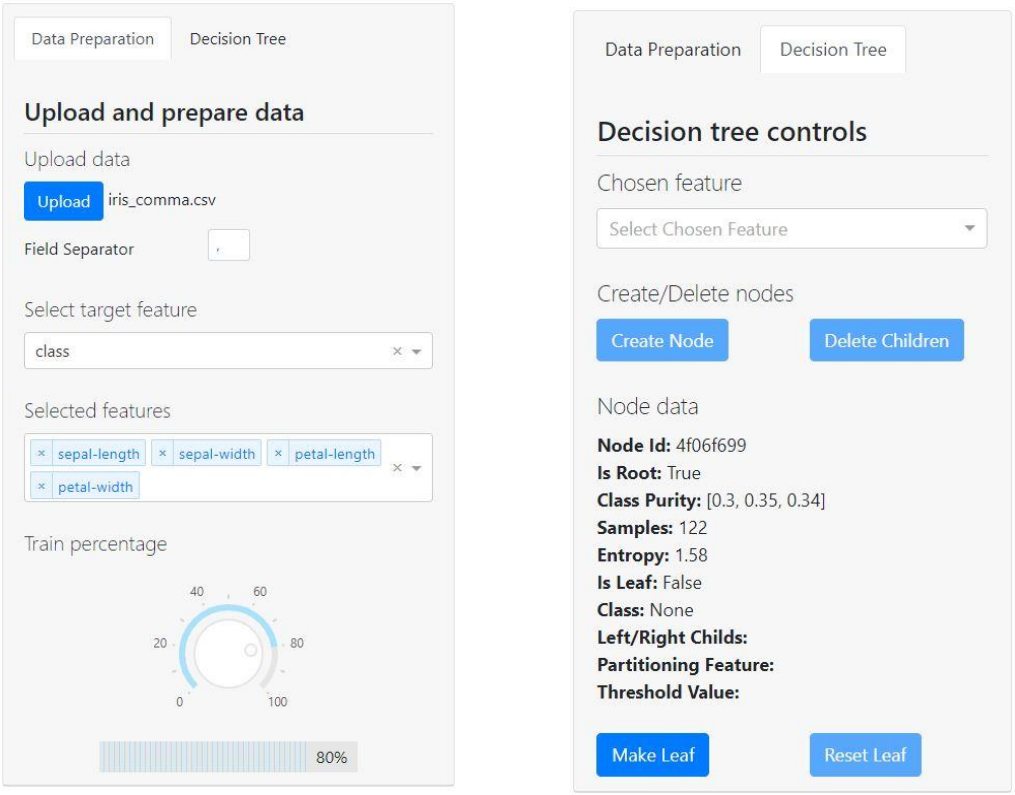


Figura 6: Controles de carga y preparación de datos y de construcción del árbol respectivamente

En la parte de carga y preparación de datos tenemos los siguientes elementos:

- Botón de carga de datos: permite seleccionar el archivo en formato de campos separados por comas (CSV por sus siglas en inglés) que contiene los datos con los que vamos a trabajar.

- Campo de separador de campos: permite configurar el tipo de separador de campos con el que se ha construido el CSV.

- Selector de variable objetivo: elemento desplegable que muestra todas las variables del conjunto de datos para seleccionar cuál es la que contiene la clase objetivo del clasificador que queremos construir.

- Selector de variables múltiple: elemento desplegable que permite seleccionar las variables que queremos usar como variables predictoras en nuestro clasificador.

- Regulador de porcentaje: permite seleccionar el porcentaje de los datos que se usarán para entrenar a nuestro clasificador y, por lo tanto, el porcentaje de datos de prueba que se usarán para probar la eficacia del clasificador construido. El porcentaje seleccionado se muestra también en una barra de progreso indicando el valor exacto seleccionado.

Por otro lado, en la pestaña de control del árbol de clasificación, contamos con los siguientes elementos:

- Selector de variable para particionar: elemento desplegable que muestra todas las variables predictoras para poder elegir cuál de ellas usar para particionar un nodo del árbol de decisión. Este elemento sólo se activa al seleccionar un nodo que aún no se ha particionado. En el caso de estar seleccionado un nodo ya particionado, el selector no se activa, pero muestra la variable por la cual se particionó el nodo seleccionado.

- Botones de creación/borrado de nodos: permiten crear dos nuevos nodos a partir de uno seleccionado en base a una variable predictora seleccionada. Por otro lado, si el nodo seleccionado ya ha sido particionado, el botón de borrado permite deshacer la parte del árbol de decisión que cuelga de él.

- Campo de texto: campo que contiene información relevante sobre el nodo seleccionado.

- Botones de forzar/deshacer hoja: permiten forzar un nodo a ser de tipo hoja. Los nodos tipo hoja son los que asignan la clase en el proceso de clasificación y, por lo tanto, no pueden tener hijos. Un nodo puede ser hoja si sólo contiene observaciones de una única clase o si el usuario de la herramienta decide que no quiere particionarlo más y quiere que se asigne la clase mayoritaria a las observaciones que caigan en ese nodo durante el proceso de clasificación.

En la segunda zona de la interfaz de la herramienta es dónde se muestran las visualizaciones. Por un lado, se muestra la proyección de los datos en el espacio bidimensional que produce LDA en combinación con los ejes SC que resultan de hacer coincidir ambas proyecciones, tal y como se ha explicado en el capítulo 2.4.

Como puede verse en la Figura 7 las proyecciones de las observaciones pertenecientes a cada clase se representan en un color diferente para poder visualizar de mejor manera la distribución en el espacio proyectado de las diferentes clases. Se ofrece también información sobre cada dato del gráfico al pasar el cursor por encima.

Una funcionalidad muy interesante que se obtiene por el hecho de desarrollar la herramienta en **`Dash`** es que los gráficos, que son objetos de **`Plotly`**, tienen por defecto todas las opciones de interactividad y reactividad que ofrece este paquete como son: funciones de zoom, encuadre, filtrado o selección de datos etc.

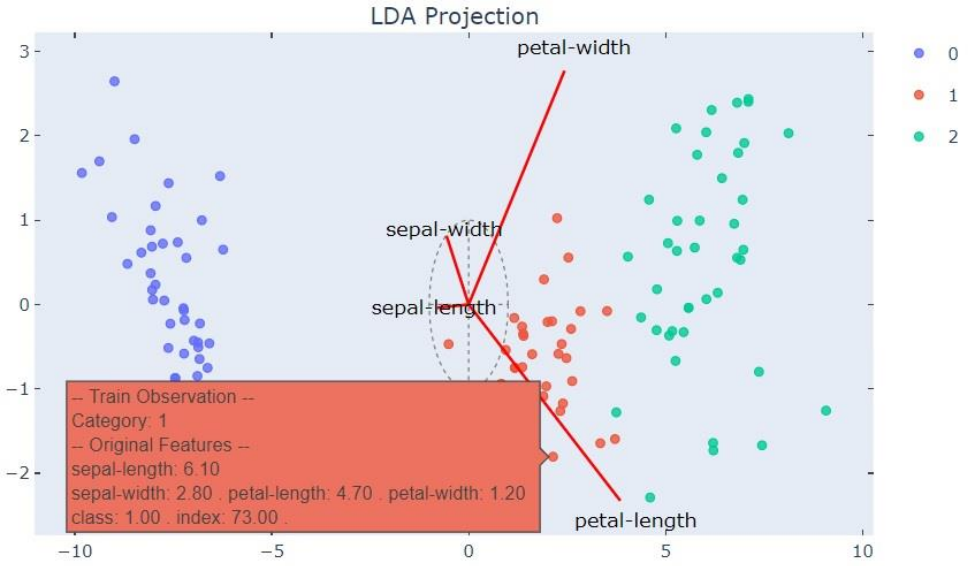


Figura 7: Gráfico bidimensional de las observaciones del conjunto de datos

En esta misma zona se muestra también el árbol de clasificación que se va construyendo a medida que se eligen variables y se particionan los nodos en base a ellas. Como se muestra en la Figura 8 el usuario puede seleccionar un nodo del árbol (que se destacará del resto resaltando en negro su borde) para:

