Universidad de **Valladolid**

Escuela de Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática
Mención Computación

Integración de herramientas estadísticas con Python

Autor:
Elvira Delgado de Paz

Universidad de **Valladolid**

Escuela de Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática
Mención Computación

Integración de herramientas estadísticas con Python

Autor:
Elvira Delgado de Paz

Tutores:
Mario Corrales Astorgano
Cristian Tejedor García

A los que ya no están, pero no se han ido.

Agradecimientos

Quiero agradecer a todos los profesores que me han formado, ya sea en la universidad como en el colegio e instituto, por ayudarme a desarrollar habilidades que ni yo sabía que poseía. Gracias a mis tutores Mario y Cristian por su gran ayuda y esfuerzo a lo largo de todo este proyecto y por la confianza depositada en mi desde el primer momento.

Gracias a todas las personas que he conocido a lo largo de estos años en la universidad y fuera de ella y que me han enseñado y ayudado a ser quien soy ahora mismo.

Gracias a mi familia, de los que aprendo cada día y de los que seguiré aprendiendo.

Por último, gracias a todas las mujeres de mi vida, luchadoras incansables.

Resumen

En los últimos años, el cambio de las tecnologías analógicas a digitales ha supuesto un gran crecimiento de la generación de datos de todo tipo. A partir de estos datos puede ser interesante obtener información que pronostique acciones futuras y mida el desempeño de las acciones de una empresa. Por ello, existe una alta variedad de herramientas para analizar datos entre las que escoger, ya que cada una proporciona una funcionalidad distinta. Debido a la facilidad de aprendizaje, su versatilidad y su extensa comunidad de desarrollo, Python se ha impuesto últimamente ante otros softwares como primera elección entre los desarrolladores. Sin embargo, aunque es uno de los lenguajes de programación con un aprendizaje más sencillo, para realizar un análisis profundo de los datos es necesario utilizar múltiples librerías y ejecutar comprobaciones sobre la composición y estructura de los datos y así elegir las técnicas más adecuadas. Por todo lo anterior, el objetivo principal de este proyecto es implementar una aplicación que integre múltiples funcionalidades estadísticas en una sola librería de Python, accesible a cualquier usuario y fácil de usar. Además de la posibilidad de ampliar su funcionalidad en un futuro.

Abstract

In the last years, the change from analog to digital technologies has led to a great growth in the generation of all kinds of data. From these data it could be interesting to obtain information that forecasts future actions and measures the performance of a company's actions. Therefore, there is a wide variety of data analysis tools that the user can choose, since each one provides a different functionality. Python has prevailed over other software programs as the first choice among developers due to its easy learning, versatility and extensive development community. Even though it is one of the programming languages easier to learn, to perform a deep analysis of the data it is necessary to use multiple libraries and run checks on the composition and structure of the data and thus choose the most appropriate techniques. For all of the above reasons, the main goal of this project is to implement an application that integrates multiple statistical functionalities in a single Python library, accessible to any user and easy to use. Futhermore, the application code will be extensible and flexible for adding new funcionality.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Motivación

El análisis estadístico está relacionado con los procedimientos aplicados para obtener conclusiones a partir de conjunto de datos [1]. En el ámbito de la investigación, el análisis estadístico de datos es una tarea común y se lleva a cabo con distintos lenguajes de programación. Sin embargo, la mayoría de las veces los *scripts* utilizados durante el desarrollo de una investigación no son reutilizables para otros estudios teniendo que volver a escribir código y adaptarlo al conjunto de datos utilizado.

Hoy en día existen numerosas plataformas para el análisis estadístico, como Python, SPSS, R o SAS, entre otras. No obstante, sin conocimientos previos de informática es difícil introducirse en un lenguaje de programación que nunca se ha utilizado, ya que conlleva una gran curva de aprendizaje, especialmente en las primeras etapas. R y SPSS son softwares principalmente empleados para el análisis de datos, ya que ambos son herramientas enfocadas al estudio de conjuntos de datos. Sin embargo, en este TFG se ha decidido emplear Python, debido a su legibilidad de código, eficiencia, desarrollo constante del lenguaje y la cantidad de bibliotecas integradas que posee. Además, es un lenguaje de código abierto, en el que se puede acceder a sus recursos gratuitamente [2].

A lo largo de los años se han ido añadiendo bibliotecas estadísticas o de análisis estadístico en Python para extender la funcionalidad de este, importando paquetes y módulos disponibles, con el objetivo de reducir el tiempo y el esfuerzo en el desarrollo. Además, la creación de librerías en este lenguaje de programación es algo simple y accesible para cualquier programador de código. Para la utilización de las bibliotecas integradas en Python es necesario leer y profundizar en la documentación y, normalmente, este no es un proceso sencillo. Además, no existe una única librería que realice un análisis profundo de los datos y que aplique la mejor técnica estadística según la composición de estos.

Durante los últimos años, en el grupo ECA-SIMM de la UVa se ha utilizado Python junto a R y SPSS para el análisis estadístico en investigación. Sin embargo, no se dispone de un repositorio único en el que se pueda reutilizar las herramientas de análisis estadístico, llevando en ocasiones a perdida de tiempo programando una y otra vez los mismos procedimientos.

Por todo lo anterior, este TFG pretende implementar una serie de servicios software en una aplicación fácil de usar, que integre sistemas de clasificación automática y estadística descriptiva e inferencial. Todo ello en un mismo repositorio y accesible a cualquier usuario que pueda necesitarlo.

1.2. Objetivos

El objetivo principal del proyecto es el desarrollo de un repositorio común para herramientas de clasificación automática, estadística descriptiva e inferencial mediante el lenguaje de alto nivel Python.

El objetivo general se puede dividir en objetivos específicos:

- **Objetivo 1:** Implementación de un método de reconocimiento, fusión y escritura de ficheros.
- **Objetivo 2:** Desarrollo de un sistema que incluye clasificadores y selección de características.
- **Objetivo 3:** Creación de un sistema que realice descripciones estadísticas, test inferenciales, correlaciones y concordancias.
- **Objetivo 4:** Desarrollo de diferentes pruebas semiautomáticas para asegurarse de su correcto funcionamiento.

1.3. Metodología

Debido a las características del proyecto, la tecnología empleada y el hecho de que una sola persona sea el desarrollador se ha escogido un modelo de trabajo **iterativo e incremental** [3]. Esta planificación permite la división del trabajo en varios bloques denominados *iteraciones* los cuales se fijan desde un inicio y permite un cierto grado de modificación.

Todo ello requiere ejecutar un rápido proceso de implementación e iteraciones posteriores que ofrezcan mejoras sobre el proyecto inicial [3]. Principalmente este modelo de trabajo se fundamenta en dos premisas [4]:

- Los usuarios nunca saben bien que es lo que necesitan para satisfacer sus necesidades.
- Durante el desarrollo del proyecto, los procesos tienden a cambiar.

Algunas de las ventajas de esta metodología serían el conocimiento del cliente de la evolución del proyecto después de finalizada cada iteración y la reducción del *gold-plating* (funcionalidad no requerida por el cliente).

En la Figura 1.1, se pueden observar los pasos a seguir mediante una metodología iterativa e incremental. En el Capítulo 4 se explicarán detalladamente cada una de las iteraciones.

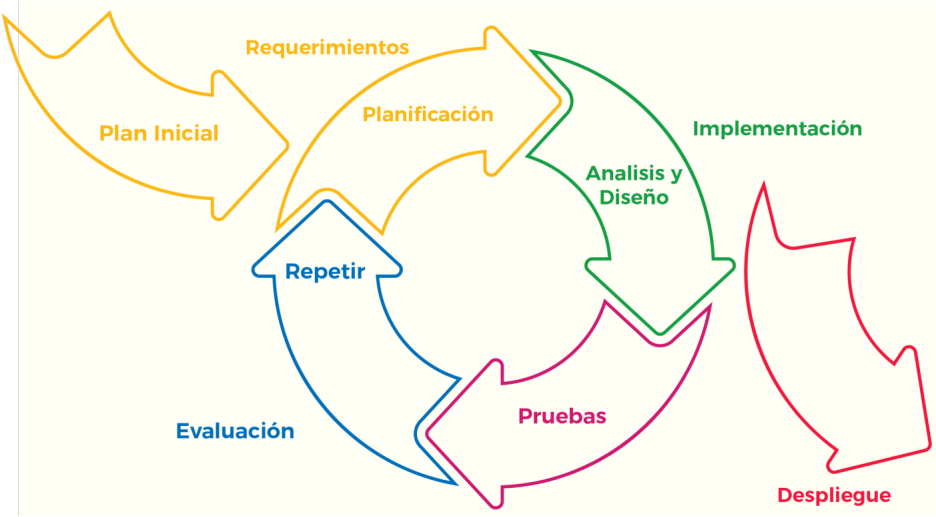


Figura 1.1: Pasos de una metodología iterativa e incremental. Extraída de [5]

1.4. Estructura de la memoria

La memoria de este proyecto se ha dividido en las siguientes partes:

- **Introducción:** Se expone el proyecto de una forma general, tratando su motivación, objetivos, metodología utilizada y como se estructura la memoria del TFG.
- **Contexto:** Se describe el contexto del TFG, es decir, el estado de la cuestión y los fundamentos teóricos.
- **Planificación y análisis de riesgos:** Descripción de las iteraciones que se van a llevar a cabo, estimación del tiempo que se va a utilizar para cada iteración, análisis de riesgos y descripción del entorno tecnológico utilizado.
- **Descripción de las iteraciones:** Descripción del trabajo realizado en cada iteración.
- **Estado final de la aplicación:** Análisis y diseño de la aplicación.
- **Pruebas:** Conjuntos de datos y pruebas llevadas a cabo durante el proyecto, así como los resultados obtenidos.
- **Conclusiones:** Resultados y conocimientos extraídos del proyecto realizado. Asimismo, muestra los posibles trabajos futuros que se podrían realizar para mejorar la funcionalidad de la aplicación.
- **Apéndices:** Documentos adicionales para la complementación del trabajo como acrónimos, manual de despliegue, manual de uso de la aplicación y contenido del TFG.
- **Bibliografía:** Referencias a artículos, páginas web, proyectos o libros que se han consultado a lo largo del desarrollo de la aplicación.

Capítulo 2

Contexto

2.1. Fundamentos teóricos

2.1.1. ¿Qué es la Estadística?

La palabra *Estadística*, derivada de *Estado* se utilizó por primera vez en Venecia y Florencia durante el Renacimiento para aludir a la obtención de datos de interés para el Estado. A lo largo de los siglos ha ido evolucionando convirtiéndose en lo que se encuentra hoy en día en todos los ámbitos.

Una definición del término *Estadística* sería «el arte para aprender a partir de datos». Esta palabra se relaciona con la recopilación de datos, su posterior descripción, análisis y como consiguiente, la extracción de conclusiones [6].

La Estadística se puede subdividir en dos grandes ramas: descriptiva e inferencial. A continuación, se van a describir cada una de ellas:

1. **Estadística descriptiva**: Su objetivo consiste en observar, resumir y describir un conjunto de datos con el fin de facilitar su interpretación [7]. Dentro de la Estadística descriptiva se pueden realizar análisis *univariantes*, que describen la distribución de una sola variable con medidas como la varianza, la asimetría o métodos gráficos; y *bivariantes y multivariantes*, que muestra la relación entre pares de variables (tablas de contingencia, diagramas de dispersión, correlación o covarianza, entre otros [8].

2. **Estadística inferencial**: Permite realizar inferencias sobre las características de una población a partir de una muestra de la misma [7]. La Estadística inferencial se subdivide a la vez en dos grandes subtipos: paramétrica y no paramétrica.

2.1.2. Test de significancia estadística

Como se ha dicho anteriormente, la estadística inferencial está formada por la estadística paramétrica y la no paramétrica. La principal diferencia entre ambas sería que las pruebas paramétricas son más robustas que las no paramétricas ya que la distribución subyacente de los datos es conocida, mientras que en la no paramétrica no se conoce la distribución de estos. Las pruebas paramétricas suelen tener más potencia estadística que las pruebas no paramétricas. Por lo tanto, hay más probabilidad de encontrar efectos que resulten significativos si estos existen.

En la Tabla 2.1 se clasifican las pruebas más importantes relacionadas con estadísticos paramétricos y no paramétricos que se han considerado en este TFG.

TIPO TEST	ESTADÍSTICO PARAMÉTRICO	ESTADÍSTICO NO PARAMÉTRICO
Comparación media observada con una teórica	Prueba t una muestra	Prueba de los rangos con signo Wilcoxon
Comparación dos medias independientes	Prueba t para dos muestras independientes	Prueba Mann-Whitney
Comparar varias medias independientes	ANOVA	Prueba de Kruskal-Wallis
Comparar dos medias observadas dependientes	Prueba t para dos muestras dependientes	Prueba Wilcoxon
Comparar varias medias observadas dependientes	ANOVA de medidas repetidas, modelos mixtos	-
Asociación entre dos variables cualitativas	Prueba Chi-cuadrado	-
Asociación entre dos variables cuantitativas	Correlación de Pearson	Correlación de Spearman

Tabla 2.1: Test realizados según el tipo de datos que se posean

2.1.3. Concordancia y correlación

Para observar la relación existente entre variables se usan términos como la correlación y la concordancia. La **correlación** en estadística se podría definir como la forma en la que se relacionan (fuerza y dirección) dos variables [9]. Es decir, existe una correlación positiva entre dos variables X e Y, si al aumentar los valores de X, los valores de Y aumentan y viceversa. Hay numerosas formas de medir esta correlación. Los coeficientes más utilizados son el *coeficiente de Pearson*, utilizado cuando ambas variables son continuas y numéricas y el *coeficiente de Sperman*, donde las variables pueden ser continuas o discretas.

En cuanto a la **concordancia**, se determinaría como la forma (hasta qué punto) coinciden dos observadores en su medición. En este caso se van a utilizar dos tipos:

- **Coeficiente kappa (k):** representa la relación de reducción del error entre la clasificación y la clasificación aleatoria de los datos. El coeficiente kappa (k) toma valores entre -1 y +1; mientras más

cercano a +1, mayor es el grado de concordancia. Este coeficiente se usa con datos categóricos.

- **Coeficiente de correlación de Kendall**: Se utiliza cuando la variable que se está evaluando es numérica.

2.1.4. Machine Learning

Machine Learning es un campo de la inteligencia artificial, el cual consiste en dejar que los algoritmos descubran patrones recurrentes en conjuntos de datos [10]. Estos datos deben poder almacenarse de forma digital y utilizarse para que el algoritmo aprenda.

Estos algoritmos aprenden de forma autónoma a realizar una tarea o hacer predicciones a partir de datos y mejorar su rendimiento con el tiempo. Una vez entrenado, el algoritmo podrá encontrar los patrones en nuevos datos.

Este tipo de aprendizaje se puede realizar de diferentes formas [11] dependiendo de los datos que haya disponibles y la tarea que se quiera realizar tal y como se puede ver en la figura 2.1:

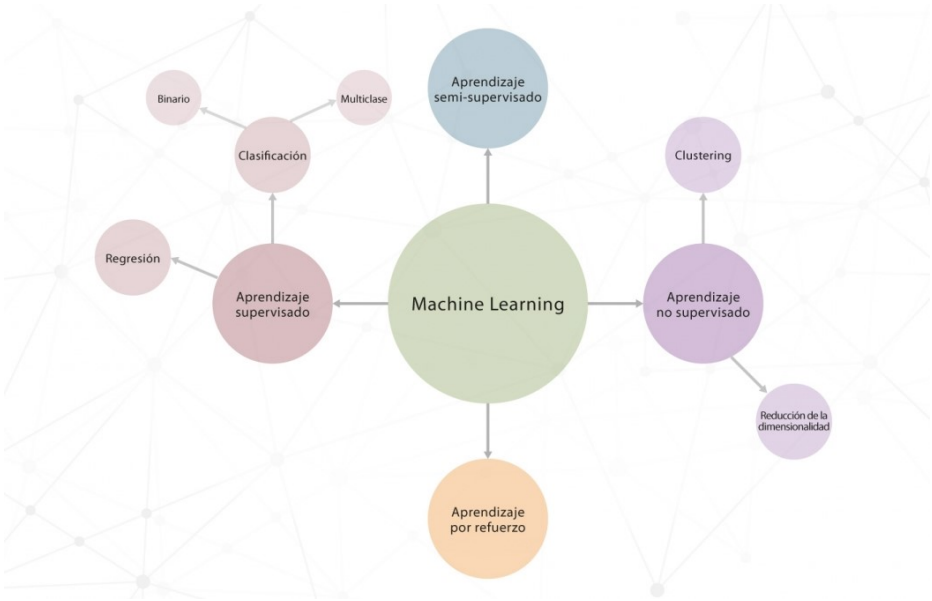


Figura 2.1: Tipos de Machine Learning [11]

2.1.5. Aprendizaje supervisado

El aprendizaje supervisado es una técnica para deducir una función a partir de datos de entrenamiento [12]. Es necesario que estos datos se encuentren etiquetados para poder decirle al modelo qué es lo que se desea que aprenda. El objetivo de este aprendizaje es crear una función capaz de predecir el valor correspondiente a cualquier objeto de entrada válida después de haber visto una serie de

ejemplos, los datos de entrenamiento. Para conseguirlo, es necesario generalizar mediante los datos que se poseen a situaciones nunca vistas anteriormente.

Los datos de entrenamiento están formados por pares de objetos (vectores) donde una componente son los datos de entrada y la otra los resultados del entrenamiento. La salida de la función realizada puede ser de dos tipos: un dato numérico (cómo el resultado de la **regresión**) o una etiqueta de clase (resultado de los problemas de **clasificación**).

El propósito de este aprendizaje es encontrar una función *g* dados unos puntos de la forma *x,g(x)* [12].

El conjunto de puntos es conocido y normalmente almacenado en una base de datos y está formado por una muestra de variables aleatorias independientes igualmente distribuidas con una distribución p de probabilidad no conocida.

Por otro lado, la función de pérdida *L* se representa como:

$$L : Y \times Y \to R^{\geq 0} \tag{2.1}$$

Donde *Y* es el dominio de *g*, y *L* es una función de mapas en un número no negativo *s*.

El riesgo asociado con una función *g* es la esperanza de la función de pérdida:

$$R(f) = \sum_i L(f(x_i), g(x_i))\ p(x_i) \tag{2.2}$$

Por último, el objetivo es encontrar una función *f** entre un número fijo de funciones para las que el riesgo *R(f*)* es mínimo.

Esta selección se conoce como *minimización empírica de riesgos* y la más común se calcula aproximando al riesgo real, mediante el riesgo empírico, como se puede ver en la fórmula (4.3):

$$\tilde{R}_n(f) = \frac{1}{n} \sum_{i=1}^{n} L(f(x_i), y_i) \tag{2.3}$$

Dependiendo del tipo de etiqueta, dentro del aprendizaje supervisado existen dos tipos de modelos: clasificación y regresión.

A continuación, se van a explicar los algoritmos supervisados implementados durante el proyecto tanto de clasificación (SVM y árboles de decisión) como de regresión (lineal o logística).

2.1.6. Máquinas de Vectores de Soporte

Las máquinas de vectores de soporte (SVM) son un conjunto de algoritmos de aprendizaje supervisado desarrollados por *Vladimir Vapnik*, los cuales implementan métodos para problemas de clasificación y de regresión [13].

Dado un conjunto de puntos del espacio (en forma de vector p-dimensional) en el que cada uno de ellos pertenece a una categoría, un algoritmo basado en SVM construye un modelo capaz de predecir si un punto nuevo (cuya categoría no es conocida) pertenece a una categoría o a la otra.

Por lo tanto, el algoritmo SVM busca un *hiperplano* que sea capaz de separar los puntos en una de las dos clases posibles y buscan que ese hiperplano tenga la máxima distancia (margen) con los puntos que estén más cerca de él mismo. Los puntos que estén a un lado del hiperplano se les clasificará en una categoría y los puntos que estén al otro lado se les asignará otra la categoría distinta.

Al vector formado por los puntos más cercanos al hiperplano se le llama *vector de soporte*.

Por ello, a los algoritmos SVM muchas veces se les puede llamar *clasificadores de margen máximo*.

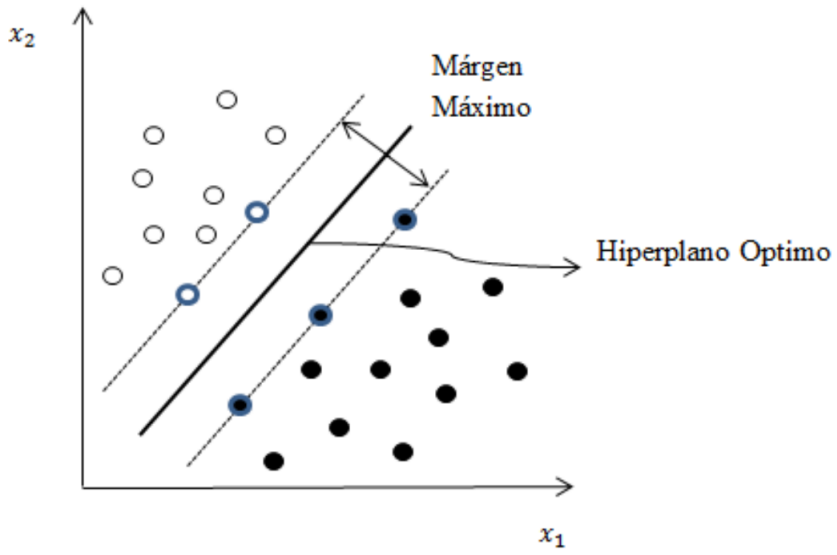


Figura 2.2: Ejemplo de Máquinas de Vectores de Soporte [14]

Esta separación de los datos se puede hacer mediante una recta, un plano recto o un hiperplano N-dimensional.

Normalmente los problemas no se encuentran en dos dimensiones, por lo que se utilizan funciones Kernel que proyectan la información a un espacio de características de mayor dimensión el cual aumenta la capacidad computacional de las máquinas de aprendizaje lineal.

Los tipos de funciones Kernel pueden ser:

- **Polinomial:**

$$K(x_i, x_j) = (x_i, x_j)^n \tag{2.4}$$

- **Perceptrón:**

$$K(x_i, x_j) = ||x_i, x_j|| \tag{2.5}$$

- **Gaussiano:** Función de base radial Gaussiana.

$$K(x_i, x_j) = e^{-\frac{||x_i, x_j||}{2 \cdot \sigma}} \tag{2.6}$$

- **Sigmoide:** Función que equivale a un modelo de perceptrón de dos capas de una red neuronal.

$$K(x_i, x_j) = \tanh(x_i \cdot {x_j}^{-\theta}) \tag{2.7}$$

2.1.7. Árboles de decisión

Estos métodos se han convertido en unos de los más utilizados hoy en día para la predicción y análisis de datos debido a su sencilla interpretación y representación, su capacidad de usar variables numéricas o categóricas sin que cumplan ningún tipo de supuesto paramétrico y su gran robustez frente a outliers, preprocesado de datos.

Un árbol de decisión es un algoritmo predictivo que representan y clasifican una serie de datos según unas condiciones propuestas que ocurren sucesivamente [15].

En este árbol cada nodo interno está etiquetado con una función de entrada. Los arcos procedentes de un nodo etiquetado con una característica están etiquetados con cada uno de los posibles valores de la característica. Cada hoja del árbol se marca con una clase o una distribución de probabilidad sobre las clases [16]. Tal y como se puede ver en ejemplo de la Figura 2.3 si la característica cumple el valor de uno de los arcos, se pasa al segundo nivel, así hasta llegar al resultado (pudiendo tener tantos niveles como el usuario desee).

Los dos árboles más utilizados en minería de datos son: regresión y clasificación.

El árbol de clasificación es el método que se ha implementado en este proyecto donde el resultado predicho es la clase a la que pertenecen los datos.

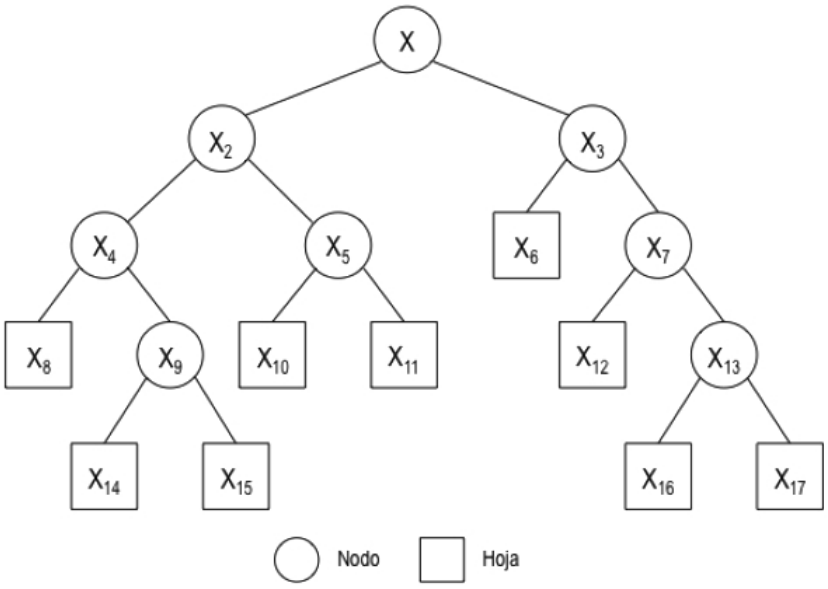


Figura 2.3: Ejemplo de Árbol de Decisión [17].

2.1.8. Regresión

El término regresión se propuso por primera vez en el estudio de variables antropométricas: comparando la altura de padres e hijos, donde resultó que los hijos cuyos padres tenían una estatura muy superior al valor medio, tendían a igualarse a este, mientras que aquellos cuyos padres eran muy bajos tendían a reducir su diferencia respecto a la estatura media; es decir, regresaban al promedio [18]. Existen varios tipos de regresión, y en este TFG se van a analizar dos: regresión lineal y regresión logística.

2.1.8.1. Regresión lineal

La regresión lineal se podría definir como la relación entre una variable dependiente *Y*, *m* variables independientes X_i y un término aleatorio ϵ, y proporciona al modelo su carácter estocástico [18].

Este modelo se podría expresar como:

$$Y = \beta_0 + \beta_1 X_1 + ... + \beta_m X_m + \varepsilon \tag{2.8}$$

Donde *Y* es la variable respuesta, las X_i son las variables explicativas o regresoras y los β_i se refieren a los coeficientes del modelo ajustado, que miden la influencia de las variables independientes sobre la variable dependiente.

Los valores ε_i se refieren al residuo e indican la bondad del ajuste realizado para cada punto. Se calculan de la siguiente forma:

$$e_i = y_i - \hat{y}_i \tag{2.9}$$

Además, el modelo de regresión clásico debe cumplir una serie de hipótesis para que sus resultados sean válidos:

- **Homocedasticidad:** La varianza de los errores del modelo se mantiene constante a lo largo del tiempo y los residuos siguen una igual dispersión.
- **Independencia de las variables regresoras:** No existencia de relación lineal entre los regresores.
- **Media Cero:** Para cada valor *X*, la perturbación tomará algunos valores negativos y otros positivos para que su media esperada final sea 0.

2.1.8.2. Regresión logística

La regresión logística es otro tipo de regresión de uso tanto predictivo como explicativo. Esta regresión resulta interesante cuando las variables independientes X_i (regresores) son variables numéricas (covariables) o categóricas (en cuyo caso hay que transformarlas a numéricas) y la variable respuesta *Y* es dicotómica, es decir, solamente es capaz de tomar dos valores [19].

Resumiendo, el objetivo principal de esta regresión es predecir la probabilidad de que ocurra o no cierto *'evento'* y determinar que variables aumentan o disminuyen la probabilidad de que ese evento ocurra.

2.1.9. Aprendizaje no supervisado

A diferencia del aprendizaje supervisado, en este tipo de aprendizaje los datos que se tienen de entrada son un conjunto de variables aleatorias donde no existe un conocimiento a priori. Estos algoritmos pueden clasificar conjunto de datos sin referencia a resultados etiquetados.

Por ello, este aprendizaje permite la realización de tareas de procesamiento más complejas que el supervisando. Sin embargo, puede ser más impredecible [20].

2.1.10. Clustering o análisis de conglomerados

El algoritmo de agrupamiento (clustering) es el procedimiento de particionar un conjunto de datos u objetos, en subclases denominadas **clusters**. Este proceso se realiza mediante diferentes criterios co-

mo distancia o similitud. Debido a que se está hablando de clasificación no supervisada, en el clustering no se conocen las etiquetas de las clases y puede que tampoco el número de grupos [21].

Se conocen dos grandes técnicas de agrupamiento de datos:

1. Clustering jerárquico: aglomeración o división.
2. Clustering no jerárquico: El número de grupos se establece con anterioridad y las observaciones se van asignando a cada grupo según la distancia a la que se encuentren y el número de grupos seleccionado. Un ejemplo sería el algoritmo **k-medias** (que se puede observar en la Figura 2.4, que es de los más utilizados y extendidos debido a su sencillez y eficacia.

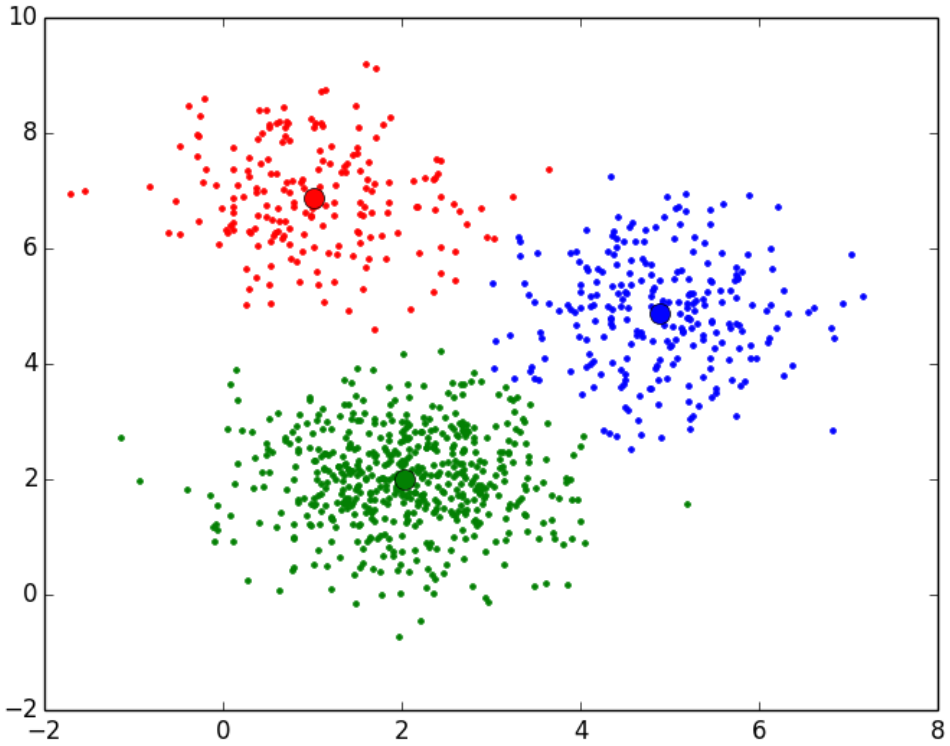


Figura 2.4: Ejemplo de algoritmo K-medias con 3 clusters (centros) [22].

Por último, para finalizar con los fundamentos teóricos del proyecto se va a exponer en que consiste la **selección de características** dentro de un conjunto de datos.

2.1.11. Selección de características

Este proceso se podría definir como la selección de las características más importantes y/o relevantes de un conjunto de datos para mejorar el rendimiento de los predictores, proporcionar predictores más rápidos y una mejor comprensión del proceso subyacente que generó los datos [23].

Los métodos más empleados son: los **métodos de filtro**, **métodos de envoltura** y **métodos integrados**.

Los *métodos de filtro* se emplean para el preprocesamiento de los datos y la selección de características y no dependen de ningún algoritmo de Machine Learning [23] tal y como se puede observar en la Figura 2.5. Algunos de estos métodos serían: ANOVA, Chi-Cuadrado, correlación, análisis discriminante lineal...

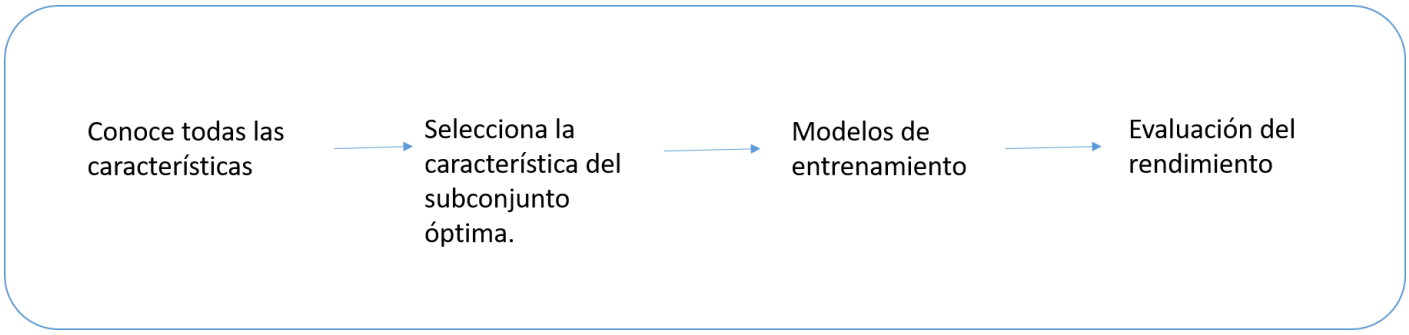


Figura 2.5: Descripción de los métodos de selección de características basados en filtro [24].

En cuanto a los *métodos de envoltura*, estos si necesitan un algoritmo de Machine Learning como criterio de evaluación. El objetivo es la búsqueda de la característica más adecuada para el algoritmo mejorando el rendimiento de este. Por lo tanto, estos métodos analizan *problemas de búsqueda* con un coste computacional normalmente muy elevado. En la Figura 2.6 se puede observar la descripción de estos métodos por pasos.

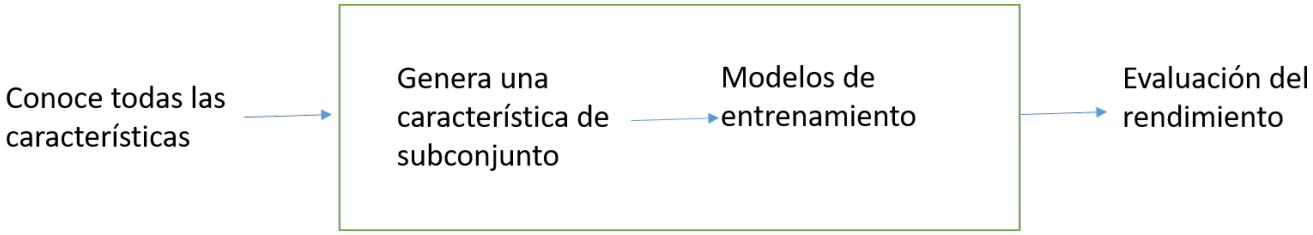


Figura 2.6: Descripción de los métodos de selección de características basados en envoltura [24].

Los Métodos de Envoltura implementados en la aplicación han sido:

- El algoritmo Random Forest, que es una combinación de predictores de árboles donde todos siguen la misma distribución [25].
- Selección hacia delante (forward selection) que *añade* características en función del resultado de la puntuación de la validación cruzada.
- Selección hacia atrás (backward selection) que *elimina* características en función del resultado de la puntuación de la validación cruzada.

Por último, los *métodos integrados* combinan ambos métodos mencionados anteriormente. Su implementación se lleva a cabo con algoritmos que tienen sus propios métodos de selección de características incorporados. La regresión LASSO o RIDGE, son ejemplos de estos métodos que reducen el sobreajuste con funciones de penalización incorporadas [23].

2.2. Estado de la cuestión

2.2.1. ¿Qué es Python?