A. Mostrar en la zona de los controles los datos asociados a dicho nodo.

B. Mostrar en el gráfico de proyección LDA las observaciones que caen en ese nodo.

C. Usar los controles de crear/borrar nodos y forzar/deshacer nodo hoja.

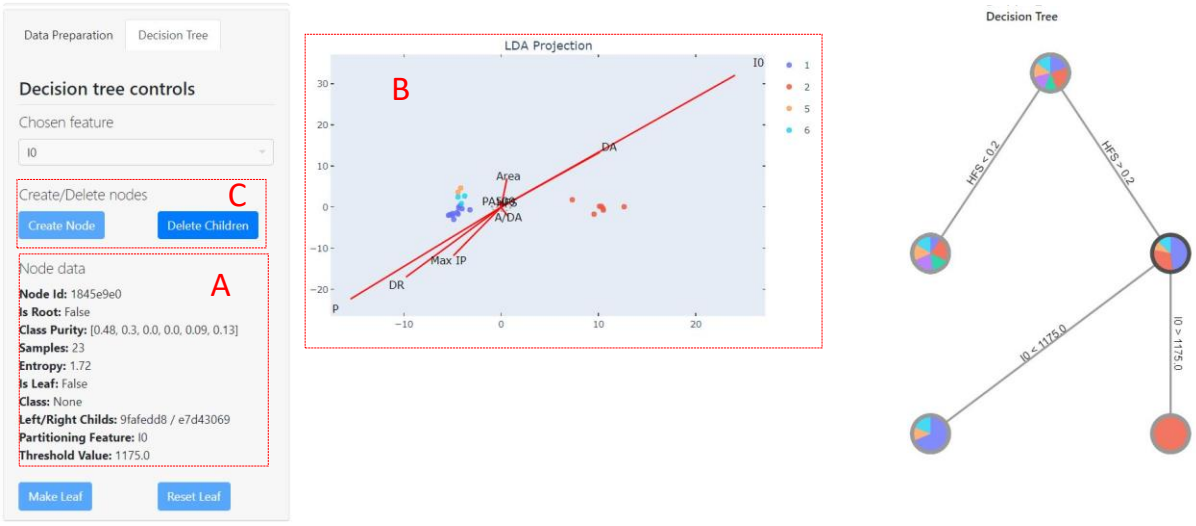


Figura 8: Selección de un nodo del árbol de clasificación

En el ejemplo de la Figura 8, aparecen en la zona A los datos del nodo seleccionado, como, por ejemplo:

- Id del nodo
- Porcentaje de cada clase
- Número de observaciones de ese nodo
- Valor de la entropía
- Variable de partición del nodo
- Etc.

Además, como el nodo seleccionado ya está particionado, los botones de forzar/deshacer hoja están desactivados. Por otro lado, el botón de crear nodo también lo está y el único botón activado es el de borrar nodos, como se ve en la zona C. Finalmente, en el desplegable de selección de variable para particionar el nodo, se muestra la variable ('I0' en este caso) por la cual se particionó el nodo.

Cada nodo se representa además incluyendo un gráfico de pastel en el que el área de cada color representa el porcentaje de observaciones de la clase correspondiente a dicho color en las observaciones que caen en ese nodo del árbol. Así en la imagen de la Figura 8 se puede observar como el nodo raíz contiene aproximadamente la misma cantidad de observaciones de las 6 clases que tiene el conjunto de datos. En concreto para ese nodo la ratio de observaciones de cada clase sobre el total es $[0.18, 0.2, 0.14, 0.12, 0.16, 0.19]$. Por otro lado, en el nodo seleccionado en la imagen se observa en el gráfico de pastel que sólo contiene observaciones de cuatro clases diferentes (las clases 1, 2, 5 y 6) y además la clase morada (la 1 en este caso) parece mayoritaria. En efecto, en los datos del nodo representados en el cuadro de texto de la pestaña de controles del árbol, se observa que las ratios de las clases para las observaciones de ese nodo son $[0.48, 0.3, 0.0, 0.0, 0.09, 0.13]$.

En las zonas 3 y 4 de la interfaz principal se muestran, por un lado, los controles para ejecutar los árboles de clasificación y el exportado del resultado y por otro unas tablas comparativas con las principales medidas de eficacia de ambos árboles. Como se observa en la Figura 9, existe la posibilidad de configurar dos parámetros principales para que se construya de manera automática un árbol de clasificación para los datos cargados a través de la clase **`DecisionTreeClassifier`** del módulo **`tree`** de la librería **`Sklearn`** de **`Python`**: criterio de particionado (entropía - para usar la ganancia de información como criterio de particionado - o gini - para usar la medida de impureza de Gini como criterio de particionado -) y profundidad del árbol.

Una vez que todos los nodos que no tienen hijos del árbol construido manualmente se han declarado como nodos de tipo hoja, entonces se activan los controles que acabamos de mencionar y se puede activar la clasificación (botón de encendido/apagado de la clasificación). En ese momento, la herramienta construye el árbol usando **`Sklearn`**, toma el que el usuario ha construido manualmente y ejecuta la clasificación de las muestras de prueba del conjunto de datos en ambos. En las tablas de la zona 4 de la interfaz gráfica, aparece entonces el resultado de ambos mostrando los valores, para cada clase, de la precisión (porcentaje de acierto al clasificar elementos como pertenecientes a esa clase), exhaustividad (capacidad del clasificador

de encontrar los casos de una clase) y marcador F1 (que es una medida que combina ambas, precisión y exhaustividad). Dichas tablas se obtienen con la función **`classification_report`** de la clase **`metrics`** del paquete **`Sklearn`**.

Figura 9: Secciones 3 y 4 de la interfaz gráfica

4.2 Proceso de construcción de un árbol de clasificación

Para construir un árbol de clasificación hay que seguir los siguientes pasos:

Carga y preparación de datos

En primer lugar, cargar los datos. Los datos tienen que venir en un fichero CSV. Los datos deben tener ciertas características como, por ejemplo:

- Las variables predictoras deben ser numéricas.
- La variable objetivo de la clasificación debe ser un número entero.
- El CSV debe tener el nombre de las variables en la primera línea del fichero.

Por otro lado, los campos del fichero CSV pueden estar separados por cualquier carácter, pudiéndose configurar en los controles de entrada.

A continuación, debemos configurar cuál es la variable que contiene la clase que vamos a predecir con el clasificador y después, del resto de variables, seleccionamos las que queremos usar como variables predictoras pudiendo dejar fuera del clasificador aquellas que el usuario considere que no aportan información en base a la representación gráfica que se muestra en la zona de visualización.

Como último paso de la preparación de datos, el usuario puede determinar el porcentaje de datos que se utilizarán como conjunto de entrenamiento para construir el clasificador. El conjunto de datos se divide de manera aleatoria en base a este porcentaje en grupo de entrenamiento y grupo de prueba, usándose este último para evaluar el clasificador construido manualmente y compararlo con otro construido automáticamente por la librería **`Sklearn`**.

Visualización de los datos

Una vez cargados los datos y preparados, se muestra la visualización del conjunto de datos de entrenamiento de tal manera que:

1. Las observaciones se proyectan en el espacio bidimensional dado por los dos primeros discriminantes lineales que nos da LDA.

2. Las observaciones pertenecientes a cada clase de la variable objetivo se muestran en un color diferente.

3. Se muestran los ejes de una proyección SC que resultarían de proyectar los datos tal y como los proyecta LDA, de acuerdo con lo ya explicado en el capítulo 2.4.

4. El gráfico es reactivo a los controles de preparación de datos, es decir, si por ejemplo se deselecciona una de las variables explicativas, el gráfico se actualiza para mostrar la nueva proyección.