Python es un lenguaje de programación de *alto nivel* interpretado cuyo primer propósito es la legibilidad de código [26]. Fue creado en 1991 por Guido Van Rossum el Centro para las Matemáticas y la Informática (CWI), en los Países Bajos, como un sucesor del lenguaje de programación ABC, capaz de manejar excepciones e interactuar con el sistema operativo Amoeba. Hoy en día se encuentra dirigido por **Python Software Fundation** [27].

En los últimos años su empleo ha ido creciendo a grandes pasos y hoy en día (2022) es uno de los lenguajes más utilizados tanto para el desarrollo software como para cualquier análisis *Big Data*, debido a su facilidad de procesamiento de datos. En la Figura 2.7 se puede observar su logotipo.

Algunas de las ventajas de Python serían [28]:

- Como es un **lenguaje multiparadigma**, permite varios estilos de programación: funcional, orientada a objetos e imperativa.
- **Amplia variedad de bibliotecas** y frameworks, con una gran cantidad de módulos ampliables.
- **Portabilidad**, ya que es compatible con cualquier Sistema operativo.
- **Gratuito y de código abierto**, por lo que se puede usar y distribuir de forma libre.
- **Fácil aprendizaje**, ya que debido a su sencilla sintaxis permite escribir programas con una alta funcionalidad en pocas líneas de código.

Figura 2.7: Logotipo de Python [29]

2.2.2. Otros softwares estadísticos

2.2.2.1. R

R es un lenguaje de programación utilizado especialmente para el análisis de datos [30]. Este software está formado por varias herramientas estadísticas y permite también la creación de gráficos. En la Figura 2.8 se puede observar su logotipo.

Fue desarrollado por Robert Gentleman y Ross Ihaka del Departamento de Estadística de la Universidad de Auckland en 1993. Hoy en día, R Development Core Team se encarga de su mantenimiento y desarrollo[30].

Además, R cuenta con entorno de desarrollo integrado (informático-estadístico) de fácil manejo, **R-studio**, y disponible para Windows, Linux y Mac [31].

Algunas de las ventajas de R son [32]:

- Se puede descargar de forma gratuita.
- Como se ha dicho anteriormente, funciona con cualquier sistema operativo.
- Reduce el tiempo de ejecución para grandes conjuntos de datos.
- Permite incluir gráficos y sacar conclusiones de forma rápida.

Y algunas de sus desventajas [32]:

- En comparación con otros lenguajes de programación (como Java o Python) es mucho más lento.
- No tiene una compatibilidad directa con otros lenguajes de programación.
- Es bastante vulnerable a ataques cibernéticos, ya que tiene poca seguridad.

Figura 2.8: Logotipo de R [33]

2.2.2.2. SPSS

SPSS es un programa estadístico utilizado para la investigación en ciencias sociales y ciencias aplicadas, y en la investigación de mercado [34]. Tiene una alta capacidad de trabajar con bases de

datos de gran tamaño y consta de una interfaz muy sencilla para cualquier usuario, lo que le hace uno de los softwares más conocidos hoy en día. En la Figura 2.9 se puede observar su logotipo.

Fue creado en 1968 por Norman H. Nie, C. Hadlai (Tex) Hull y Dale H. Bent y en 2009 fue adquirido por IBM por 1200 millones de dólares [34].

Las ventajas de este software serían [35]:

- Existe una amplia variedad de análisis estadísticos tanto simples como avanzados que se pueden realizar.
- La interfaz es sencilla y de fácil aprendizaje.
- Permite trabajar con bases de datos grandes.
- Compatibilidad con muchos otros programas de análisis de datos.

Y algunas de sus desventajas [35]:

- La mayoría de los resultados que extrae tienen un nivel de información excesivo que podría llegar a confundir al usuario.
- No es gratuito y la obtención de las licencias puede tener un alto coste.

Figura 2.9: Logotipo de SPSS [36]

2.2.2.3. SAS

SAS es un software estadístico utilizado para la gestión de datos, la investigación y el análisis predictivo . Está formado por varios módulos capaces de desarrollar diferentes procesos. También proporciona una interfaz de usuario gráfica para no expertos [37]. Su logotipo se puede ver en la Figura 2.10.

Fue creado entre 1966 y 1976 en la Universidad de Carolina del Norte.

Las ventajas de este software serían [38]:

- SAS tiene una sintaxis de un código que es sencilla de aprender y su código se puede depurar fácilmente.
- Tiene una alta funcionalidad para trabajar con grandes bases de datos

- Existe una alta asistencia al usuario dentro de este software.

Y algunas de sus desventajas [38]:

- No es gratuito, y su costo no es accesible para todos los usuarios.
- Es complicado realizar gráficos detallados sin una gran cantidad de código.
- No es de código abierto, es decir, trabaja en un dominio cerrado y nuevos usuarios no pueden editar código.

Figura 2.10: Logotipo de SPSS [39]

2.2.3. Otros trabajos similares

En esta sección se va a comparar la tarea desempeñada en este proyecto con otros trabajos similares, ya sean Trabajos de Fin de Grado, repositorios de GitLab o artículos publicados.

El primer TFG comparado [40], implementa una aplicación que realiza análisis estadísticos sobre un conjunto de datos (concentraciones de sustancias) con sus respectivos gráficos interactivos. Al igual que en este trabajo, se ha decidido usar Python, ya que en comparación con otros lenguajes es mucho más estable, claro y sencillo para el análisis de datos. Este trabajo se encuentra más enfocado hacia un análisis sencillo y descriptivo de los datos, como histogramas, correlaciones o detección de outliers. La aplicación propuesta en nuestro TFG, consta de un análisis descriptivo de los datos algo más sencillo que el trabajo de Alberto Sánchez Romero [40]. En el TFG de Alberto se incluyen gráficos y resultados mucho más específicos que en nuestra aplicación no se han implementado, ya que también se centra en clasificación automática e inferencial.

Otro trabajo similar a nuestro TFG ha sido realizado por Artur de Osset Greño [41]. Este proyecto desarrolla una sencilla interfaz que, introducidos un conjunto de datos, realiza diferentes análisis estadísticos. En este caso, solo lleva a cabo análisis descriptivos, análisis clustering y predicciones. El trabajo ejecutado en nuestra aplicación, aparte de implementar todo lo anterior pone en marcha otros tipos de aprendizaje, como el algoritmo SVM o los árboles de decisión. Sin embargo, no se ha llevado a la práctica ninguna interfaz como tal, aunque se puede llegar a usar las funcionalidades de Python (JupyterLab y Jupyter Notebook) como esta.

Otra librería comparada tiene su repositorio en GitHub [42]. En esta biblioteca, se realizan test estadísticos paramétricos (ANOVA, MANOVA), no paramétricos (Friedman, Kruskal Wallis, MannWhitney...),

tablas de contingencia y test post-hoc. Este trabajo se centra especialmente en la estadística inferencial, ejecutando más test e implementándolos de forma más específica que los test llevados a cabo en nuestra aplicación. Sin embargo, han aumentado las funcionalidades de nuestro TFG añadiendo ANOVA dentro de grupos (*within*) y mixto, ya que el repositorio [42], solo realiza ANOVA entre grupos. También se ha decidido y diseñado la realización de regresiones lineales y logísticas, algo que se echa en falta en esta librería comparada.

Si se visualiza otro repositorio en GitHub creado por Chris Fonnesbeck [43], se puede observar que se ha llevado a cabo un desarrollo similar de nuestro proyecto. Ambas bibliotecas utilizan los datos en formato *DataFrame*, usan métodos para el tratado de datos faltantes y fusión de ficheros. Sin embargo, este repositorio [43] se encuentra más enfocado al tratado y visualización de datos (histogramas, *scatterplots, trellis plot*...) que nuestra librería también implementa, pero no en tanta profundidad.

Finalmente, se ha comparado el proyecto aquí desarrollado con un último repositorio creado por *@suneelpatel* [44]. Este repositorio se centra en desarrollar archivos Python que implementen funciones relacionadas con describir estadísticamente los datos y con temas como el teorema de Bayes y las distribuciones de probabilidad. Nuestro TFG tiene un archivo que utiliza estadísticos descriptivos para organizar y presentar los datos en el que también se incluyen gráficos. No obstante, no profundiza tanto en la organización de los datos, ya que se centra en la obtención de clasificadores y en análisis inferencial.

Capítulo 3

Planificación y análisis de riesgos

3.1. Planificación inicial

3.1.1. Distribución temporal

Este trabajo se lleva a cabo durante parte del primer cuatrimestre y parte del segundo en el curso 2021-2022. La primera iteración comenzará el día 16 de noviembre de 2021 y la fecha de finalización prevista es el 1 de marzo de 2022. El estudiante trabajará diariamente con una carga aproximada diaria de 4 horas. Normalmente un trabajo de Fin de Grado requiere 300 horas como mínimo de trabajo, por lo que se tendría un total de 85 días. Terminar el trabajo en dos meses y medio sería una actitud muy optimista y muy poco realista, ya que pueden surgir muchos inconvenientes y días en los que el alumno no pueda realizar esas 4 horas estipuladas.

Si no se pudiesen cubrir las horas acordadas, se podrán realizar horas extras los fines de semana para paliar la falta de tiempo. Por lo tanto, se añadirían 60 horas a mayores para la realización del proyecto. Dicho esto, serían un total de 360 horas, es decir unos 100 días de trabajo, alrededor de 3 meses y medio.

Cabe destacar que se llevarán a cabo reuniones con los tutores aproximadamente cada 2 semanas, donde se plantean los nuevos objetivos a realizar y se aclaran las dudas que hayan podido surgir.

Se ha dividido el trabajo en iteraciones y ahora se va a mostrar la propuesta inicial:

- **Primera iteración (2 semanas, 30 horas):** Se realizará la primera reunión con los tutores del TFG, en la cual los clientes expondrán en que va a consistir en proyecto a rasgos generales y la primera entrega para el alumno. En esta iteración, se va a efectuar la planificación del trabajo y un análisis de los posibles riesgos que pudiesen surgir a lo largo de él. También, se llevarán a cabo varios tutoriales de Python y varias pruebas con el entorno de Git, para integrarse de forma adecuada en el lenguaje y la plataforma, respectivamente.

- **Segunda iteración (3 semanas, 80 horas):** A lo largo de esta iteración, se van a realizar las primeras pruebas propias con el entorno de desarrollo, para que el alumno vaya cogiendo fluidez, así como la definición de requisitos del proyecto. Por último, se efectuará un análisis del estado del arte.

- **Tercera iteración (4 semanas, 110 horas):** Durante esta iteración, el desarrollador tendrá más de una reunión con los tutores, ya que es un periodo largo de trabajo. Se llevará a cabo la implementación de los requisitos definidos en la segunda iteración y las clases y funciones correspondientes en Python. Asimismo, se realizará una batería de pruebas del proyecto.

- **Cuarta iteración (3 semanas, 80 horas):** A lo largo de este periodo de tiempo, el alumno implementará los gráficos asociados a las implementaciones anteriores.

- **Quinta iteración (2 semanas, 50 horas):** Por último, se completará el proyecto, y se llevará a cabo la memoria de este, completándola y añadiendo lo necesario. Al final de la iteración, el trabajo debe estar finalizado.

La Figura 3.1 se expone la distribución del número de semanas que se esperan utilizar para cada iteración, y en la Figura 3.2 el número de horas que se prevén utilizar en cada iteración.

De color rosa, se exponen las semanas/horas esperadas para cada iteración y de color gris, las semanas/horas acumuladas.

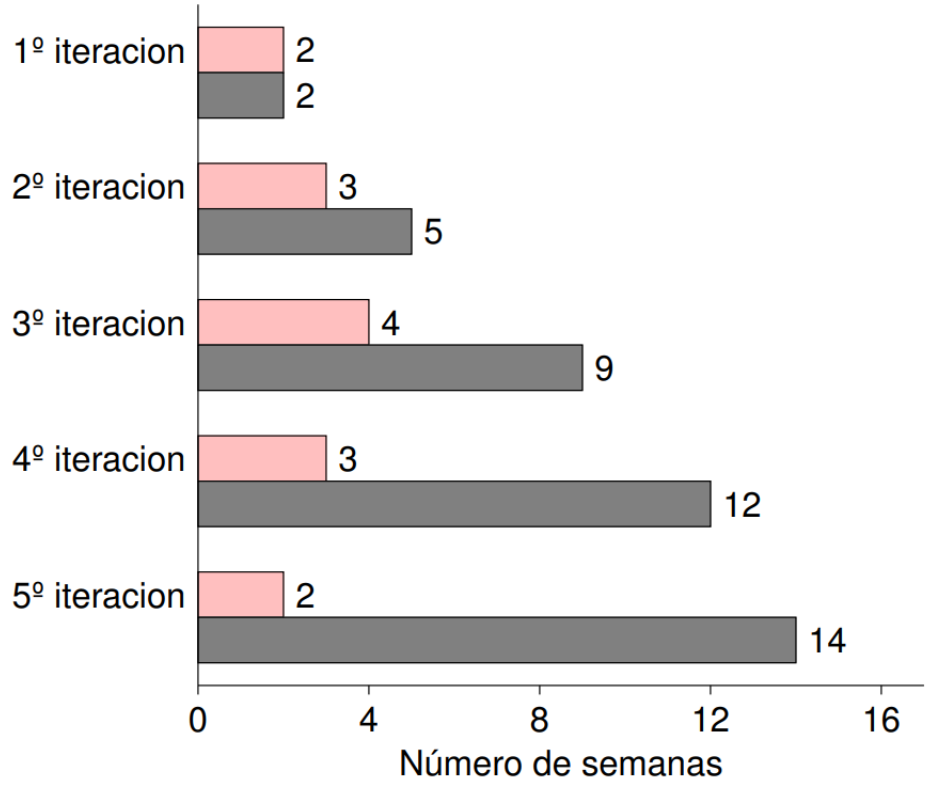


Figura 3.1: Distribución del número de semanas en cada iteración.

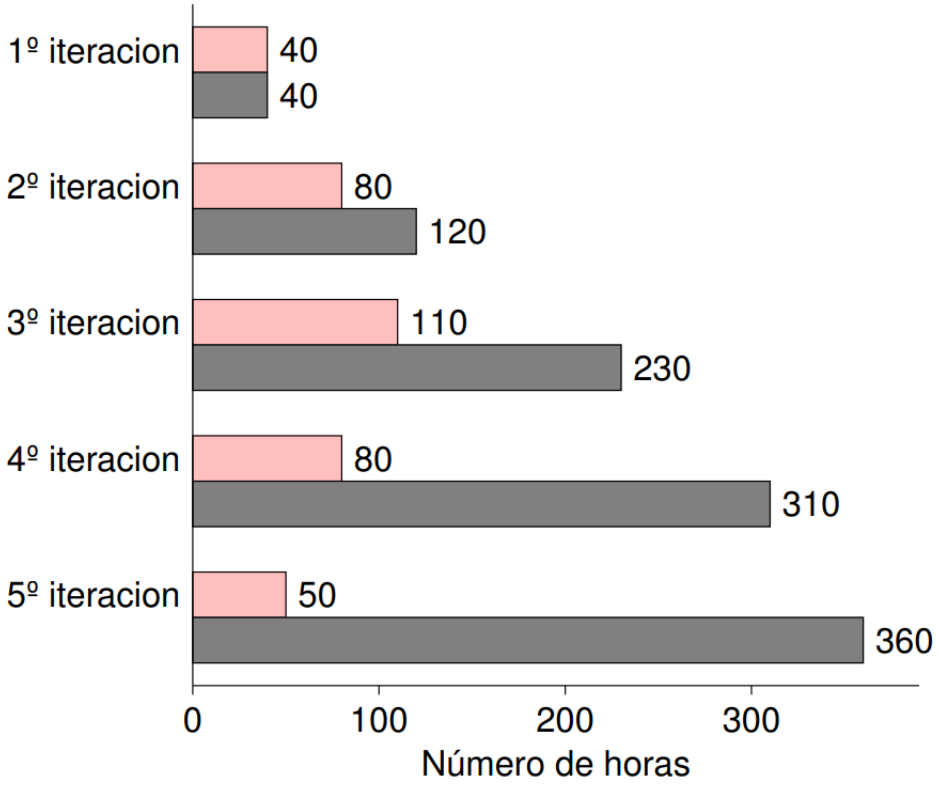


Figura 3.2: Distribución del número de horas en cada iteración.

3.1.2. Análisis de riesgos

En esta parte del documento, se van a presentar los diferentes riesgos que podrían surgir durante el desarrollo del proyecto. Estos riesgos se muestran en una tabla con la correspondiente probabilidad de que ocurran, una pequeña descripción del riesgo y el retraso de días que acarrearía si ese problema llegase a ocurrir.

Además de los riesgos, se ha llevado a cabo un plan de prevención para reducir la probabilidad de cada riesgo y un plan de acción por si el riesgo llegase a producirse.

La exposición al riesgo que aparece en la Tabla 3.2 sería el tiempo promedio perdido debido a cada uno de los riesgos. Es el resultado de la multiplicación entre la probabilidad de que el riesgo ocurra y el retraso (Tabla 3.1).

ID	Riesgo	Probabilidad	Descripción	Retraso (días)
R01	Suspender la asignatura de Programación de Aplicaciones Gráficas en la convocatoria ordinaria	0.1	La asignatura se ha suspendido y hay que dedicar un tiempo extra para la recuperación en la convocatoria extraordinaria	16
R02	Suspender la asignatura de Profesión y Sociedad en la convocatoria ordinaria	0.01	La asignatura se ha suspendido y hay que dedicar un tiempo extra para la recuperación en la convocatoria extraordinaria	10
R03	Suspender la asignatura de Programación de Aplicaciones Gráficas en la convocatoria extraordinaria	0.05	La asignatura se ha suspendido y hay que realizarla el siguiente cuatrimestre del próximo año	100
R04	Suspender la asignatura de Profesión y Sociedad en la convocatoria extraordinaria	0.0001	La asignatura se ha suspendido y hay que realizarla el siguiente cuatrimestre del próximo año	100
R05	Prácticas en empresa	0.6	Comienzo de las prácticas en empresa y sobrecarga de trabajo	20
R06	Retraso con el TFG de estadística	0.4	El proyecto final del grado en Estadística se ha retrasado por algún motivo y habrá que echarle más horas	30
R07	Contagio por COVID	0.08	El alumno se contagia por COVID-19 y debe guardar reposo	4
R08	Enfermedad	0.3	El alumno vuelve a retomar la enfermedad que le tuvo en reposo durante un mes	30
R09	Falta de experiencia en el desarrollo de un trabajo largo	0.4	Primera toma de contacto del alumno con un trabajo de larga extensión y gran planificación que puede llevar a no hacerlo de manera correcta	15
R10	Pérdida de información durante el desarrollo del proyecto	0.2	La pérdida de datos o partes del trabajo conllevaría a un gran retraso en el tiempo de entrega	25
R11	Falta de conocimientos sobre algún tema/software utilizado	0.5	El alumno desconoce el uso de algún programa o conocimientos teóricos utilizados durante el trabajo	15

Tabla 3.1: Riesgos posibles durante el desarrollo del proyecto. ID: identificación del riesgo

ID	Riesgo	Exposición	Plan de Prevención	Plan de acción
R01	Suspender la asignatura de Programación de Aplicaciones Gráficas en la convocatoria ordinaria	1.6	Se realizarán todos los trabajos semanales y se estudiaría día a día	Se pospondrá la entrega del trabajo unas semanas
R02	Suspender la asignatura de Profesión y Sociedad en la convocatoria ordinaria	0.1	Se realizarán todos los trabajos semanales de la asignatura	Se pospondrá la entrega del trabajo unas semanas
R03	Suspender la asignatura de Programación de Aplicaciones Gráficas en la convocatoria extraordinaria	5	Se estudiará una mayor cantidad de tiempo desde que se sepa la nota en la convocatoria ordinaria	Se pospondrá la entrega del proyecto a el primer cuatrimestre de 2022
R04	Suspender la asignatura de Profesión y Sociedad en la convocatoria extraordinaria	0.01	Se estudiará una mayor cantidad de tiempo desde que se sepa la nota en la convocatoria ordinaria	Se pospondrá la entrega del proyecto a el primer cuatrimestre de 2022
R05	Prácticas en empresa	12	Se intentará compaginar las horas de trabajo y estudia y organizarse adecuadamente	Se usarán horas del fin de semana para paliar el retraso
R06	Retraso con el TFG de estadística	12	Cumplir con el plan establecido de entrega del proyecto de la carrera de Estadística	Se retrasará la entrega del trabajo unas semanas
R07	Contagio por COVID	0.32	Respetar las medidas sanitarias	Se usarán horas del fin de semana para paliar el retraso
R08	Enfermedad	9	-	Se retrasará la entrega de trabajo unas semanas
R09	Falta de experiencia en el desarrollo de un trabajo largo	6	Intentar llevar una planificación y un cumplimiento de los plazos	Se usarán horas del fin de semana y a mayores entre semana para recuperar el retraso
R10	Pérdida de información durante el desarrollo del proyecto	5	Se utilizará Gitlab para evitarlo, así como la subida de los datos a la nube	Se utilizarán las copias que haya guardadas aunque se haya perdido algo de información
R11	Falta de conocimientos sobre algún tema/software utilizado	7.5	Estudiar tutoriales y manuales de uso del software	Se usarán horas del fin de semana para evitar un retraso

Tabla 3.2: Planes de prevención y acción del proyecto. ID: identificación del riesgo

3.2. Entorno tecnológico

3.2.1. Herramientas utilizadas

En la siguiente sección se va a describir las herramientas utilizadas durante el proyecto, y la función de cada una de ellas, así como las características que las componen.

- **Anaconda Navigator [45]**: Suite de código abierto utilizada principalmente para el desarrollo de código en Python. Esta interfaz gráfica de usuario (GUI) permite iniciar aplicaciones y administrar fácilmente paquetes, entornos y canales sin la necesidad de usar comandos. En este trabajo se ha utilizado la versión 4.11.0. El menú principal de esta GUI se puede observar en la Figura B.1.
- **Astah UML [46]:** Herramienta de desarrollo y modelado UML originalmente creada por la compañía japonesa *Change Visio*.

  Las herramientas que la componen permiten visualizar ideas y diseños de software. Asimismo, es posible la creación diagramas de forma rápida y eficientes que generen una comprensión clara entre los equipos. Los diagramas pueden ser desde diagramas UML, ER, de flujo de datos hasta mapas mentales. Puede ser utilizado tanto por estudiantes y facultades universitarias hasta por equipos de desarrollo e individualmente.
- **Chrome 96.0.4664.110 [47]:** Navegador web creado por *Google* y usado para la búsqueda de información.
- **Excel 365 [48]:** Hojas de cálculos para crear, editar, ver o guardar los datos del proyecto. También permite crear hojas de datos online entre varios usuarios.
- **GitLab (versión Web) [49]:** Repositorio para el control de versiones y desarrollo de software colaborativo basado en *Git*. El sistema también ofrece alojamiento de wikis, gestión de repositorios y un sistema de seguimiento de errores.
- **Google Drive (versión Web) [50]:** Servicio para el almacenamiento y compartición de archivos relacionado con el proyecto, creado por la empresa *Google*.
- **LucidChart (versión Web) [51]:** Herramienta utilizada para la creación de diagramas, esquemas, prototipos de software... todo ello de forma online y con la posibilidad de que sea de forma colaborativa.
- **Mozilla Firefox 99.0.1 [52]:** Navegador web creado por *Corporación Mozilla* desarrollado para diferentes plataformas.
- **Overleaf (versión Web) [53]:** Editor de texto online para escribir el documento del proyecto.
- **Telegram 3.3 [54]:** Plataforma de mensajería enfocada a la mensajería instantánea y el envío de archivos. Utilizada para la comunicación con los tutores.

3.2.2. Entorno de desarrollo

En primer lugar, en la Tabla 3.3 se muestra el equipo utilizado para el desarrollo del proyecto y de las pruebas.

Ordenador portátil	
HP Pavilion Laptop 14-bk0xx	
Hardware	
Procesador	Intel(R) Core(TM) i5-7200U a 2.50 GHz
RAM	8,00 GB
SSD	128 GB
HDD	1 TB
GPU	NVIDIA GForce 940 Mx
Software	
Sistema Operativo	Microsoft Windows 10 Home
Arquitectura	64 bits

Tabla 3.3: Características del ordenador utilizado durante el desarrollo del trabajo.

Para la creación de código de la librería en Python, se ha usado el entorno de programación *Anaconda* [45]. **Anaconda** se trata de un software libre que proporciona un conjunto de herramientas orientadas a la investigación, la ciencia y el análisis de datos. Al instalar Anaconda, se pueden utilizar diferentes entornos de desarrollo integrado (IDE) con acceso a lenguajes como R o Python [55] y a sus respectivas bibliotecas. Además, Anaconda mantiene al día las actualizaciones de todas las bibliotecas. Para ver más acerca de la instalación de anaconda consultar el Apéndice B.

Dentro de este entorno, se puede encontrar desde Jupyter Lab, Jupyter Notebook y Spyder (herramientas utilizadas para escribir el código en Python y poder ejecutarlo directamente) hasta Visual Studio Code y R-Studio.

Jupyter Notebook

Jupyter Notebook es un IDE que utilizar el navegador web predeterminado de tu PC. Está formado por varios bloques donde se puede escribir código y ejecutar cada bloque por separado, lo que hace que sea muy cómodo de usar y de presentar código, ya que estéticamente es muy agradable. Es el más sencillo de usar y es recomendable empezar por él si nunca se ha usado Anaconda. (Figura 3.3a)

JupyterLab

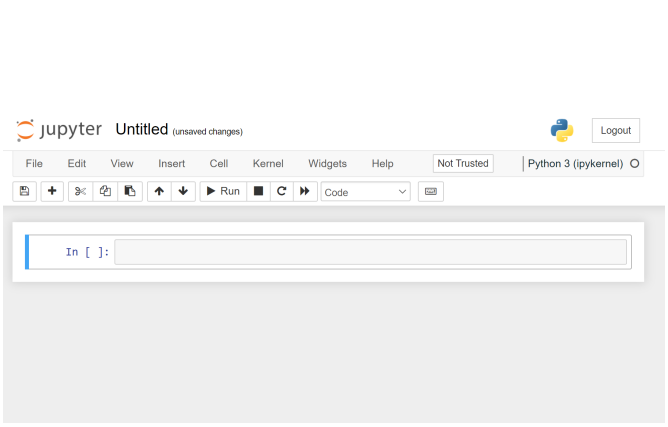
Es una extensión de Jupyter Notebook, ya que además de portar todas las funcionalidades de este le añade una terminal de línea de comandos, una consola de código y un editor de texto (Figura 3.3b).

Spyder IDE

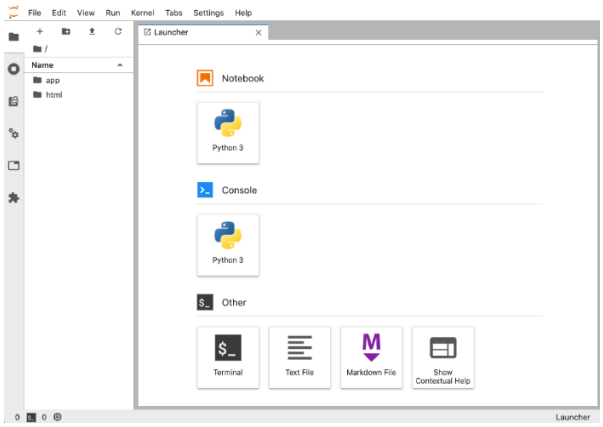
Contiene un editor donde modificar scripts, un explorador de variables, una consola Python y herramientas para la depuración de código (muy útiles para detectar errores). La interfaz se puede persona-

lizar y elegir la disposición de los paneles, el diseño o el color del IDE (Figura 3.3c).

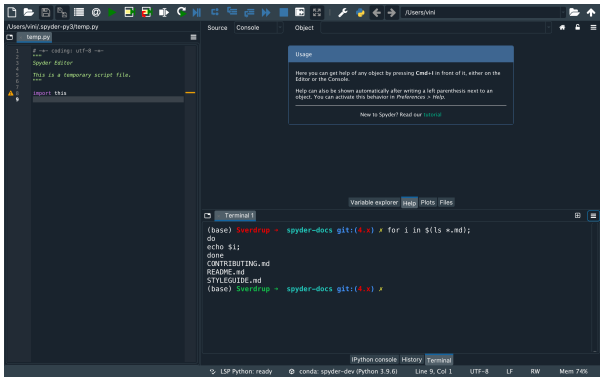
Estos tres editores se pueden ver en la Figura 3.3:



(a) Jupyter Notebook



(b) JupyterLab



(c) Spyder

Figura 3.3: Editores de Anaconda

3.3. Estimación de costes

En esta sección se va a detallar la estimación de los costes humanos y técnicos del proyecto. En todas las cantidades se encuentra incluido el IVA.

3.3.1. Salario del trabajador

Para hacer la estimación del salario de la persona que ha realizado el TFG se va a comparar con el salario de un programador Junior que realiza un tarea similar. El salario de programador promedio en España es de 22.495€/año o 11,54€/hora [56]. Por lo tanto, con un total de 420 horas de trabajo, el coste total del salario sería de 4.846,8 €.

3.3.2. Costes indirectos aplicados al espacio de trabajo

El trabajo se ha desarrollado en el domicilio del trabajador con un modelo no presencial. Por lo tanto, se tendrán en cuenta los gastos de alquiler y de abastecimiento de la residencia. En la Tabla 3.4 se puede ver la estimación de estos costes:

Servicio	**Coste mensual**	**Nº meses**	**Jornada**	**Coste total**
Luz y gas	130 €	7	2 h/día	76 €
Agua	15 €	7	2 h/día	9 €
Internet	50 €	7	2 h/día	30 €
Alquiler	300 €	7	2 h/día	175 €
Total				290 €

Tabla 3.4: Estimación de los costes indirectos del espacio de trabajo

3.3.3. Costes del hardware

Para estimar los costes del hardware, se ha observado el coste del portátil en el momento de la compra, el número de horas de uso totales y en el proyecto y, el coste aproximado que todo ello habría acarreado. Se pueden ver en la Tabla 3.5.

Costes del hardware				
Dispositivo	Costes de compra	Horas totales de uso del dispositivo	Horas utilizadas en el proyecto	Coste total
Portátil	750 €	50000 h	420 h	6,3 €

Tabla 3.5: Estimación de los costes de hardware del proyecto

3.3.4. Costes de software y servicios

Se han utilizado algunas herramientas gratuitas como Anaconda Navigator, Telegram o Chrome y algunas de pago. Tanto Astah como GitLab son gratuitas por pertenecer a la Escuela de Ingeniería Informática. A continuación se muestran los precios si el proyecto se hubiese realizado en un entorno profesional:

- Astah UML vale 4,99 €/ mes. Solo se usó durante un mes, en la tercera iteración.
- El repositorio de GitLab tiene un coste de 4 € al mes y se ha usado durante 7 meses, con un total de 28 €.
- El programa LucidChart cuesta 7,95 € /mes. Solamente se usó durante un mes, para describir la estructura del proyecto.

En la siguiente Tabla 3.6 se pueden observar los costes de software y el total:

Costes del software y servicios	
Software	***Costes***
Astah	4,99 €
GitLab	28 €
LucidChart	7,95 €
Total	***40,94 €***

Tabla 3.6: Estimación de costes de software y de servicios durante el desarrollo del proyecto

3.3.5. Costes totales

En la Tabla 3.7 se recopilan todos los costes estimados anteriormente y se calcula el total:

Costes totales	
Salario del trabajador	4.846,8 €
Espacio de trabajo	290 €
Hardware	6,3 €
Software	40,94 €
Total	**5.184,1 €**

Tabla 3.7: Estimación de costes totales del proyecto

3.4. Planificación final

El proyecto no ha cumplido la planificación propuesta inicialmente. Esta organización del trabajo fue muy optimista, ya que el alumno pensaba que no se daría ninguno de los riesgos estimados. El primer problema que surgió fue el contagio por COVID del alumno en enero (R07) con un aislamiento en cama de una semana. Esto aumentó en 1 semana el desarrollo de la segunda iteración con una duración de 110 horas.

Otro importante problema que apareció durante el avance del proyecto fue el comienzo de las prácticas del grado en Estadística en el Instituto Tecnológico Agrario de Castilla y León (ITACyL) (R05). En un primer momento, el alumno estimó que las prácticas no retrasarían el desarrollo del trabajo, ya que se creyó que el alumno iría 4 horas al día con un modelo semipresencial. Sin embargo, el contrato de prácticas con el Instituto fue de 8 horas al día yendo presencialmente todos los días, tardando 1 hora en ir y otra en volver. Esto sucedió durante la tercera iteración aumentando el tiempo del proyecto en cuatro semanas.

Uno de los riesgos que se produjo fue R06 (retraso con el TFG de Estadística). Esto fue debido a que el tema del TFG se cambió en el momento de finalización de las prácticas, ya que se estableció un convenio de trabajo con el ITACyL. Esto ocurrió durante la tercera y cuarta iteración aumentando en dos semanas el trabajo en ambas. Además, hubo problemas con la compilación del código y se añadieron nuevos requisitos y funcionalidades interesantes que aumentaron la carga de trabajo estipulada.

Por último, se tuvieron problemas con softwares no utilizados (R11), como LaTeX. Ha sido la primera vez que el alumno ha realizado un trabajo de larga extensión en un software que no dominaba, por lo que el desarrollo de la iteración 5 aumentó en 4 semanas, ya que a este riesgo se unió la falta de experiencia en el desarrollo de un trabajo largo (R09).

Analizando el proyecto de forma global, se puede deducir que se realizó una planificación demasiado optimista, subestimando los softwares utilizados.

Finalmente, la distribución total de semanas del proyecto quedaría de la siguiente forma:

- **1º iteración: 2 semanas**
- **2º iteración: 4 semanas**
- **3º iteración: 10 semanas**
- **4º iteración: 5 semanas**
- **5º iteración: 6 semanas**

Además, se han realizado reuniones bisemanales con los tutores para ir analizando el desarrollo del proyecto. En total se han realizado 10 reuniones por videoconferencia, que han sido apuntadas en una hoja de Excel. En este archivo, se ha anotado el avance hasta la fecha y las tareas realizadas por el alumno.

La duración total ha sido 27 semanas en contraposición a las 14 semanas establecidas al principio del proyecto. En la Figura 3.4 se muestra la duración real del proyecto en morado y en color caqui las horas acumuladas. Como se ha dicho anteriormente, de color rosa, se exponen las semanas/horas esperadas para cada iteración y de color gris, las semanas/horas acumuladas.