5. A su vez, al lado del gráfico bidimensional de los datos, se muestra el nodo raíz del árbol que vamos a construir. El área de cada color en el nodo raíz nos da las proporciones existentes de las clases en el conjunto de datos de entrenamiento.

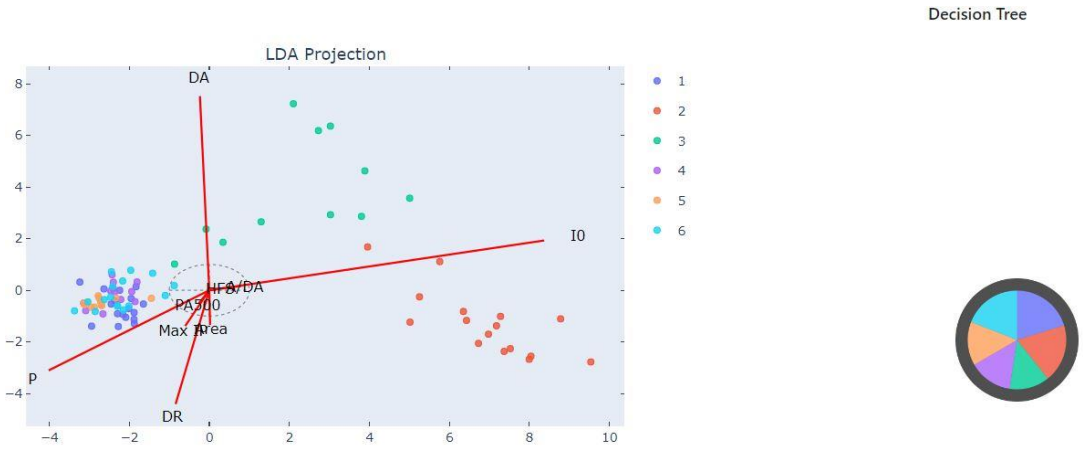


Figura 10: Representación gráfica al cargar un conjunto de datos

Este gráfico puede resultar de ayuda al usuario en un primer momento para hacer una selección inicial de variables que luego se usarán en la construcción del clasificador. Así, de acuerdo con lo ya explicado en el capítulo 2.4, la longitud y dirección de los ejes SC junto con la posición de las observaciones de cada una de las clases en el gráfico, puede darnos una idea de qué variables se pueden eliminar ya que su valor en la tarea de separar las clases sea mínimo.

En la Figura 11, a la izquierda se muestra el conjunto de datos de tipos de vinos [18] proyectado según LDA y a la derecha un zoom sobre la parte central para tener algo más de detalle sobre los ejes SC correspondientes a dicha proyección. Este conjunto de datos tiene 13 variables explicativas. La proyección que nos da LDA separa bastante bien los tres valores de la variable objetivo.

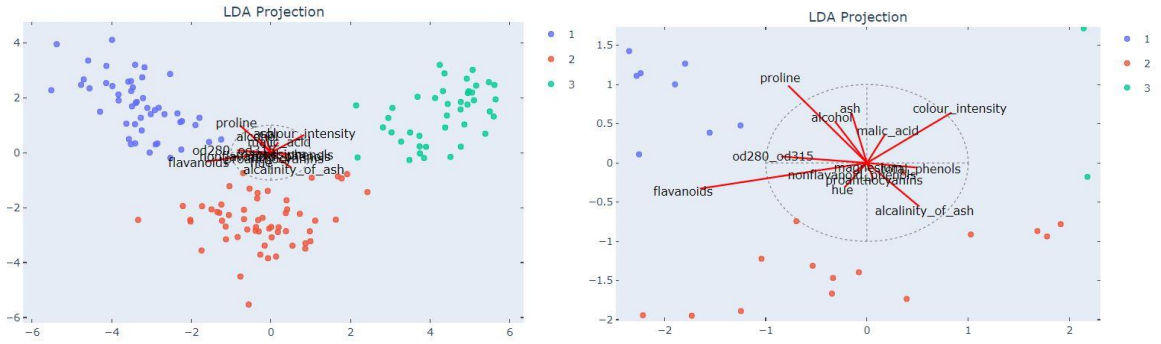


Figura 11: A la izquierda proyección LDA del conjunto de datos de tipos de vino. A la derecha, zoom sobre la parte central para ver el detalle de los 13 ejes SC correspondientes a las 13 variables predictoras que contiene el conjunto de datos

En la Figura 12 podemos ver la proyección de los mismos datos, pero después de descartar los 6 ejes de menor longitud. Con las 7 variables con las que nos hemos quedado, la separación entre clases sigue siendo bastante clara en esta proyección dada por LDA.

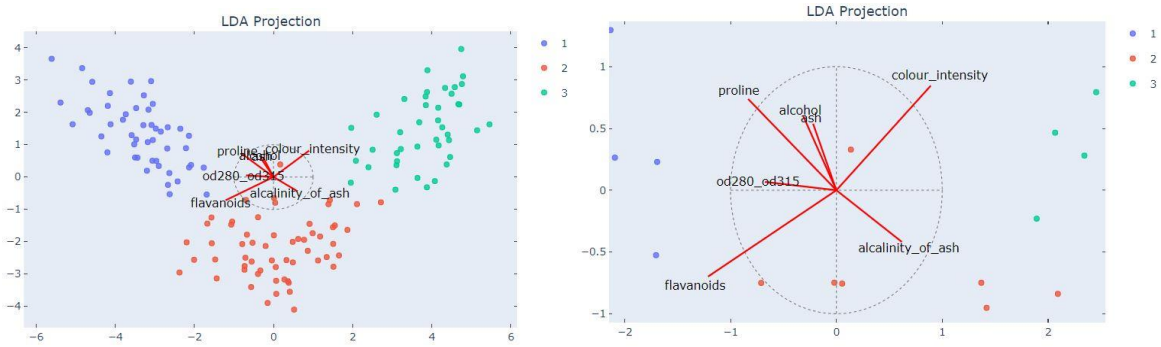


Figura 12: A la izquierda proyección LDA del conjunto de datos de tipos de vino después de descartar los 6 ejes SC de menor longitud. A la derecha, zoom sobre la parte central para ver el detalle de los 7 ejes restantes

Construcción del árbol de clasificación

Una vez que el usuario ha descartado las variables que tienen menor influencia en la separación de las clases, se pasa a construir el árbol. Se selecciona el nodo raíz y en la pestaña "Decision Tree" de los controles, se selecciona la variable por la cual se quiere particionar dicho nodo. Es en este punto en el que el usuario impone su conocimiento del campo de trabajo y no deja en manos del algoritmo de construcción del árbol la elección de la variable. Aunque el algoritmo seleccionará la variable que mejor particione el nodo (desde el punto de vista de la función que se haya elegido para cuantificar dicha partición), el usuario puede imponer su conocimiento del área de trabajo y elegir la que crea mejor en base a su experiencia.

El programa computará entonces la ganancia de información (ver capítulo 2.1) que se obtendrá al particionar dicho nodo, en base a esa variable elegida, para todos los valores posibles de la misma. El resultado será usado para dividir el conjunto de datos de dicho nodo en dos y se mostrarán entonces en el árbol, indicando en cada una de las ramas a qué condición pertenecen los nodos de los que cuelgan, esto es, si para un valor de la variable de partición menor o mayor del valor umbral que se ha encontrado para maximizar la ganancia de información. En cualquier momento, además, se puede elegir ese mismo nodo ya particionado y borrar sus hijos (botón

"Delete Children") para volver a particionarlo usando otra variable y comparar visualmente con el resultado anterior.

En la Figura 13 se muestra cómo el conjunto de datos Iris [19] se ha particionado por la variable "$petal - length$" y cómo el algoritmo ha encontrado el valor 2.6 como valor que maximiza la ganancia de información para particionar el nodo. Puede observarse también cómo, de los dos nuevos nodos creados, uno de ellos (el correspondiente a las observaciones con "$petal - length \leq 2{,}6$") se ha convertido en un nodo hoja al contener únicamente observaciones de la clase 0.