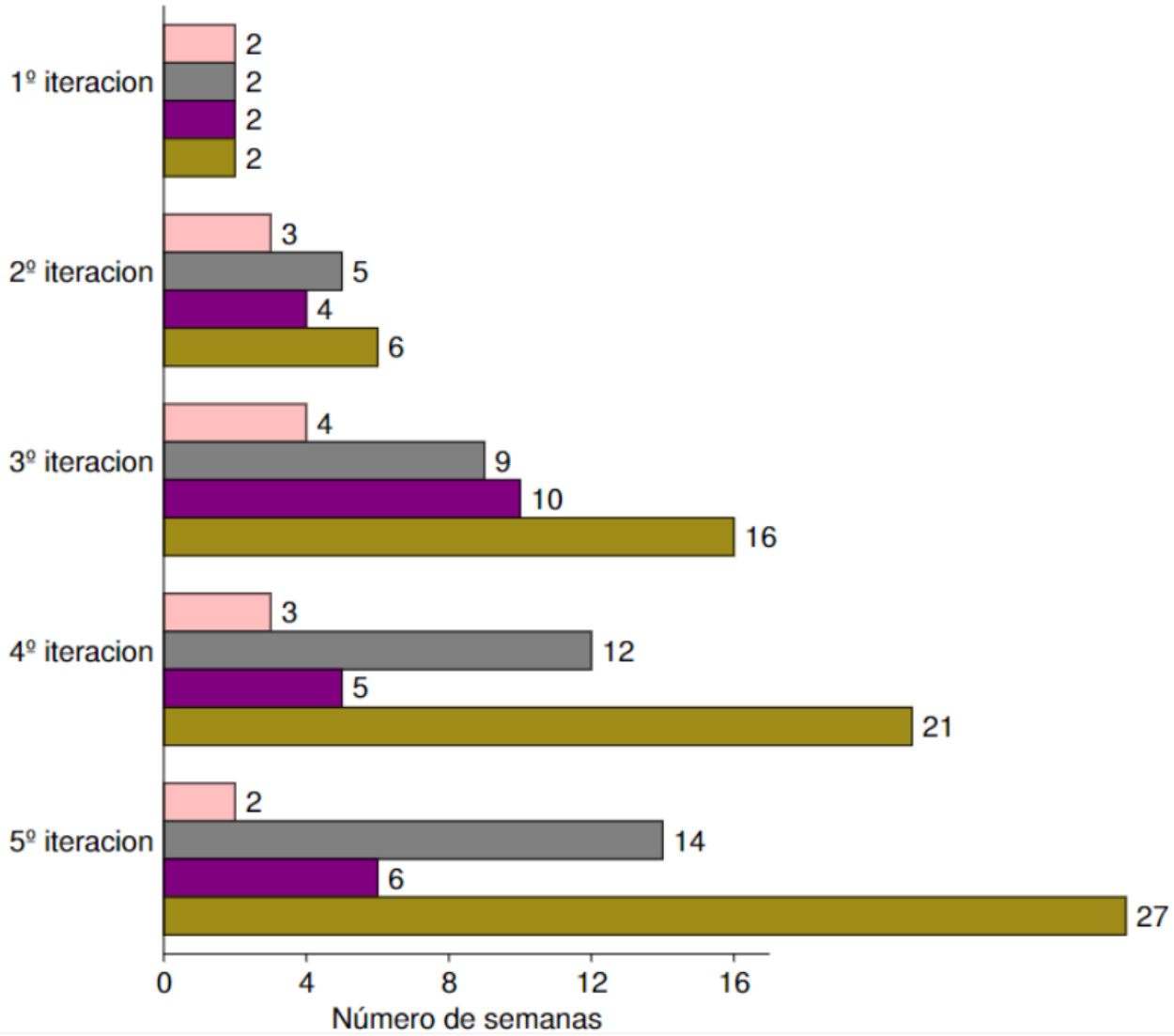


Figura 3.4: Distribución del número de semanas reales en cada iteración.

Capítulo 4

Descripción de las iteraciones

En este capítulo de la memoria del proyecto se van a explicar cada una de las iteraciones del TFG y el trabajo desarrollado en cada una de ellas.

4.1. Iteración 1

Durante la primera iteración el desarrollador se reunió con los clientes del proyecto, en este caso los tutores del TFG para conocer los requisitos funcionales iniciales.

Una vez explicado a grandes rasgos el desarrollo del proyecto, se le encargó a el alumno un primer trabajo con la memoria del proyecto, efectuando la planificación y el análisis de riesgos (3). Además, se realizaron tutoriales de uso de Python y de GitLab para que el desarrollador recordarse y fijase conceptos. Por último, el trabajador intentó aprender LaTeX desde cero, ya que nunca había realizado un documento con esta herramienta.

Algunos de los tutoriales realizados fueron:

- Curso de Python orientado al análisis de datos: `https://github.com/koldLight/curso-python-analisis-datos`
- Construir una GUI con Streamlit: `https://www.section.io/engineering-education/streamlit-ui-tutorial/`
- Manual de LaTeX: `https://manualdelatex.com/tutoriales`

4.2. Iteración 2

En esta iteración se realizó la primera toma de contacto con el entorno de desarrollo, Python. En primer lugar, se efectuaron más tutoriales para ir desarrollando fluidez en este lenguaje:

- Tutorial clases/objetos Python: `https://www.youtube.com/watch?v=wfcWRAxRVBA`
- Generar "$javadoc$" en Python: `https://realpython.com/documenting-python-code/`
- Pruebas en Python: `https://realpython.com/python-testing/`

Una vez realizados los tutoriales y expuestas las características de la aplicación por el cliente, el desarrollador tiene la capacidad de identificación de los requisitos, el objetivo del proyecto y las historias de usuario.

4.2.1. Requisitos funcionales

En este apartado se describen los requisitos funcionales del proyecto. Estos requisitos ofrecen una descripción del servicio que va a ofrecer la aplicación [57]. En la siguiente Tabla 4.1 se muestran los requisitos con su ID, su nombre, su descripción y su prioridad (Alta, Media, Baja).

ID	**Nombre**	**Descripción**	**Prioridad**
RF01	Crear clasificador	El sistema deberá permitir la creación de un clasificador automático o no automático (clustering)	Alta
RF02	Calcular correlación	El sistema deberá permitir al usuario realizar correlaciones	Alta
RF03	Calcular concordancia	El sistema deberá permitir medir la concordancia entre variables	Alta
RF04	Obtener información	El sistema deberá permitir la obtención de información sobre el conjunto de datos elegido por el usuario y entre variables, como media, mediana e intervalos de confianza	Alta
RF05	Cargar fichero	El sistema deberá permitir cargar un fichero de datos formato .csv o .txt	Alta
RF06	Escribir en fichero	El sistema deberá permitir escribir un DataFrame en un fichero	Alta
RF07	Fusionar dos ficheros	El sistema deberá permitir fusionar dos ficheros de datos mediante un campo común	Alto
RF08	Mostrar gráficos	El sistema deberá permitir mostrar gráficos por pantalla	Media
RF09	Calcular ganancia de información	El sistema deberá permitir calcular la ganancia de información de los atributos seleccionados	Alta
RF10	Crear regresión	El sistema deberá permitir crear una regresión	Alta
RF11	Seleccionar características	El sistema deberá permitir la selección de características mediante diferentes algoritmos	Alta
RF12	Probar aplicación	El sistema deberá permitir probar la librería mediante test	Alta
RF13	Llamar a funciones de R en Python	El sistema deberá permitir llamar a una función de R desde Python	Alta
RF14	Realizar test paramétricos	El sistema deberá permitir realizar test paramétricos	Alta
RF15	Realizar test no paramétricos	El sistema deberá permitir realizar test no paramétricos	Alta
RF16	Creación de tablas	El sistema deberá permitir la creación de tablas en distintos formatos	Alta
RF17	Seleccionar columnas	El sistema deberá permitir la selección de columnas elegidas por el usuario	Alta
RF18	Sustituir valores perdidos	El sistema deberá permitir al usuario elegir si quiere o no sustituir los valores perdidos de un conjunto de datos	Alta

Tabla 4.1: Tabla de requisitos funcionales

4.2.2. Requisitos no funcionales

En este apartado, se muestran los requisitos no funcionales del sistema que describen propiedades de este o imponen restricciones en el diseño o en la implementación. En la Tabla 4.2 se recogen estos requisitos junto a su ID, nombre, descripción y relevancia (Crítica, Deseable o Baja).

ID	Nombre	Descripción	Relevancia
RNF01	Multiplataforma	El sistema se podrá utilizar al menos en Windows 10 y en Linux (Ubuntu 20.04)	Crítica
RNF02	Facilidad de instalación	El sistema se podrá instalar siguiendo el manual de despliegue	Deseable
RNF03	Notificación de errores	El sistema informará al usuario en caso de que se produzca algún error	Crítica
RNF04	Protección de datos	El sistema deberá proteger los datos de los archivos utilizados	Crítica
RNF05	Formato de los archivos	El sistema utilizará archivos .csv o .txt	Crítica
RNF06	Visualización de resultados	El sistema mostrará por pantalla los resultados de ejecución	Crítica
RNF07	Versión de compilación	El sistema debe compilarse con la última versión de Python (3.9.0)	Crítica
RNF08	Codificación	El sistema permitirá la codificación de caracteres UTF-8	Deseable

Tabla 4.2: Tabla de requisitos no funcionales

4.2.3. Otros trabajos realizados

Por último, en esta iteración, se realizaron las **historias de usuario** del Capítulo 5 y se relacionaron con los requisitos para probar su veracidad. Asimismo, se creó el **diagrama de dominio** de la aplicación con sus respectivas clases, atributos y relaciones.

4.3. Iteración 3

En esta iteración se inició con la implementación de la funcionalidad de la aplicación. Se trata del periodo de trabajo que se ha desarrollado durante más cantidad de tiempo con numerosas reuniones con los tutores.

4.3.1. Implementación de código

Primeramente, se estableció como se iba a distribuir la biblioteca en Python, con sus respectivos directorios y funciones y se comenzó a programar. Más tarde se fueron implementando uno a uno los requisitos del proyecto, encontrando problemas en alguno de ellos debido a la inexperiencia realizando un proyecto extenso en complejidad y desarrollo. Por todo ello, esta iteración incrementó el tiempo de desarrollo del proyecto en varias semanas.

Muchos de los problemas encontrados se han detallado en el Capítulo 6 junto al archivo en el que se ubicó el problema.

Además, durante el avance de esta iteración, el cliente añadió más requisitos al proyecto, lo que aumentó la cantidad de trabajo y por ende de tiempo.

Una de las funciones interesantes implementadas en la aplicación ha sido aquella que selecciona las columnas. Se plantearon diferentes formas de selección, pero no cubrían todos los casos. Al final, y tras varias reuniones con los tutores, se decidió una función que al introducir una cadena de texto con las columnas requeridas (Ej.: 3:8,12,18) devuelva un *DataFrame* con las columnas seleccionadas (siendo ':' un intervalo y ',' una columna individual). Esta función se muestra en la siguiente Figura 4.1:

```
def seleccion_col(data, col_x, col_y = None):
    """Función que devuelve las columnas ya limpias, para poder realizar una regresión, anova..

    Args:
        data (Dataframe): Conjunto de datos sobre el que se va a realizar la regresión.
        col_x (str o list): Columnas de las variables que se van a utilizar como regresores. Si el usuario introduce
        all, la función tomará todas las columnas como regresores, excepto la que haya introducido como variable
        dependiente. Las columnas seleccionadas se deben introducir entre comillas, tanto como si queremos columnas individuales
        como intervalos, separados por comas. Ejemplo( '3:8,10,11')
        col_y(int): Columna en la que se encuentra la variable dependiente. Por defecto, no se intrpduce ninguna.

    Return:
        nuevo_df (Dataframe): Subconjunto de dataframe de las variables regresoras, en relación a lo que el
        usuario ha introducido por parámetro
    """

    #Si queremos todas las columnas excepto, la variable respuesta
    if ((col_x == all) and (col_y == None)):
        return data

    elif (col_x == all):
        nuevo_df = data.drop(data.columns[[col_y]], axis='columns')

    else:
        #Separamos las columnas y cada una queda en una posicion de la lista.
        lista = col_x.split(',')
        nuevo_df = pd.DataFrame()

        #Recorremos la lista para encontrar las columnas/intervalos
        for i in lista:
            #Añadimos a un nuevo dataframe las columnas intervalos
            lista1 = i.split(':')
            if(len(lista1) == 2):
                data1 = data.iloc[:,int(lista1[0]):(int(lista1[1]))+1]
                nuevo_df = pd.concat([nuevo_df, data1], axis=1)

            #Si solo tenemos una columna
            else:
                num = int(i)
                data2 = data.iloc[:, num]
                nuevo_df = pd.concat([nuevo_df, data2], axis=1)

    return nuevo_df
```

Figura 4.1: Función que selecciona las columnas de un *DataFrame*.

4.3.2. Pruebas de la aplicación

Al mismo tiempo que se han ido implementando los requisitos se han ido realizando pruebas mediante un módulo incluido en una biblioteca de Python, *doctest*. Este módulo permite la realización de pruebas basadas en la salida del *shell* de Python, en forma de *String* [58]. Además, se ha utilizado para escribir la documentación de cada función, ilustrado con ejemplos. En el Capítulo 6, en la Figura 6.17 se puede observar un ejemplo de las pruebas y documentación de una función de la aplicación.

4.3.3. Desarrollo de la memoria

Por último, se continuó escribiendo partes de la memoria al mismo tiempo que se iba implementando el código.

4.4. Iteración 4

4.4.1. Implementación gráfica

En un primer momento se planteó la realización de una interfaz gráfica de usuario (GUI). Sin embargo, se decidió con los tutores que era demasiada carga de trabajo y que se iban a implementar funciones gráficas interesantes para algunas de las funciones.

Se creó un directorio que alberga un archivo llamado *funciones_graficas.py* donde se representan los siguientes gráficos:

- Recta de regresión.
- Gráfico de dispersión.
- Gráfico de distribución.
- Número de características seleccionadas por un modelo.
- Clustering k-medias.
- Clasificador SVM.
- Diagrama de cajas.

4.4.2. Pruebas de la aplicación

Una vez realizada toda la implementación del proyecto, se realizaron pruebas de todas las funciones cubriendo todos los casos de entrada. Para ello se utilizaron distintos conjuntos de datos con diferentes características. Estas pruebas se pueden observar en el archivo *pruebas.py*.

Después de realizar estas pruebas, se encontraron errores que hubo que solucionar (algunos están detallados en el Capítulo 6), por lo que se realizó una revisión completa de este para solucionar individualmente cada error.

4.5. Iteración 5

Por último, en esta iteración se realizó la redacción formal de la memoria. Se comenzó completando aquellas secciones que estaban vacías y añadiendo diagramas realizados anteriormente, ya sea con Astah o con LucidChart. También se revisó la memoria de forma completa y se corrigió todos los fallos que hubiese con ayuda de los tutores.

Capítulo 5

Estado final de la aplicación

En este capítulo se va a tratar todo aquello relacionado con el diseño de la aplicación mediante los diagramas y la metodología necesaria.

5.1. Historias de usuario

En esta sección del TFG se van a detallar las historias de usuario llevadas a cabo de forma independiente durante el proyecto. Los prerrequisitos necesarios para que esta funcionalidad pueda realizarse son que el usuario se haya descargado el repositorio y haya pedido permiso para la descarga de los datos del proyecto, si quisiera utilizarlos.

Los campos de la tabla utilizada para todas las historias son los siguientes:

- ID: Identifica la historia de usuario.
- Nombre: Nombre que describe la historia realizada.
- Prioridad: Tipo de prioridad en el desarrollo de la funcionalidad de la historia de usuario. Puede ser: baja, media y alta.
- Riesgo: Tipo de impacto sobre el proyecto si se produce un fallo en la historia de usuario. Puede ser: bajo, medio y alto.
- Descripción: Se compone de un texto que describe la funcionalidad que debe llevar a cabo la historia de usuario.
- Validación: Condiciones que se deben cumplir para poder considerar la historia de usuario válida y finalizada.

ID	HU01
Nombre	Cálculo de los coeficientes de correlación
Prioridad	Alta
Riesgo	Bajo
Descripción	Como usuario quiero poder calcular el coeficiente de correlación de un conjunto de datos según el tipo de variables que se posean
Validación	- Quiero que se me informe si mis datos son o no paramétricos - Quiero que se me informe de la dispersión de las variables respecto a la variable dependiente - Quiero que se me informe de la distribución para cada variable numérica
RFs	RF02

Tabla 5.1: Historia de usuario HU01

ID	HU02
Nombre	Análisis de concordancia entre variables
Prioridad	Alta
Riesgo	Bajo
Descripción	Como usuario quiero poder calcular la concordancia entre variables de un conjunto de datos.
Validación	- Quiero que la aplicación calcule el índice de Kappa si las variables que analizo son categóricas. - Quiero que la aplicación calcule el índice de Kendall si las variables que analizo son numéricas.
RFs	RF03

Tabla 5.2: Historia de usuario HU02

ID	HU03
Nombre	Obtención de un clasificador
Prioridad	Alta
Riesgo	Bajo
Descripción	Como usuario quiero poder aplicar un clasificador supervisado o no supervisado a un conjunto de datos.
Validación	- Quiero poder elegir el tipo de clasificador que quiero calcular - Quiero que la aplicación me informe de medidas para el acierto en la predicción de datos. - Quiero que la aplicación entrene los datos en caso de que se quiera usar aprendizaje supervisado.
RFs	RF01

Tabla 5.3: Historia de usuario HU03

ID	HU04
Nombre	Descripción del conjunto de datos
Prioridad	Alta
Riesgo	Bajo
Descripción	Como usuario quiero poder obtener información sobre el conjunto de datos que estoy manejando.
Validación	- Quiero poder obtener intervalos de confianza según cada grupo de mi variable separadora, y medidas como media, mediana...
RFs	RF04

Tabla 5.4: Historia de usuario HU04

ID	HU05
Nombre	Carga de un fichero de datos
Prioridad	Alta
Riesgo	Bajo
Descripción	Como usuario quiero poder cargar un fichero de datos.
Validación	- Quiero poder cargar un fichero eligiendo la ruta, el separador, la cabecera y el índice de este.
RFs	RF05

Tabla 5.5: Historia de usuario HU05

ID	HU06
Nombre	Fusión de dos ficheros
Prioridad	Alta
Riesgo	Bajo
Descripción	Como usuario quiero poder fusionar dos ficheros de datos a partir de un campo común.
Validación	- Quiero poder elegir el tipo de fusión ejecutada - Quiero que la columna común a ambos ficheros se establezca como índice del nuevo fichero
RFs	RF07

Tabla 5.6: Historia de usuario HU06

ID	HU07
Nombre	Escribir un DataFrame en un fichero
Prioridad	Alta
Riesgo	Bajo
Descripción	Como usuario quiero poder escribir un DataFrame obtenido en un fichero.
Validación	- Quiero que si el fichero existe, se borre el contenido y escriba el DataFrame. - Quiero que si el fichero no existe, cree un nuevo fichero con el contenido del DataFrame.
RFs	RF06

Tabla 5.7: Historia de usuario HU07

ID	HU08
Nombre	Quiero que algunas funciones de mi aplicación sean representadas gráficamente
Prioridad	Alta
Riesgo	Bajo
Descripción	Como usuario quiero poder ver visualmente ciertos gráficos.
Validación	- Quiero poder elegir las características del gráfico que me ofrezca la aplicación.
RFs	RF08

Tabla 5.8: Historia de usuario HU08

ID	HU09
Nombre	Cálculo de la ganancia de información
Prioridad	Alta
Riesgo	Bajo
Descripción	Como usuario quiero poder obtener la ganancia de información de un conjunto de variables
Validación	- Quiero poder elegir el criterio utilizado para el cálculo de la ganancia. - Quiero obtener los valores de la ganancia de información en una tabla ordenados de mayor a menor junto a la variable a la que pertenece.
RFs	RF09