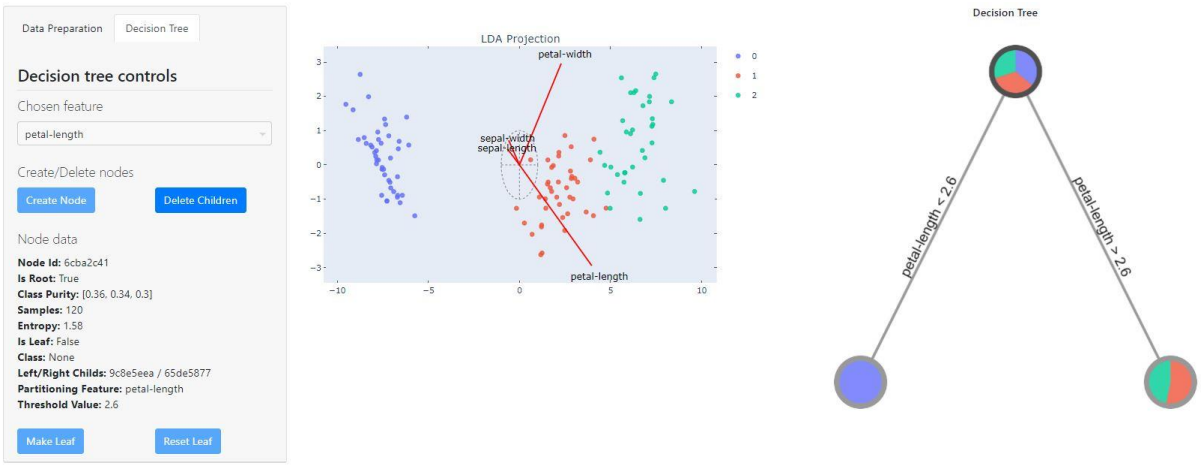


Figura 13: Conjunto de datos Iris particionado por la variable "$petal - length$". Dos nuevos nodos se añaden al árbol de decisión

Si seleccionamos el nodo correspondiente a la condición "$petal - length \leq 2{,}6$", la parte de visualización de las observaciones no mostrará ninguna proyección, ya que no tiene sentido aplicar LDA sobre un conjunto de datos con una única clase. Por el contrario, si nos situamos en el nodo correspondiente a la condición "$petal - length > 2{,}6$" observaremos una visualización diferente de la proyección LDA bidimensional que ya hemos mostrado en otros ejemplos. En la Figura 14 se muestra dicha visualización.

En este caso las observaciones de ese nodo sólo tienen dos valores de la clase objetivo. Como ya se explicó en el capítulo 2.3 LDA produce, como máximo, un número de discriminantes lineales que es el menor de las cantidades $N_c - 1$ o n. Donde N_c es el número de clases diferentes de la variable respuesta y n es el número de variables. Por lo tanto, en ese nodo sólo obtendremos un discriminante lineal y la proyección LDA de las observaciones de ese conjunto de datos se hará sobre una única dimensión.

Para ese caso, hemos elegido un diagrama de distribución de manera que obtenemos una visualización de cómo se distribuyen, a lo largo de ese único eje de proyección dado por LDA, las diferentes observaciones de las dos clases del nodo. Debajo de la proyección, y en vez de los ejes SC que mostrábamos en proyecciones bidimensionales, se muestra una tabla que contiene la contribución de cada una de las variables en ese discriminante lineal, de manera que nos puede ayudar a elegir de nuevo qué variable usar para particionar ese nodo.

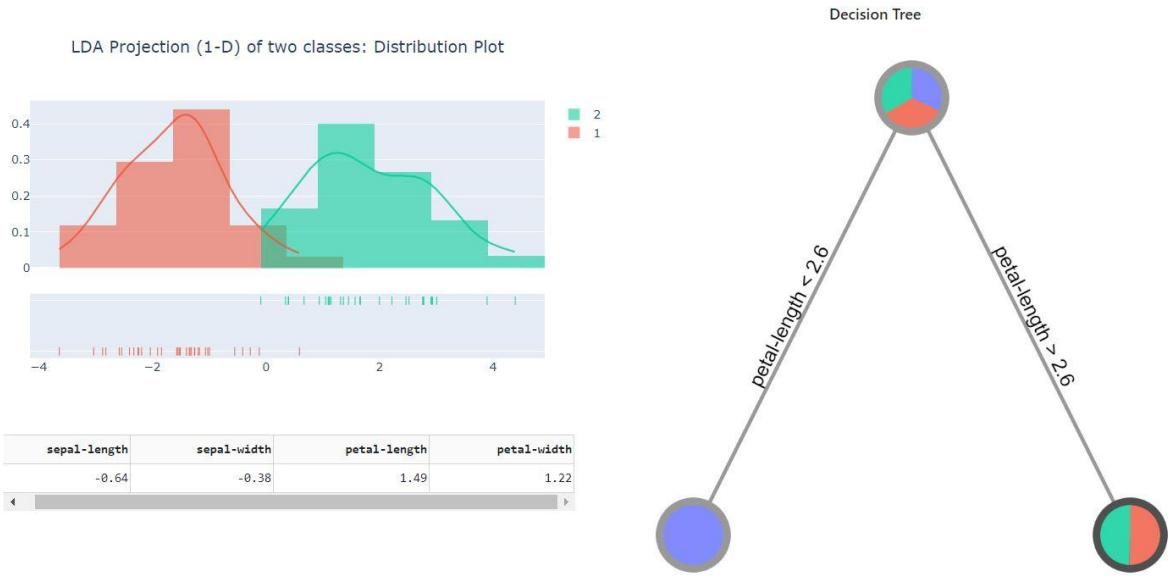


Figura 14: Representación gráfica de un nodo con dos únicas clases de la variable objetivo

4.3 Comparación de los clasificadores

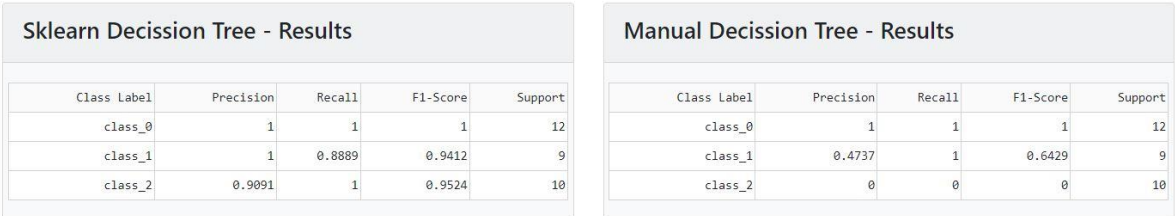
Una vez que hemos construido el árbol de clasificación y hemos caracterizado todos los nodos finales como nodos hoja, entonces podremos ejecutar el clasificador construido sobre los datos de prueba. Además, la herramienta construirá a su vez un clasificador usando **`Sklearn`** y se mostrarán unas tablas con las métricas principales de ambos para poder compararlos.

Para construir el árbol con la librería **`Sklearn`** se puede elegir dos parámetros:

- Criterio de partición el nodo: Los valores posibles son “entropía” y “gini”. En el caso de “entropía”, la función usada para medir la calidad de la partición del nodo será la ganancia de información. En el caso de elegir “gini” será el criterio de impureza de Gini. Nótese que el árbol que se construye manualmente utiliza siempre el primero de ellos (ganancia de información).

- Profundidad del árbol: Este parámetro nos permite indicar la máxima profundidad del árbol que construirá la librería.

Una vez elegidos los parámetros se puede ejecutar la clasificación. Se mostrarán entonces dos tablas como las de la Figura 15.



Sklearn Decission Tree - Results

Class Label	Precision	Recall	F1-Score	Support
class_0	1	1	1	12
class_1	1	0.8889	0.9412	9
class_2	0.9091	1	0.9524	10

Manual Decission Tree - Results

Class Label	Precision	Recall	F1-Score	Support
class_0	1	1	1	12
class_1	0.4737	1	0.6429	9
class_2	0	0	0	10

Figura 15: Tablas comparativas de los dos clasificadores

Las tablas se construyen automáticamente con la ayuda de la función **`classification_report`** de la clase **`metrics`** del paquete **`Sklearn`**. Tenemos dos tablas, una correspondiente al árbol que la herramienta genera automáticamente usando la clase **`DecisionTreeClassifier`** del módulo **`tree`** de la librería **`Sklearn`** de **`Python`**, y otra correspondiente al árbol que el usuario ha construido manualmente apoyado en la visualización de los datos.

Las tablas muestran, para cada clase, la precisión, la exhaustividad, el marcador F1 (siendo éste una métrica que combina las dos anteriores) y el número de observaciones de dicha clase.

- Precisión: número de observaciones de una clase correctamente clasificadas sobre el total de las observaciones clasificadas como de esa clase.

- Exhaustividad: Número de observaciones clasificadas como de una clase sobre el total de observaciones totales de esa clase.

- Marcador F1: Se calcula como $F_1 = 2 \cdot \frac{Precisión \cdot Exhaustividad}{Precisión + Exhaustividad}$

La precisión de una clase nos ayuda a responder a la pregunta: ¿qué proporción de las observaciones clasificadas como pertenecientes a una clase realmente pertenecen a esa clase? Por otro lado, la exhaustividad nos ayuda a responder a la pregunta: ¿qué proporción de las observaciones de una clase se clasifica correctamente?

Como paso final de esa comparación el usuario puede exportar a un fichero PDF ambos árboles para así poder comparar su estructura y sus principales datos, como variables de partición, puntos de corte, distribución de observaciones las clases en cada uno de los nodos etc. Para ello se hace uso de la librería **`Graphviz`** [20]. La librería **`Sklearn`** contiene una función para obtener una representación en formato DOT [21] del árbol que ha construido. Por otro lado, la herramienta diseñada contiene una función que es capaz de generar una representación en ese mismo formato del árbol que hemos construido manualmente. Ambas representaciones se exportan a un archivo PDF. La Figura 16 y la Figura 17 muestran el resultado del exportado de los árboles de clasificación construidos manualmente y de manera automática tras ejecutar la herramienta para el conjunto de datos Iris.

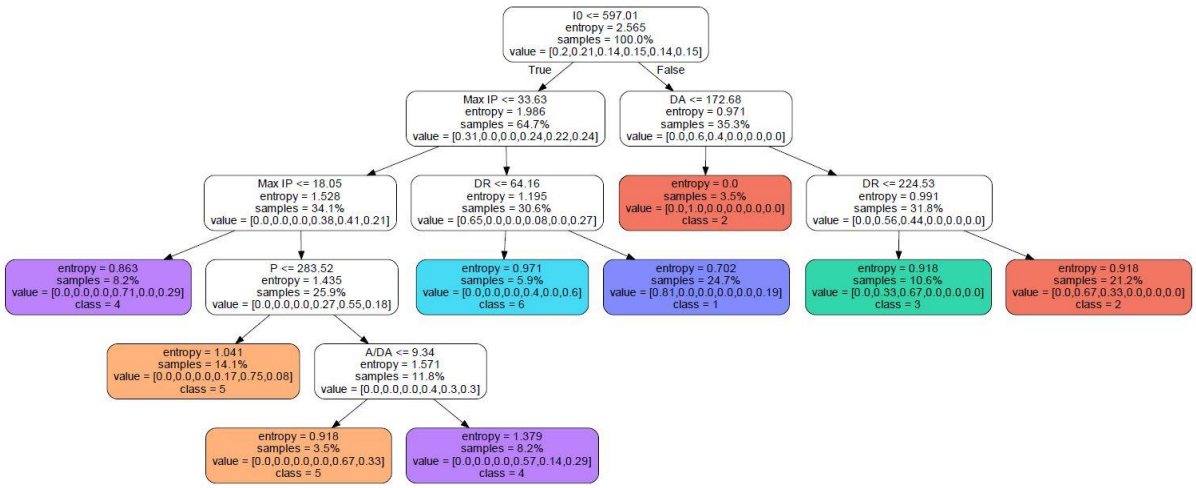


*Figura 16: Árbol construido por el usuario exportado por **Graphviz** a un fichero PDF*

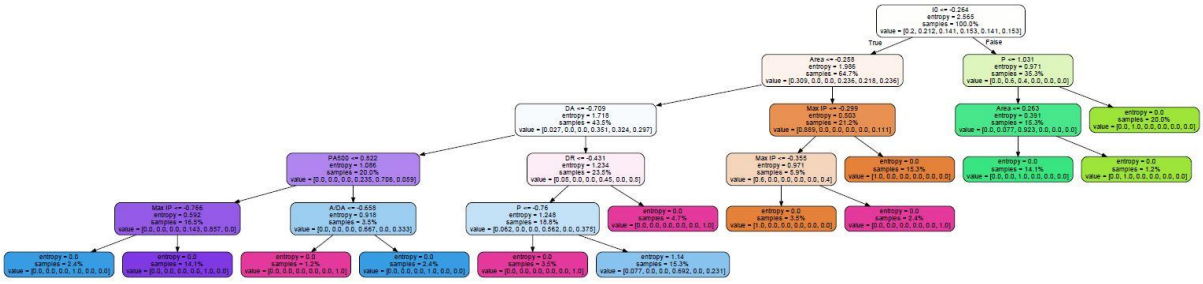


*Figura 17: Árbol generado automaticamente por la librería **Sklearn** exportado a un fichero PDF*

En el caso del árbol generado automáticamente, la librería de **`Sklearn`** genera una representación DOT en la que utiliza un degradado de colores para indicar la proporción de las clases mayoritarias en cada nodo. En el caso del árbol exportado basado en el que el usuario ha construido manualmente, hemos optado por usar únicamente color en los nodos hoja, es decir, nodos en los que se clasifica cada observación como perteneciente a una clase. Nótese que la paleta de colores utilizada en este caso para cada clase coincide con la usada para representar las clases en los gráficos de la proyección LDA y en los gráficos de pastel de los nodos del árbol.

Parte II:

Caso práctico

Capítulo 5.

5 Caso práctico

5.1 Introducción

Como se ha descrito en el resumen de este TFM, la idea inicial del trabajo surge del artículo de investigación 'Visually guided classification trees for analyzing chronic patients' [1], uno de cuyos autores es el director de este TFM.

En dicho artículo de investigación se describe en detalle el proceso de construcción de árboles de decisión basado en la visualización de los datos. En concreto, el artículo describe los pasos y resultados de usar dicha técnica para crear árboles de clasificación usando datos reales sobre pacientes del HUF sanos o con diagnósticos de enfermedades crónicas como la diabetes o la hipertensión.

A lo largo del artículo se describen los pasos dados y los resultados obtenidos en la construcción de diferentes árboles de clasificación, cada uno de ellos centrado en el diagnóstico de un tipo concreto de pacientes. El proceso seguido es el ya descrito en el capítulo 4, haciendo uso de las visualizaciones combinadas de SC y LDA sobre el conjunto de datos en estudio. Los usuarios del método (personal clínico) construye diferentes árboles de clasificación en base a las visualizaciones de los datos, pero también, y más importante, aplicando su conocimiento del campo de actividad sobre dichas visualizaciones a la hora de seleccionar las variables usadas para particionar cada nodo a lo largo del árbol.