Tabla 5.9: Historia de usuario HU09

ID	HU10
Nombre	Obtención de una regresión y sus resultados
Prioridad	Alta
Riesgo	Bajo
Descripción	Como usuario quiero poder realizar una regresión lineal o logística.
Validación	- Quiero poder elegir el tipo de regresión que deseo realizar. - Quiero que la aplicación me proporcione la pendiente, el R-cuadrado del modelo y el error. - Quiero obtener predicciones del modelo ajustado y su representación. - Quiero que se muestre la representación de la regresión en el caso de que esta tenga un solo regresor.
RFs	RF10

Tabla 5.10: Historia de usuario HU10

ID	HU11
Nombre	Llamar a funciones R dentro de Python
Prioridad	Alta
Riesgo	Bajo
Descripción	Como usuario quiero poder llamar a funciones de R dentro de Python.
Precondición	Tener instalado R y la librería donde se encuentra la función a llamar
Validación	- Quiero poder elegir parámetros de la función de R a probar.
RFs	RF13

Tabla 5.11: Historia de usuario HU11

ID	HU12
Nombre	Selección de características según un criterio.
Prioridad	Alta
Riesgo	Bajo
Descripción	Como usuario quiero poder seleccionar las características de un conjunto de datos según importancia y con diferentes algoritmos.
Validación	- Quiero poder elegir entre varios algoritmos de selección. - Quiero poder seleccionar que columnas deseo comparar.
RFs	RF11

Tabla 5.12: Historia de usuario HU12

ID	HU13
Nombre	Creación de tablas
Prioridad	Alta
Riesgo	Bajo
Descripción	Como usuario quiero poder crear distintos tipos de tablas, especialmente en formato LaTeX.
Validación	- Quiero poder introducir cabecera si así lo deseo. - Quiero elegir el formato en el que obtener la tabla. - Quiero poder alinear el contenido de cada celda. - Quiero elegir si la tabla resultante va a tener índice o no.
RFs	RF16

Tabla 5.13: Historia de usuario HU13

ID	HU14
Nombre	Realización de test no paramétricos
Prioridad	Alta
Riesgo	Bajo
Descripción	Como usuario quiero poder realizar tests no paramétricos.
Validación	- Quiero poder elegir el test no paramétrico a efectuar. - Quiero obtener información de si es recomendable usar ese test con los datos ejecutados. - Quiero poder obtener el p-valor y estadístico resultado de cada test. - Quiero poder obtener test post-hoc de cada test realizado.
RFs	RF15

Tabla 5.14: Historia de usuario HU14

ID	HU15
Nombre	Realización de test paramétricos
Prioridad	Alta
Riesgo	Bajo
Descripción	Como usuario quiero poder realizar test paramétricos.
Validación	- Quiero poder elegir el test paramétrico a efectuar. - Quiero obtener información de si es recomendable usar ese test con los datos ejecutados. - Quiero poder obtener el p-valor y estadístico resultado de cada test. - Quiero poder obtener test post-hoc de cada test realizado.
RFs	RF14

Tabla 5.15: Historia de usuario HU15

ID	HU16
Nombre	Selección de columnas.
Prioridad	Alta
Riesgo	Bajo
Descripción	Como usuario quiero seleccionar las columnas que me interesen de un conjunto de datos.
Validación	- Quiero poder elegir un intervalo de columnas o columnas individuales.
RFs	RF17

Tabla 5.16: Historia de usuario HU16

ID	HU17
Nombre	Sustitución de valores NaN.
Prioridad	Alta
Riesgo	Bajo
Descripción	Como usuario quiero elegir si deseo sustituir los valores NaN de las columnas numéricas.
Validación	- Quiero poder elegir el intervalo de columnas a sustituir.
RFs	RF18

Tabla 5.17: Historia de usuario HU17

5.2. Modelo de dominio

En esta sección se va a definir el modelo de dominio de la aplicación con sus respectivas entidades, relaciones y multiplicidades. En la Figura 5.1 se presenta este modelo:

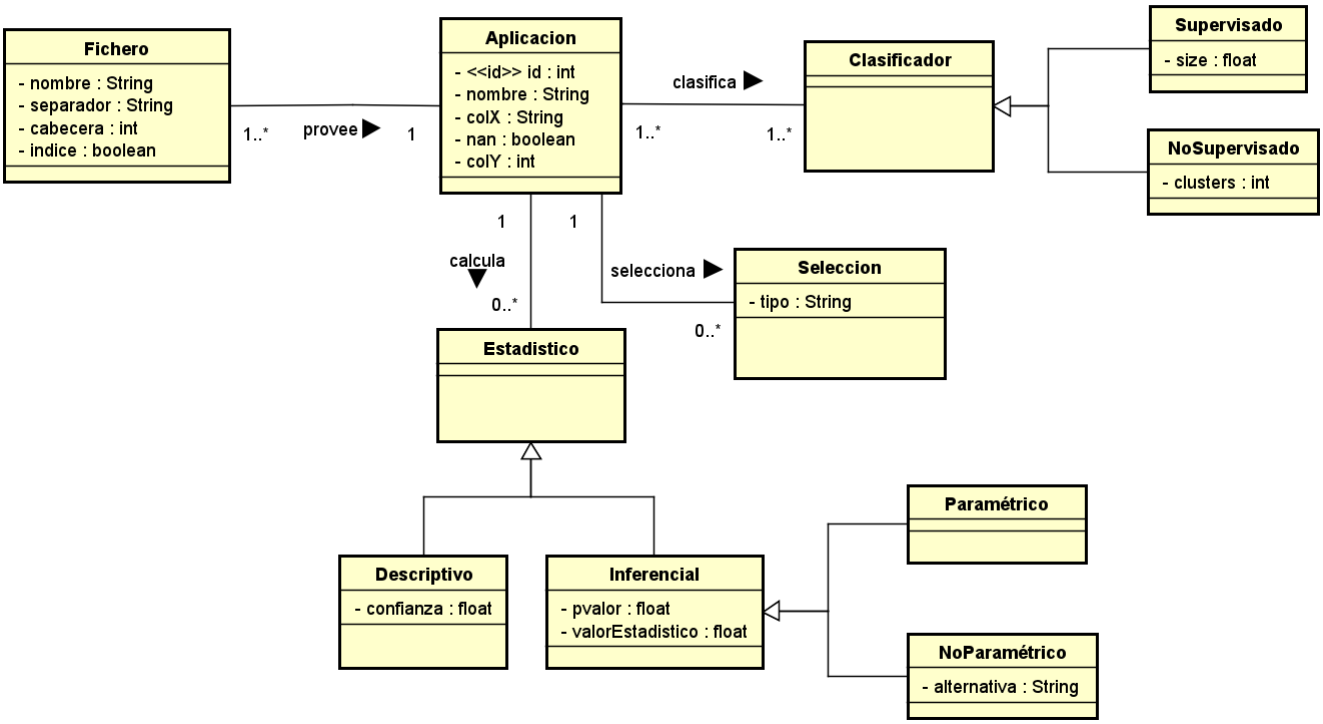


Figura 5.1: Diagrama de dominio del modelo.

En las siguientes tablas se van a describir cada una de las entidades del modelo:

Fichero	
Descripción	Entidad que modela los ficheros utilizados durante la aplicación
Responsabilidades	Modelar un fichero con los atributos necesarios para que sea empleado durante el desarrollo de la aplicación.
Atributos	- nombre: nombre que identifica al fichero. - separador: tipo de separador utilizado entre columnas. - cabecera: número de fila que se emplea como cabecera. - indice: señala si el fichero quiere utilizarse o no con índice.

Tabla 5.18: Entidad Fichero

Aplicacion	
Descripción	Entidad que modela la utilización de la aplicación
Responsabilidades	Modelar un uso de la aplicación, con los datos de interés.
Atributos	- id: identificador que determina la operación que se quiere efectuar. - nombre: nombre que identifica la operación que se desea realizar. - colX: columnas utilizadas como variables independientes. - cabecera: columna empleada como variable respuesta. - nan: señala si las variables independientes se van a sustituir o no si existen valores perdidos.

Tabla 5.19: Entidad Aplicacion

Clasificador	
Descripción	Entidad que modela la creación de un clasificador
Responsabilidades	Modelar los detalles de creación y uso de un clasificador.

Tabla 5.20: Entidad Clasificador

Supervisado	
Descripción	Entidad que modela la creación de un clasificador mediante aprendizaje supervisado
Responsabilidades	Modelar los detalles de creación y uso de un clasificador mediante aprendizaje supervisado.
Atributos	- size : tamaño del conjunto entrenamiento que se desea utilizar.

Tabla 5.21: Entidad Supervisado

NoSupervisado	
Descripción	Entidad que modela la creación de un clasificador mediante aprendizaje no supervisado
Responsabilidades	Modelar los detalles de creación y uso de un clasificador mediante aprendizaje no supervisado.
Atributos	- clusters: número de clusters que se quieren generar.

Tabla 5.22: Entidad NoSupervisado

Seleccion	
Descripción	Entidad que modela la creación de un método de selección de características.
Responsabilidades	Modelar el proceso de selección de características de un conjunto de datos para mejorar las predicciones.
Atributos	- tipo : método de selección que de desea utilizar.

Tabla 5.23: Entidad Seleccion

Estadístico	
Descripción	Entidad que modela la creación de un estadístico.
Responsabilidades	Modelar el proceso de creación de un estadístico, con todas sus características.

Tabla 5.24: Entidad Estadístico

Descriptivo	
Descripción	Entidad que modela la descripción de un conjunto de datos.
Responsabilidades	Modelar la descripción de un conjunto de datos, con todas sus características.
Atributos	- confianza: nivel de confianza para la creación de intervalos de confianza

Tabla 5.25: Entidad Descriptivo

Inferencial	
Descripción	Entidad que modela la creación de un estadístico inferencial.
Responsabilidades	Modelar la creación de un estadístico que realice un test inferencial.
Atributos	-pvalor: p-valor obtenido como resultado en la realización del test inferencial. -valorEstadistico: valor del estadístico del test obtenido.

Tabla 5.26: Entidad Inferencial

Parametrico	
Descripción	Entidad que modela la realización de un test paramétrico.
Responsabilidades	Modelar la realización de un test paramétrico con todos los atributos de este.

Tabla 5.27: Entidad Parametrico

NoParametrico	
Descripción	Entidad que modela la realización de un test no paramétrico.
Responsabilidades	Modelar la realización de un test no paramétrico con todos los atributos de este.
Atributos	-alternativa: tipo de hipótesis que se desea utilizar en el test no paramétrico.

Tabla 5.28: Entidad NoParametrico

5.2.1. Relaciones entre entidades

De la entidad **Aplicacion** salen cuatro asociaciones. Con la entidad **Fichero**, la relación **provee** hace referencia a los ficheros utilizados durante el desarrollo de la aplicación. La relación **clasifica** con la entidad **Clasificador** alude a la creación de un clasificador. La entidad **Estadistico** se relaciona con

calcula, que simboliza el cálculo de uno o varios estadísticos. Por último, la relación **selecciona** con la entidad **Seleccion** hace referencia a la posibilidad de selección de 0 o varios atributos en la aplicación, de acuerdo a diversos criterios.

5.3. Estructura del proyecto

En esta sección del proyecto se va a mostrar cómo se estructuran los directorios de la aplicación y su contenido. En la Figura 5.2 se puede ver la división de esta aplicación en directorios.

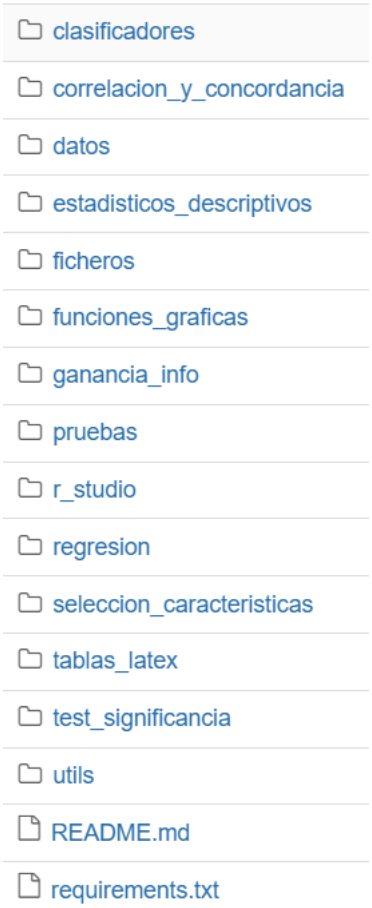


Figura 5.2: Estructura del proyecto

Seguidamente, se va a explicar el contenido de cada directorio. En todos los directorios que contienen archivos Python siempre hay dos archivos, uno *.py* (archivo Python de código con texto sin formato) y otro *.ipynb* (archivo de Jupyter Notebook que contiene el código y resultados de la ejecución).

- **clasificadores:** Directorio donde se encuentra los archivos que permiten crear clasificadores automáticos y no automáticos.
- **correlacion_y_concordancia:** En esta carpeta se localizan los archivos para realizar correlaciones y concordancias con distintos conjuntos de datos.
- **datos:** Directorio donde se encuentran todos los archivos de datos que se han empleado para las pruebas unitarias y *doctest*. Este directorio es privado debido a los datos que contiene y para acceder desde el repositorio es necesario pedir permiso a los creadores.
- **estadisticos_descriptivos:** En esta carpeta se encuentran los archivos que permiten describir estadísticamente el conjunto de datos.
- **ficheros:** Directorio que alberga los archivos para leer, fusionar y escribir conjuntos de datos.
- **funciones_graficas:** Carpeta donde se encuentran los archivos que realizan los gráficos de la aplicación.
- **ganancia_info:** Directorio donde se localizan los archivos para calcular la propiedad estadística *ganancia de información*.
- **pruebas:** Carpeta que alberga las pruebas unitarias realizadas para cada función, con sus respetivos resultados.
- **r_studio:** En esta carpeta se encuentra un ejemplo de cómo utilizar la interfaz de R dentro de un proceso *Python*.
- **regresion:** Directorio donde se encuentran los archivos que realizan regresiones lineales o logísticas.
- **seleccion_caracteristicas:** Carpeta que alberga archivos que utilizan distintos métodos para la selección de características de un conjunto de datos.
- **tablas_latex:** Directorio con archivos que permiten crear tablas en formato LaTeX y otros formatos, como grid, simple, html...
- **test_significancia:** Carpeta donde se localizan archivos para la realización de test paramétricos y no paramétricos.
- **utils:** En este directorio se encuentran los archivos que están formados por funciones que se utilizan en varios archivos, tales como la selección de columnas, la comprobación y sustitución de valores perdidos o la conversión de matrices.
- **README.md:** Fichero de texto que contiene información sobre el proyecto.
- **requirements.txt:** Fichero de texto donde se localizan todas las librerías Python que se necesitan instalar para que el proyecto funcione correctamente

A continuación, se van a mostrar unos diagramas creados para explicar gráficamente el contenido de los archivos de *Python* relacionados con el Machine Learning y los tipos de estadística:

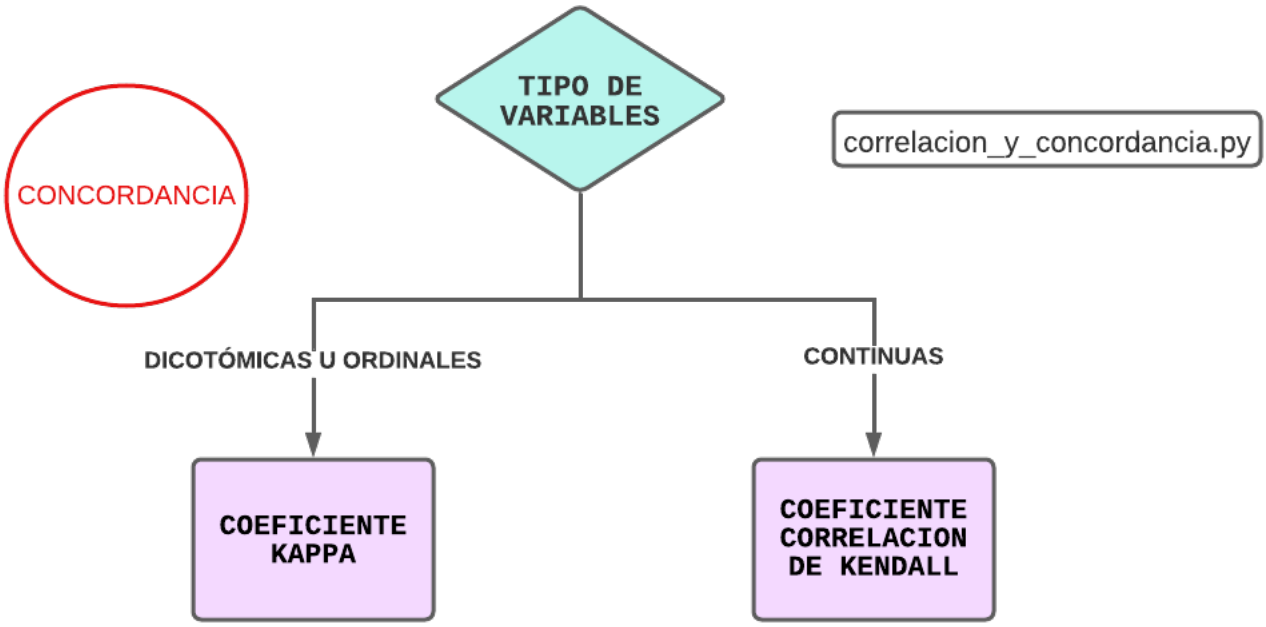


Figura 5.3: Esquema del contenido del archivo correlacion_y_concordancia.py 1.0.

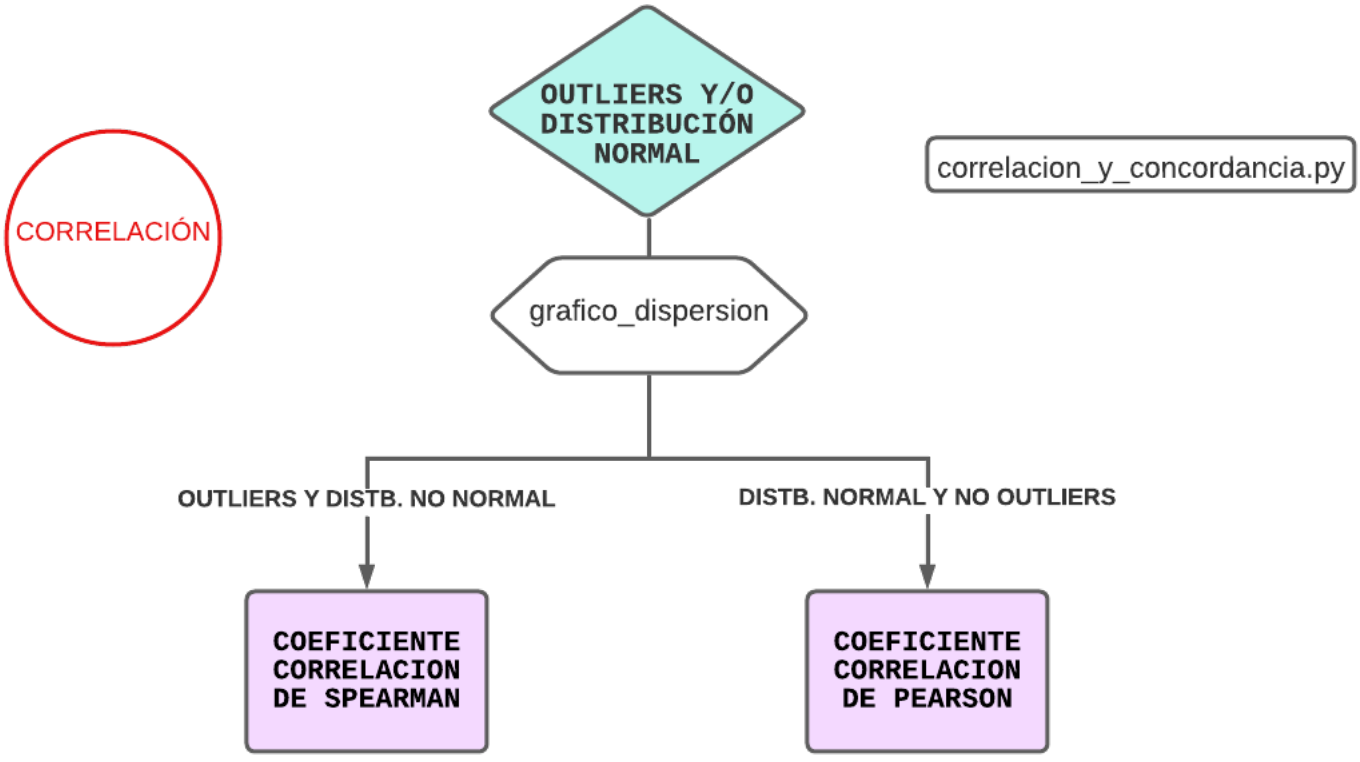


Figura 5.4: Esquema del contenido del archivo correlacion_y_concordancia.py 2.0.

Tal y como se puede observar en las Figuras 5.3 y 5.4 en el archivo $correlacion\_y\_concordancia.py$ se realizan correlaciones y análisis de concordancia. En cuanto a la concordancia, si las variables son continuas se calculará el coeficiente de **correlación de Kendall**, mientras que si las variables son dicotómicas u ordinales se calculará el **coeficiente Kappa**. Para la correlación dependerá de la distribución de los datos. Si estos se distribuyen normalmente y no hay presencia de outliers se calculará el **coeficiente de correlación de Pearson**, mientras que si la distribución de los datos no es normal y existen outliers, se calculará el **coeficiente de correlación de Spearman**.

En la siguiente Figura 5.5 se refleja el contenido del archivo $seleccion\_caracteristicas.py$, donde es posible seleccionar características de un conjunto de datos de varias formas: **método de envoltura** (Forward y Backward selection) o **método de ensamble** (Random Forest).

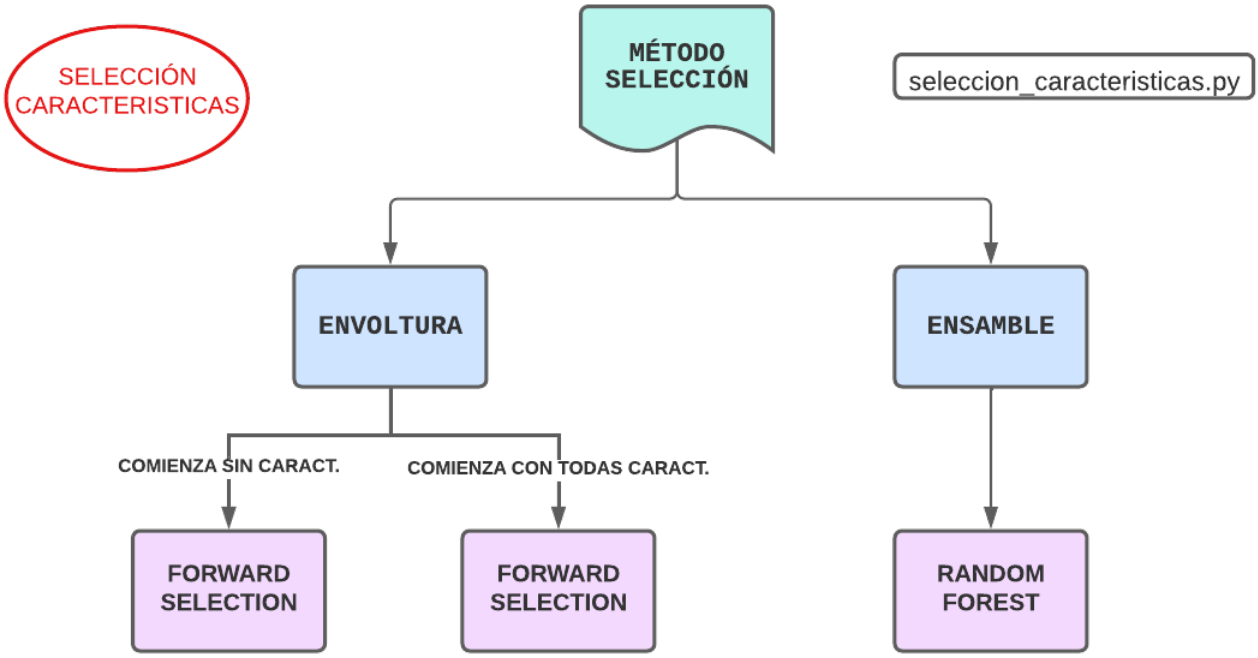


Figura 5.5: Esquema del contenido del archivo seleccion_caracteristicas.py.

En la Figura 5.6 a continuación se muestra el contenido de los archivo $clasificadores.py$ y $regresion.py$.

Existen dos tipos de aprendizaje supervisado y no supervisado. Dentro del aprendizaje supervisado se encuentran algoritmos de **regresión** (lineal y logística) o de **clasificación** (SVM y árboles de decisión). Por otro lado, en el aprendizaje no supervisado y dentro de los algoritmos jerárquicos se ha implementado el **clustering** o análisis de grupos.

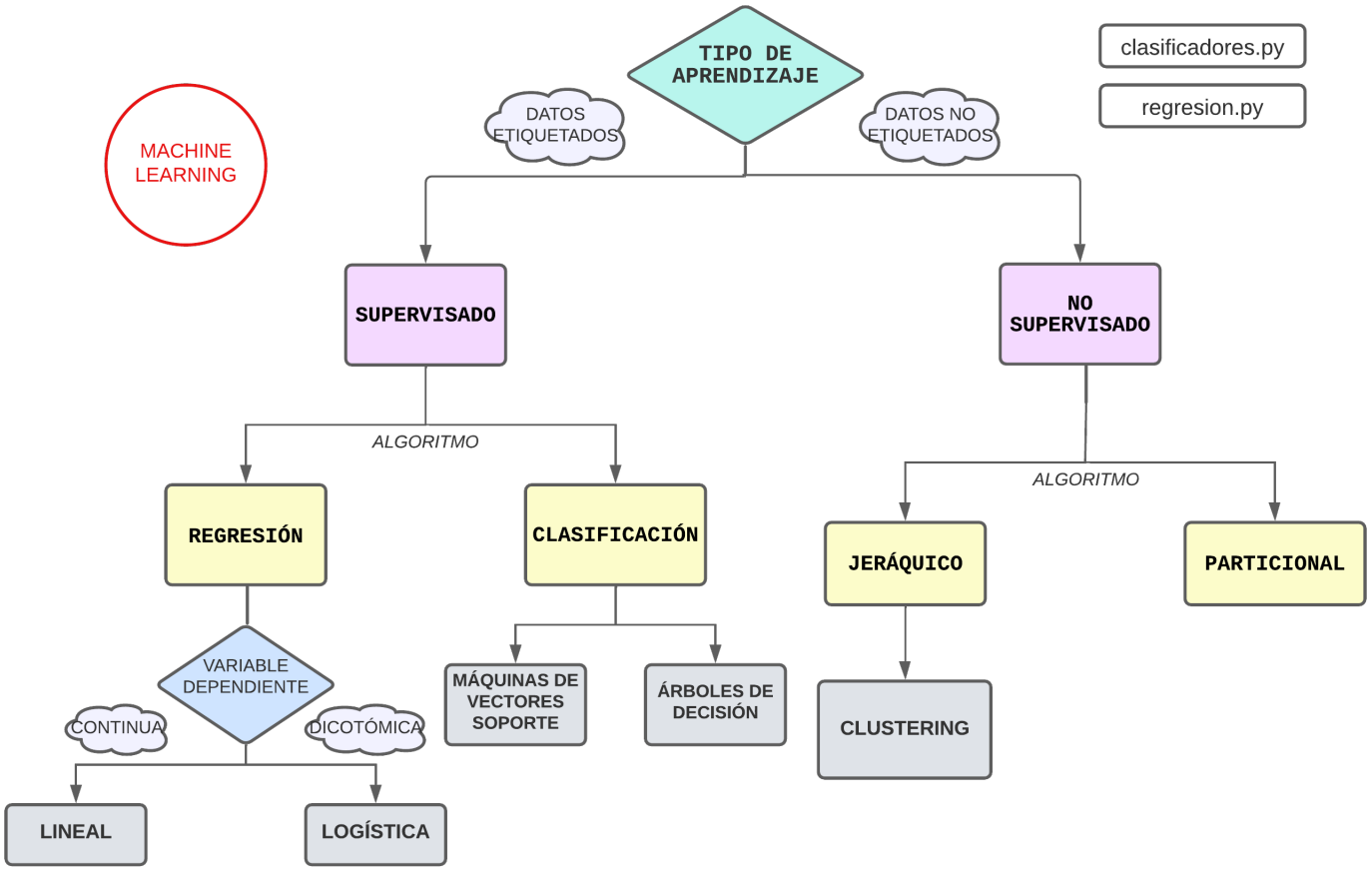


Figura 5.6: Esquema del contenido de los archivos clasificadores.py y regresion.py.

Por último, en los siguientes esquemas se muestra el contenido del archivo $test\_significancia.py$, que realiza tanto **test paramétricos** (Figura 5.8) como **test no paramétricos** (Figura 5.7).

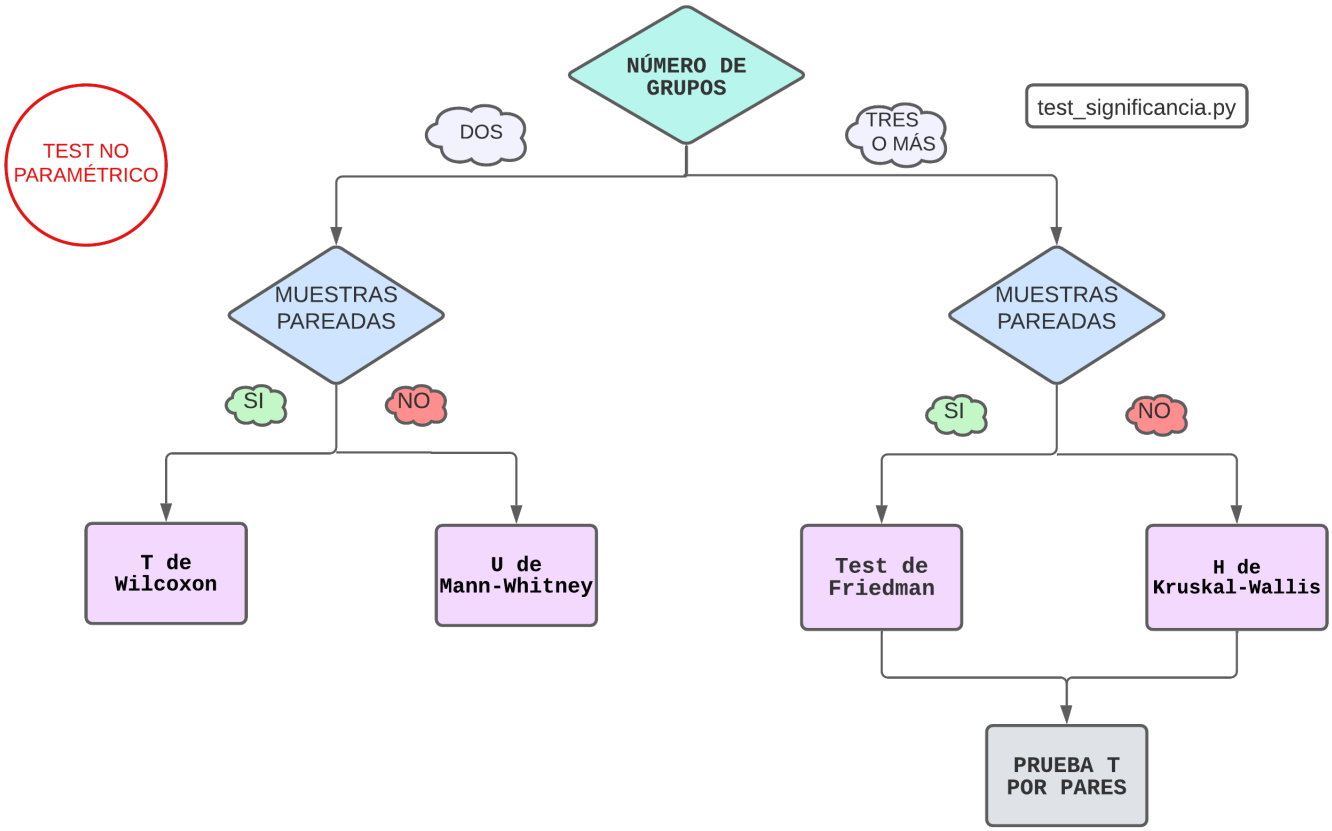


Figura 5.7: Esquema del contenido del archivo test_significancia.py test no paramétricos.

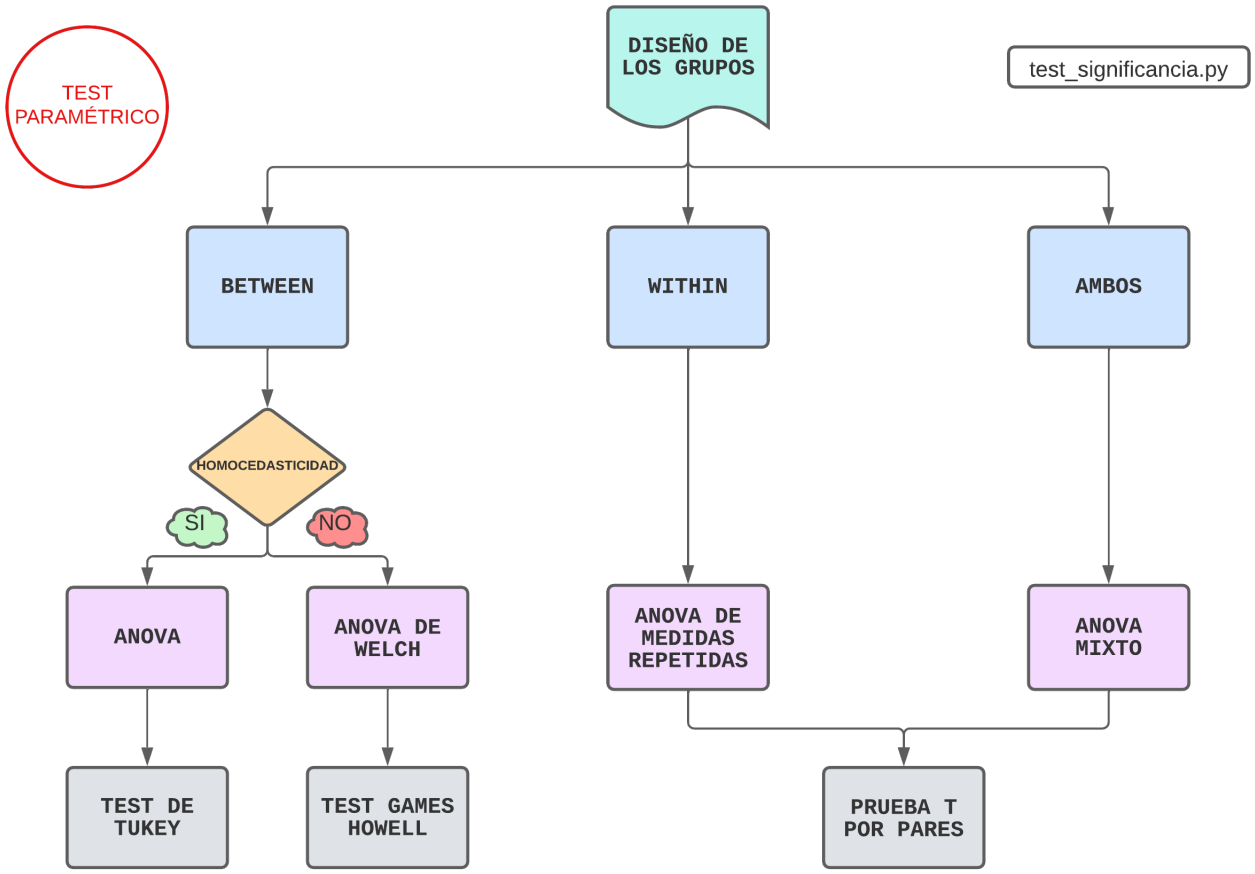


Figura 5.8: Esquema del contenido del archivo test_significancia.py test paramétricos.