En este capítulo vamos a seleccionar uno de los árboles construidos en el citado artículo y vamos a hacer uso de la herramienta desarrollada para comprobar cómo se construiría ese mismo árbol en ella y, más importante aún, haciendo uso de la funcionalidad de clasificación y comparación de la herramienta, cómo se comporta dicho clasificador construido manualmente frente a uno que construiría automáticamente la librería de **`Skelarn`**.

No es el objetivo de este ejercicio el sacar conclusiones médicas o clínicas acerca de los datos recopilados o de las clasificaciones construidas. Todas las conclusiones están ya analizadas y explicadas en el citado artículo. El principal objetivo de este ejercicio es, por el contrario, utilizar la herramienta en un caso práctico real y añadir a las conclusiones del artículo las que se pueden derivar de la comparación de un árbol construido manualmente frente a uno construido automáticamente por una librería de aprendizaje máquina.

5.2 Descripción de los datos

Los datos usados son los mismos que en el citado artículo de investigación. Se recopilaron durante el año 2012 por el HUF, en Madrid, España. Para cada paciente se recopilaron los siguientes datos:

- Edad

- Sexo
- Fármacos dispensados (ejemplo: A10BA)
- Códigos de diagnóstico (ejemplo: 250)

Los fármacos dispensados se codificaron usando el sistema de clasificación anatómica, terapéutica, química (ATC por sus siglas en inglés) [22]. El ATC es un índice de sustancias farmacológicas y medicamentos, organizados según grupos terapéuticos. El código recoge el sistema u órgano sobre el que actúa, el efecto farmacológico, las indicaciones terapéuticas y la estructura química del fármaco.

Por su parte los códigos de diagnóstico están codificados usando método de clasificación de enfermedades internacional, 9ª revisión, modificación clínica (ICD9-CM por sus siglas en inglés) [23]. Aunque estos códigos permiten la categorización usando subcategorías, se hizo un trabajo previo para omitir las subcategorías y tener un número suficiente de pacientes con los mismos códigos. Por lo tanto, pacientes codificados con la misma categoría (codificado usando 3 caracteres alfanuméricos) pero diferente subcategoría tendrán el mismo código de diagnóstico.

Finalmente, basado en los datos demográficos, de diagnóstico y farmacológicos, cada individuo se asignó a un único y mutuamente exclusivo grupo de riesgo (CRG) identificado por un número de cinco cifras. Al igual que en el caso de los códigos de diagnóstico, se hizo un trabajo de consolidación de los códigos omitiendo el quinto dígito que indica la severidad de la enfermedad. Con todo esto, el ejercicio replicado en este TFM se fija en los siguientes grupos de riesgo, que constituirán las clases de nuestra variable objetivo en el clasificador construido:

1. CRG5192: Hipertensión (codificado como clase 2)
2. CRG6144: Diabetes e hipertensión (codificado como clase 4)
3. CRG7071: Diabetes, hipertensión y otra enfermedad crónica dominante (codificado como clase 5)

La muestra de datos utilizada contiene 539 observaciones de cada una de esas clases. Para cada uno de los códigos de diagnóstico o los fármacos dispensados, las observaciones contendrán un 1 o un 0 en función de que se haya usado dicho código (de diagnóstico o farmacológico) en su categorización.

5.3 Construcción del árbol

Se ha construido manualmente el árbol de clasificación que replica el ejemplo construido en el artículo mostrado en la Figura 6 del mismo. Para ello se ha usado el 80% de los datos como conjunto de entrenamiento (elegido aleatoriamente) dejando el 20% restante como conjunto de datos de prueba.

En cada nodo, los usuarios (personal clínico) eligieron la variable que mejor creían que particionaba el nodo en función de la visualización generada y de su conocimiento del área de trabajo. En la Figura 18 puede verse uno de los nodos del árbol durante su generación. En este caso, en dicho nodo sólo existen observaciones de dos clases (CRG6144 y CRG7071) y, por lo tanto, la visualización asociada es un diagrama de distribución sobre el único discriminante lineal obtenido por LDA.

Figura 18: Proceso de creación de un árbol de decisión manual usando los datos sobre pacientes con enfermedades crónicas el Hospital Universitario de Fuenlabrada

En cada uno de los pasos dados para particionar un nodo, el usuario puede hacer uso de la visualización asociada para ver cuál es la variable que mejor se puede usar para separar ciertas clases, apoyado claro está en su conocimiento de la materia. Así, por ejemplo, en la Figura 19 puede verse como en el nodo seleccionado, que contiene una mayoría de observaciones de la clase 2 (CRG5192), se puede usar la variable 'A10BA' para particionar el nodo ya que parece que es la que más influye en la separación de las observaciones de la clase predominante. De hecho, como se observa en los nodos resultantes, dicha clase (CRG5192) ya sólo está contenida en uno de ellos, siendo aún mayoritaria con un mayor porcentaje y pudiendo marcar dicho nodo como tipo hoja. Las observaciones que caigan en él serán clasificadas como pacientes del grupo de riesgo CRG5192.

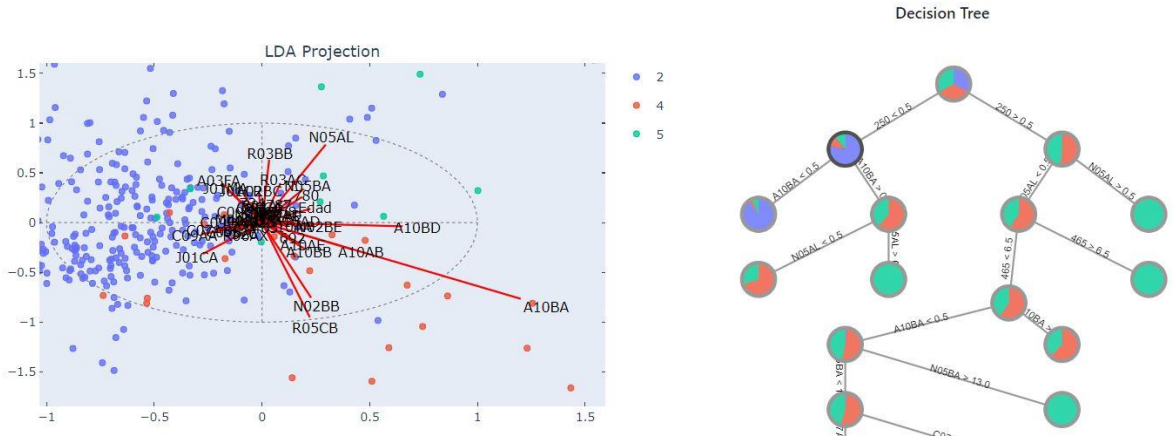


Figura 19: Detalle de la visualización SC y LDA para uno de los nodos del árbol. La variable 'A10BA' es la que más influye en la separación de las observaciones de la clase codificada como clase 2

5.4 Comparación de los clasificadores

Una vez construido el clasificador manualmente, que replica al construido por el personal clínico en el artículo, hacemos uso de la extensión de funcionalidad que tiene la herramienta para poder sacar algunas conclusiones sobre la eficacia de este método y, sobre todo, la diferencia en la interpretación del modelo que puede tener un árbol de este tipo sobre otros construidos de manera automática.

El árbol construido tiene 14 niveles de profundidad. Si configuramos ese mismo nivel de profundidad para el árbol que construye la librería de **`Sklearn`** y usamos el mismo criterio de particionado de nodos (entropía), obtenemos los siguientes resultados en las tablas de comparación, como puede verse en la Figura 20 y la Figura 21.

Sklearn Decission Tree - Results

Class Label	Precision	Recall	F1-Score	Support
class_2	0.9565	0.9821	0.9692	112
class_4	0.6783	0.7959	0.7324	98
class_5	0.7979	0.6579	0.7212	114

*Figura 20: Tabla de métricas de la clasificación del árbol construido por la librería de **Sklearn** sobre el conjunto de los datos de prueba usando entropía como criterio de partición y 14 niveles de profundidad*

Manual Decission Tree - Results

Class Label	Precision	Recall	F1-Score	Support
class_2	0.9256	1	0.9614	112
class_4	0.62	0.949	0.75	98
class_5	0.9623	0.4474	0.6108	114

Figura 21: Tabla de métricas de la clasificación del árbol construido manualmente sobre el conjunto de los datos de prueba

Por otro lado, los árboles que cada uno de los clasificadores ha construido son los siguientes:

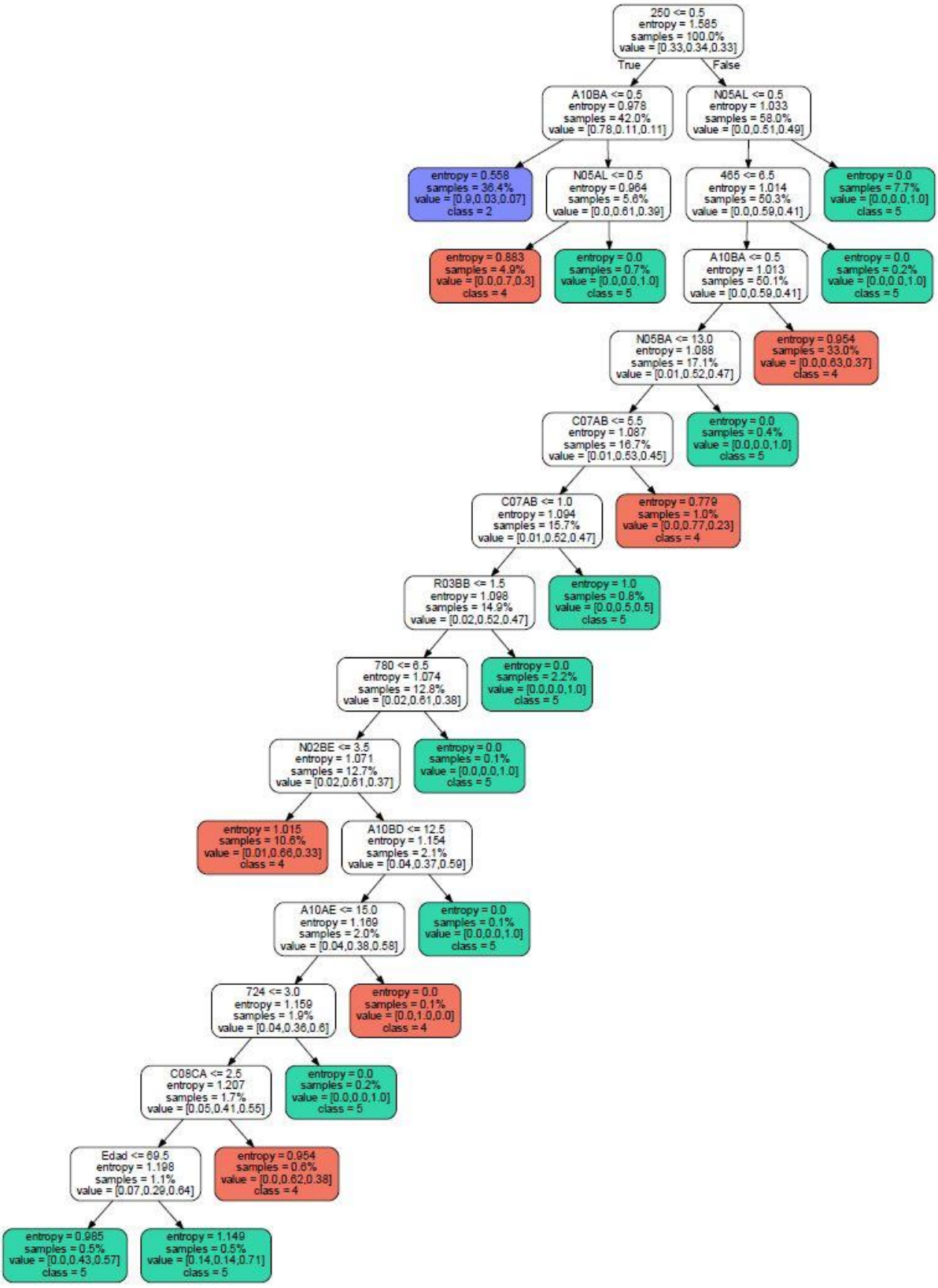


Figura 22: Árbol construido manualmente exportado por la herramienta

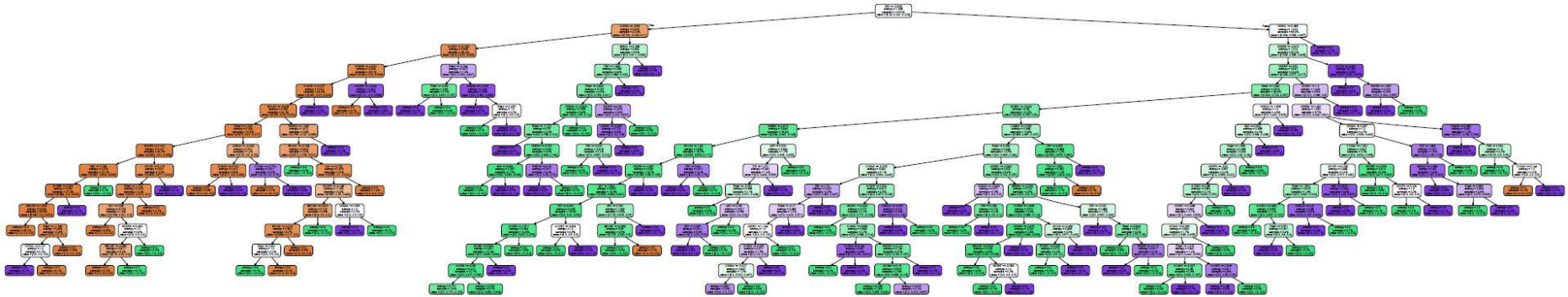


*Figura 23: Árbol construido por la librería **Sklearn** exportado por la herramienta*

De la Figura 22 y la Figura 23 puede observarse como, al dejar crecer hasta un número alto de niveles el árbol generado automáticamente, éste se vuelve muy complejo y difícil de interpretar. Mientras que el número de nodos es de 244, el árbol construido manualmente contiene únicamente 35 (ambos con 14 niveles de profundidad), mostrándose por tanto mucho más interpretable.

Es interesante, por ejemplo, comparar cómo se clasifican las observaciones de la clase 2 (CRG5192, que corresponde con pacientes incluidos en el grupo de riesgo con Hipertensión) en ambos árboles. Mientras que en el árbol construido manualmente sólo se clasifican en dos pasos (ausencia de código de diagnosis '250' y ausencia de código farmacológico A10BA), en el árbol construido automáticamente hay 15 ramas diferentes que llevan a esa misma clasificación.

Por otro lado, si tomamos el marcador F1 (que conjuga la precisión y la exhaustividad) de cada clase como métrica para comparar ambos clasificadores observamos que:

1. Para la clase CRG5192 → El clasificador **automático** mejora en un $\mathbf{0.78}$ % al clasificador **manual**

2. CRG6144 → El clasificador **manual** mejora en un $\mathbf{1.76}$ % al clasificador **automático**

3. CRG7071 → El clasificador **automático** mejora en un $\mathbf{11.04}$ % al clasificador **manual**

Siendo claro que hay una clase (CRG7071) en la que el clasificador automático es netamente mejor que el clasificador manual, en otras dos clases (CRG5192, CRG6144) ambos obtienen resultados parecidos e incluso el clasificador manual es ligeramente mejor en uno de ellos (CRG6144).

Parte III:

Conclusiones

Capítulo 6.

6 Conclusiones

6.1 Conclusiones

Se puede concluir de este trabajo que, en conjuntos de datos relativamente sencillos o pequeños (no hemos trabajado con conjuntos de datos que tuvieran cientos o miles de variables explicativas), es posible construir manualmente un árbol de clasificación que trabaja realmente bien en comparación con uno construido de forma automática por una librería de aprendizaje máquina. El conocimiento del área de trabajo, junto con la información sobre las variables y cómo influyen en la separación de las clases que uno puede obtener de ciertas visualizaciones de los datos, son herramientas muy potentes y ofrecen una alternativa a la hora de decidir qué variables deben usarse para ir clasificando las observaciones a lo largo de los diferentes nodos del árbol.

Por otro lado, una herramienta web con una interfaz sencilla y una funcionalidad adecuada puede ser muy útil para construir árboles que sean fácilmente interpretables a la vez que obtengan resultados comparables con los de un clasificador automático. El proceso de creación del árbol es rápido y sencillo, ayudando al usuario a comprender mejor la evolución de la clasificación a medida que el árbol crece y ofreciendo, con suma facilidad, opciones de prueba y error a través de la creación y borrado de nodos. Esa interactividad y su facilidad de uso hacen de esta herramienta una opción atractiva para usuarios finales de clasificadores resultantes.

En definitiva, este trabajo no hace sino reforzar la idea de que el conocimiento del dominio de trabajo es clave a la hora de crear y aplicar algoritmos de aprendizaje máquina.

6.2 Desarrollo futuro

Como desarrollo futuro se podría pensar en mejorar la interfaz de usuario. Bajo algunas circunstancias, el área de trabajo no se adapta demasiado bien al espacio disponible (reducción de la ventana del navegador, resoluciones muy bajas etc.). Una mejor disposición de los elementos visuales podría darle un aspecto más profesional. También se podría pensar en adaptarla a dispositivos móviles como tabletas o teléfonos inteligentes.

Por otro lado, quizás convendría pensar en otro tipo de área de trabajo para construir el árbol ya que cuando crece mucho se vuelve complejo trabajar con él en el área disponible actualmente en la herramienta. Una opción podría ser una ventana flotante en la que el árbol pueda visualizarse a pantalla completa mientras que en otra pantalla tenemos los controles con los que vamos manejándolo.

6.3 Apreciación personal

Este proyecto ha supuesto un reto personal interesante de afrontar. Por un lado, sin conocimientos previos sobre desarrollo web o **`Dash`** lo primero que tuve que hacer fue familiarizarme con esa tecnología. Es cierto que es muy interesante ya que permite desarrollar aplicaciones web sin necesidad de adentrarse en el mundo de **`JavaScript`** o **`CSS`** y el resultado es bastante bueno con poco tiempo de aprendizaje. El hecho de que esté basada en **`Plotly`** le da, además, un potencial enorme a la hora de representar gráficos con un gran poder de interactividad sin necesidad de codificar nada especial ya que la propia librería lo da por defecto.

Por otro lado, como la primera fase del trabajo era incluir un módulo en una herramienta ya existente, eso supuso una tarea de entender un código desarrollado previamente por otras personas. Cualquiera que haya pasado por esa tarea sabe que no es fácil.

Ese paso me dejó una cierta insatisfacción ya que no tenía algo que yo hubiera desarrollado en su totalidad. De ahí surgió la idea de desarrollar una herramienta totalmente nueva y desde cero que se centrase únicamente en la parte de creación de un árbol de clasificación con apoyo en la visualización de datos.

La parte más teórica, en la que se basa la idea de la construcción del árbol ayudado por la visualización de los datos, también me ha resultado muy interesante. La clave de dicha idea es la descrita en el capítulo 2.4: la combinación de la visualización SC con LDA haciendo coincidir los ejes SC de manera que proyecten los puntos originales en el espacio de los dos primeros discriminantes lineales que nos da LDA. Entendiendo cómo afectan dichos ejes en los puntos proyectados y cómo LDA proyecta a su vez los puntos, se puede ir construyendo el árbol manualmente y obtener resultados de clasificación realmente buenos. Ciertamente interesante, la verdad.

Abreviaturas

ATC: Anatomical Therapeutic Chemical

CRG: Clinical Risk Groups

CSV: Comma Separated Values

ICD9-CM: International Classification of Diseases, Ninth Revision, Clinical Modification

LDA: Linear Discriminant Analysis

PCA: Principal Components Analysis

PDF: Portable Document Format

SC: Star Coordinates

TFM: Trabajo Fin de Máster

HUF: Hospital Universitario de Fuenlabrada

URL: Uniform Resource Locator

Referencias

1. Soguero-Ruiz et al. BMC Bioinformatics 2020, 21(Suppl 2):92. Visually guided classification trees for analyzing chronic patients.

2. Trevor Hastie, Robert Tibshirani, Jerome Friedman. The Elements of Statistical Learning. Second Edition. February 2009 https://web.stanford.edu/~hastie/ElemStatLearn/

3. The Use of Multiple Measurements in Taxonomic Problems. Ronald A. Fisher (1936) https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1469-1809.1936.tb02137.x https://en.wikipedia.org/wiki/Linear_discriminant_analysis

4. Kandogan E. Star coordinates: A multi-dimensional visualization technique with uniform treatment of dimensions. In: Proceedings of the IEEE Information Visualization Symposium, Late Breaking Hot Topics. New Yersey: IEEE Computer Society; 2000. p. 9–12.

5. Scikit-Learn Machine Learning in Python [Online] (Marzo 2020) https://scikit-learn.org/stable/

6. sklearn.tree.DecisionTreeClassifier [Online] (Marzo 2020) https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html?highlight=decisiontree#sklearn.tree.DecisionTreeClassifier

7. sklearn.tree: Decision Trees [Online] (Marzo 2020) https://scikit-learn.org/stable/modules/classes.html#module-sklearn.tree

8. Dash User Guide [Online] (Marzo 2020) https://dash.plotly.com/

9. Flask Web development, one drop at a time [Online] (Marzo 2020) https://flask.palletsprojects.com/en/1.1.x/

10. Plotly JavaScript Open Source Graphing Library [Online] (Marzo 2020) https://plotly.com/javascript/

11. React Una biblioteca de JavaScript para construir interfaces de usuario [Online] (Marzo 2020) https://es.reactjs.org/

12. C4.5 algorithm [Online] (Marzo 2020) https://en.wikipedia.org/wiki/C4.5_algorithm

13. Entropy (information theory) [Online] (Marzo 2020) https://en.wikipedia.org/wiki/Entropy_(information_theory)

14. Information Gain and Mutual Information for Machine Learning [Online] (Marzo 2020) https://machinelearningmastery.com/information-gain-and-mutual-information/

15. RUBIO-SÁNCHEZ M., RAYA L., DÍAZ F., SANCHEZ A.: A comparative study between radviz and star coordinates. IEEE Transactions on Visualization and Computer Graphics 22, 1 (January 2016), 619–628.

16. D. Rojo, L. Raya, M. Rubio-Sánchez and A. Sanchez / A Visual Interface for Feature Subset Selection Using Machine Learning Methods. CEIG - Spanish Computer Graphics Conference (2018).

17. JavaScript [Online] (Marzo 2020) https://developer.mozilla.org/en-US/docs/Web/JavaScript

18. Machine Learning Repository Wine Dataset [Online] (Marzo 2020) http://archive.ics.uci.edu/ml/datasets/wine

19. Machine Learning Repository Iris Dataset [Online] (Marzo 2020) https://archive.ics.uci.edu/ml/datasets/Iris

20. Graphviz - Graph Visualization Software [Online] (Marzo 2020) https://www.graphviz.org/

21. The DOT Language [Online] (Marzo 2020) https://graphviz.gitlab.io/_pages/doc/info/lang.html

22. ATC/DDD Index 2020 Structure and principles [Online] (Marzo 2020) https://www.whocc.no/atc/structure_and_principles/

23. International Classification of Diseases,Ninth Revision, Clinical Modification (ICD-9-CM) [Online] (Marzo 2020) https://www.cdc.gov/nchs/icd/icd9cm.htm