5.4. Implementación del código

En esta sección del proyecto se va a explicar cómo se ha implementado el código y su estructura general.

Cada archivo Python cuenta al menos una función relacionada con el nombre del directorio en el que se encuentran. Por ejemplo, dentro de la carpeta *clasificadores*, se halla dos archivos, uno *.py* y otro *.ipynb* con el mismo contenido. Estos archivos cuentan con tres funciones: una función que crea un algoritmo SVM, otra creadora de árboles de decisión y, por último, una que realiza clustering.

Cada función tiene una estructura similar: argumentos de **entrada** y **salida**; una **documentación** que explica el contenido de la función, sus argumentos y la salida; y **pruebas** realizadas con *doctest* incluidas dentro de la documentación. Además, dentro del archivo *pruebas.py* se realizan pruebas semiautomáticas para cada función.

Por ejemplo, en las Figuras 5.9 y 5.10 se pueden observar los argumentos de entrada y salida de la función *clasificador_dt*. En la Figura 5.11 la documentación de esta función y en la Figura 5.12 las pruebas *doctest* realizadas.

```
def clasificador_dt ( data, col_x, col_y, size, criterio = 'gini', prof = None,
                                                split = 'best', ignorar_nan = True):
```

Figura 5.9: Argumentos de entrada de la función *clasificador_dt*.

```
    return c, predicciones, arbol
```

Figura 5.10: Argumentos de salida de la función *clasificador_dt*.

```
    """Los árboles de decisión (DT) son un método de aprendizaje supervisado no paramétrico que se utiliza para la
    clasificación y la regresión . El objetivo de esta funciónes crear un modelo que prediga el valor de una variable
    de destino mediante el aprendizaje de reglas de decisión simples deducidas de las características de los datos.

    Las variables introducidas como variables independientes deben ser variables numéricas, ya sean continuas o discretas,
    sino el algoritmo no lo soportará. En caso de tener variables categóricas, convertirlas primero a numéricas.

    Args:
        data (Dataframe): Conjunto de datos del que se quiere obtener la clasificación.
        col_x (str o all):Columnas de las variables que se van a utilizar como variables independientes.
        Las columnas seleccionadas se deben introducir entre comillas, separado entre comas ,
        tanto si se quiere un intervalo de columnas, como varias columnas individuales. Ejemplo: '3:8,2,9'.
        Si se desean todas las columnas, se introducirá la palabra reservada all.
        col_y (int): Columna en la que se encuentra la variable dependiente.
        size (float): Tamaño en el que se quiere dividir el conjunto de entrenamiento. Debe estar entre 0 y 1.
        criterio (str): Criterio utilizado para la medida de selección de atributos, (gini o entropy),
        por defecto se usa el indice de Gini.
        prof (int o None): Profundidad máxima que queremos para nuestro árbol, por defecto es None (ninguna),
        por lo que ajustará todo el árbol.
        split(str): Permite elegir la estrategia de división (best o random).
        ignorar_nan (bool): Parámetro que permite al usuario decidir si quiere ignorar los valores nan o sustituirlos por
        la media de cada grupo ( si existiesen )

    Return:
        c (list): Lista de medidas estadísticas utilizadas para saber el acierto en la prediccion de datos:
        accuracy, matriz de confusión, precision, recall y puntuación F equilibrada o medida F.
        predicciones (array): predicciones del modelo ajustado
        arbol (clasificador): (sklearn.tree._classes.DecisionTreeClassifier) Clasificador en bruto
```

Figura 5.11: Documentación de la función *clasificador_dt*.

```
    >>> clasificador_dt(data, 3, 12, 2/3)
    'Introduzca un valor válido de columnas'

    >>> clasificador_dt(data, '3,8', -2, 2/3)
    ('La variable respuesta no esta dentro del data set. Introduzca un valor entre 0 y', 12)

    >>> clasificador_dt(data, '3,8', 12, 1.5)
    'Introduza un valor de tamaño para el entrenamiento entre 0 y 1'

    >>> clasificador_dt(data, '3,8', 12, 2/3, 'gin')
    'Introduzca un criterio válido : entropy or gini.'

    >>> clasificador_dt(data, '3,8', 12, 2/3, 'gini', -3)
    'Introduzca una profundidad del árbol válida, mayor que 0.'

    >>> clasificador_dt(data, '3,8', 12, 2/3, 'gini', 5, 'rando')
    'Introduzca una división válida : best o random.'

    >>> data = pd.DataFrame()
    >>> clasificador_dt(data, '3,8', 12, 2/3)
    'No puede introducir un dataset vacío'

    """
```

Figura 5.12: Pruebas doctest de la función *clasificador_dt*

5.5. Bibliotecas

Del mismo modo que se ha decidido usar el lenguaje de programación Python por su sencillez, rapidez y demás ventajas comentadas anteriormente, Python consta con una gran cantidad de bibliotecas con sus correspondientes funciones. Estas bibliotecas abarcan desde la visualización de datos, *machine learning*, o *deep learning*, hasta el cálculo numérico o análisis de datos.

Para finalizar este capítulo, se van a enumerar y describir las bibliotecas de Python utilizadas durante el desarrollo del proyecto:

1. **pandas [59]:** Proporciona estructuras de datos de alto nivel y funciones para que el trabajo más sea rápido y sencillo. Los principales objetos usados en esta librería son: *DataFrame*, estructura de datos con etiquetas de filas y columnas; y *Series*, una matriz etiquetada unidimensional.

   Por lo tanto, pandas combina las operaciones de matrices de alto nivel (*numpy*) con la manipulación de bases de datos (como SQL).

2. **numpy [60]:** Es la abreviatura de *Numerical* y una de las primeras librerías creadas en Python. Facilita la creación de estructuras de datos y algoritmos. Además, usa funciones matemáticas relacionadas con el empleo de datos numéricos, operaciones matriciales y de álgebra lineal, generación de números aleatorios...

3. **scipy [61]:** Biblioteca que cuenta con una colección de paquetes que abordan diferentes problemas de computación científica, desde optimizadores de funciones y algoritmos de búsqueda (*scipy.optimize*) hasta pruebas estadísticas y estadísticos descriptivos (*scipy.stats*).

   En conjunto, *scipy* y *numpy* son la base para muchas aplicaciones informáticas científicas tradicionales.

4. **scikit-learn [62]:** Se creó en 2010, junto a *pandas* y solo en 7 años ya tenía 1500 colaboradores por todo el mundo [63]. Algunos submódulos que incluye son: regresión, clasificación, clustering, selección de modelos y características o normalización.

5. **matplotlib [64]:** Paquete de creación de imágenes y gráficos 2D, para la visualización de datos económicos, científicos y financieros. Se puede usar de forma interactiva desde el *shell* de Python o desde una GUI. [65]

6. **seaborn [66]:** Biblioteca utilizada para gráficos estadísticos que proporciona una interfaz de alto nivel a la biblioteca *matplotlib* y se integra con las estructuras de datos creadas con *pandas*.

   Además, produce gráficos completos a partir de una sola llamada a una función y facilita la creación rápida de prototipos y el análisis exploratorio de datos. [67].

7. **collections [68]:** Este módulo ayuda a manipular estructuras de datos (contenedores) especializados tales como tuplas, listas y diccionarios. Una de las clases implementadas más famosas es Counter. En ella los elementos se almacenan como llaves de diccionario y sus conteos se almacenan como valores de ese diccionario.

8. **pingouin [69]:** Paquete de Python que se encarga de proporcionar pruebas estadísticas usadas diariamente de una forma sencilla. Algunos ejemplos podrían ser : ANOVA, ANCOVA, correlación, regresión, comparaciones múltiples...

9. **doctest [70]:** Esta librería los siguientes usos [58]:

   - Revisar que los *docstrings* (variables especiales que describir para qué sirven y como se deben usar los objetos de un módulo) se encuentran actualizados.
   - Escribir documentación para cada función, con ejemplos de entrada-salida.

10. **mlxtend [71]:** Biblioteca que implementa numerosos algoritmos útiles para la minería de datos y Machine Learning. Implementa algoritmos de selección de características secuenciales, algoritmos para minería de patrones frecuentes y técnicas de evaluación de modelos para comparar el rendimiento de modelos diferentes [72].

11. **tabulate [73]:** Biblioteca de Python cuyos principales usos son:

    - Impresión de pequeñas tablas.
    - Presentación legible de datos.

12. **linearmodels [74]:** Librería que extiende las utilidades de la biblioteca *stats* con regresiones Panel y de alto nivel, estimadores variables, efectos fijos y variables... Es compatible con *pandas* y *numpy* [74].

13. **rpy2 [75]:** Librería que concentra la interfaz de Python para el lenguaje R, proporcionando acceso a R desde Python utilizando la API de R.

14. **random [76]**: Esta biblioteca implementa funciones para la generación de números aleatorios de diferentes formas.

Para más detalles acerca de la instalación y uso de las librerías, ver el Apéndice B.

Capítulo 6

Pruebas

Durante el desarrollo del proyecto se han llevado a cabo pruebas con diferentes conjuntos de datos para validar cada una de las funciones implementadas en el software. Las pruebas realizadas para las 20 funciones probadas se han resumido en la siguiente Tabla 6.1:

	CONJUNTOS DE DATOS			
FUNCIONES PROBADAS	*fea_gemaps.txt*	*ev_experts.csv*	*partidas_todas.csv*	*random*
clasificador_svm()	X			
clasificador_dt()	X			
clasificador_cluster()	X			
correlacion()	X			
concordancia()	X			
estadisticos_descrip()	X			
ganancia_info()	X			
regresion()	X		X	
random_forest()	X		X	
sequential_selection()	X		X	
tablas_latex()	X			
chi_square_test()	X		X	
cargar_fichero()	X	X	X	
fusion_ficheros()	X	X		
escribir_en_fichero()	X			
test_mann_whitney()	X		X	X
test_wilcoxon()	X		X	X
test_kruskal_wallis()	X		X	X
test_anova_unidireccional()	X			X
test_anova_bidireccional()	X			X

Tabla 6.1: Pruebas realizadas para cada función y conjunto de datos..

En todas estas pruebas están implícitos los test de los archivos *utils.py* y *funciones_graficas.py*, así como las funciones privadas de cada archivo, ya que no se prueban directamente, pero sí al llamar a una función que las implementa. Estas pruebas y datos se van a detallar a continuación.

6.1. Conjuntos de datos

Se han utilizado varios conjuntos de datos obtenidos de diferentes formas. El primer conjunto de datos *(fea_gemaps.txt)* ha sido proporcionado por los tutores, y procede de [77]. Está formado por 95 columnas y el nombre de cada dato, que se ha establecido como índice. Cada observación corresponde con un audio grabado de una persona distinta tal y como se puede ver en la Figura 6.1.

	frameTime	F0semitoneFrom27.5Hz_sma3nz_amean	F0semitoneFrom27.5Hz_sma3nz_stddevNorm	F0semitoneFrom27.5Hz_sma3nz_percentile20.0
name				
FC15075D0120R01	0.0	35.82298	0.245970	29.34496
FC15075D0220R01	0.0	37.60300	0.251011	37.71171
FC15075D0310R01	0.0	40.65500	0.064270	38.87238
FC15075D0320R01	0.0	40.90266	0.045094	39.26818
FC15075D0420R01	0.0	37.80817	0.164895	37.69392
...	...	...	...	...
MT65093D3100R02	0.0	26.03552	0.109192	24.05082
MT65093D3400R01	0.0	28.87394	0.035360	27.99399
MT65093D3400R02	0.0	27.61838	0.186418	25.31766
MT65093D3700R01	0.0	27.99758	0.118453	26.85425
MT65093D3700R02	0.0	27.79888	0.102926	26.92996

4174 rows × 95 columns

Figura 6.1: Conjunto de datos fea_gemaps.txt cargado con la aplicación desarrollada en este proyecto.

El resto de las columnas se refieren a características espectrales, de frecuencia, de energía y temporales del audio, tales como porcentaje de sonido, semitonos, silencios por segundo... Todas ellas son variables numéricas. La descripción de cada una de estas variables individualmente viene en [77].

Las dos últimas variables son categóricas y aluden a el sexo de la persona, ya sea masculino o femenino ([’F’, ’M’]) y al tipo de audio, ([’C’, ’D’, ’O’, ’T’]), donde ’C’ son muestras de niños sin discapacidad, ’D’ son muestras de personas con Síndrome de Down, ’O’ son muestras de personas con una discapacidad que no es Síndrome de Down y ’T’ son muestras de personas con desarrollo típico .

Otro conjunto de datos utilizado proporcionado por los tutores ha sido *ev_experts.csv* [77]. Este conjunto de datos consta de 966 observaciones y 3 columnas, una de las cuales se ha establecido como índice del conjunto de datos ya que es el identificador de cada audio (observación).

Las otras dos columnas se componen de dos variables categóricas (’evaluator’ y ’evaluation’) que ayudan a clasificar entre el tipo de evaluador y el tipo de evaluación (*wrong*, W o *right*, R), como se puede ver en la Figura 6.2.

La última base de datos proporcionada por los tutores ha sido *partidas_todas.csv* [78]. Este conjunto de datos está formado por 13 variables siendo la primera columna (id_partida) el identificador de cada dato, y un total de 26174 observaciones. Estos datos son registros de partidas jugadas en una aplicación educativa para la mejora de la pronunciación. Se han medido variables numéricas, como los puntos obtenidos en una partida, fechas en las que se ha realizado la partida y el correo de la persona que

	audioName	evaluator	evaluation
0	MD22022D0120R01	E01	W
1	MD22022D0120R02	E01	W
2	MD22022D0211R01	E01	R
3	MD22022D0220R01	E01	R
4	MD22022D0310R01	E01	R
...	...	...	...
961	MD11044D0420R01	E01	R
962	MD11044D0510R01	E01	R
963	MD11044D0510R02	E01	W
964	MD11044D0220R01	E01	W
965	MD11044D0310R01	E01	R

966 rows × 3 columns

Figura 6.2: Conjunto de datos *ev_experts.csv* cargado con la aplicación desarrollada en este proyecto.

ha jugado en ese momento. Como variable respuesta, se ha usado la última columna (completa), una variable discreta que proporciona información sobre si la partida a llegado a finalizarse por completo (1) o no (0) (Figura 6.3).

puntos_creador	puntos_retados	fecha_creador	fecha_creador_turno	fecha_retados_turno	posicion_creador	posicion_retados	puntos_con_extras_creador	punt
16	-1	2018-03-08 00:21:30	2018-03-08 00:23:10	NaN	14	13	17	
23	19	2018-03-15 15:54:35	2018-03-15 15:55:20	2018-03-15 20:25:42	3	6	23	
13	-1	2018-03-10 17:02:47	2018-03-10 17:04:50	NaN	262	252	23	
20	16	2018-03-14 14:25:44	2018-03-14 14:27:02	2018-03-14 20:03:45	26	16	30	
19	18	2018-03-10 21:00:54	2018-03-10 21:01:56	2018-03-10 23:22:11	79	68	30	
...	...	...	...	...	...	...	...	
16	9#13#18#14	2018-03-14 02:07:11	2018-03-14 02:08:43	2018-03-14 22:41:01#2018-03-16 12:34:31#2018-0...	202	184#186#189#185	16	
20	23	2018-03-17 17:10:47	2018-03-17 17:12:01	2018-03-17 18:36:24	9	5	20	
11	-1	2018-03-29 23:47:14	2018-03-29 23:48:48	NaN	281	222	70	
20	15	2018-03-09 22:37:02	2018-03-09 22:37:56	2018-03-10 10:06:56	131	123	28	
19	23	2018-03-20 21:42:45	2018-03-20 21:43:56	2018-03-20 21:45:55	33	25	19	

Figura 6.3: Conjunto de datos partidas_todas.csv cargado con la aplicación desarrollada en este proyecto.

Para finalizar, se ha creado un conjunto de datos con las librerías *random* y *pandas* de Python. Este conjunto de datos está formado por una columna de 20 observaciones numéricos y oscilan entre 0 y 6. El resto de las columnas son categóricas y toman diferentes valores ('SI', 'NO'), ('F', 'M') y (0, 1 o 2), respectivamente. El resultado de la ejecución del código se muestra en la Figura 6.4. Este conjunto de datos se ha creado para encontrar homocedasticidad y normalidad entre las observaciones y así poder aplicar los distintos test.

	num	grupo	grupo2	sujeto
0	4.824180	SI	F	0
1	5.341084	SI	F	0
2	1.643523	SI	F	0
3	4.150241	SI	F	0
4	2.417273	SI	F	0
5	1.664852	SI	F	1
6	2.417500	SI	F	1
7	0.519390	SI	F	1
8	4.506352	SI	M	1
9	2.735145	SI	M	1
10	5.468545	NO	M	1
11	2.162014	NO	M	1
12	2.073262	NO	M	1
13	0.119447	NO	M	1
14	4.322784	NO	M	1
15	0.331989	NO	M	1

Figura 6.4: Conjunto de datos homocedástico creado a partir de las librerías *random* y *pandas* de Python.

6.2. Pruebas de la biblioteca

Para la comprobación de que la librería funciona correctamente se ha creado un archivo en Python llamado *pruebas.py* ubicado en el directorio *pruebas*.

A continuación, se va a explicar cada método de forma individual:

- **pruebas_ficheros:** Método que comprueba el archivo *ficheros.py* llamando a cada una de las funciones que lo componen y devolviendo *DataFrames*.

  Este método devuelve los dos conjuntos de datos que se van a utilizar (*fea_gemaps.txt* y *ev_experts.csv*), dos fusiones de estos dos ficheros, la primera con solo las dos primeras columnas de ambos ficheros y la segunda con todas las columnas, y para finalizar escribe en un archivo llamado *resultados.txt* el resultado de cargar el fichero *fea_gemaps.txt*.

  Gracias a esta prueba se ha añadido a la función cargar_fichero un parámetro (*indice*), el cual permite establecer una columna o varias como índice del *DataFrame* que se va a devolver. Se establece por defecto a *False* no eligiendo ninguna columna como índice, introduciendo un número entero para establecer la columna a la que se refiere como indicador.

- **prueba_svm, prueba_arbol_decision, prueba_clustering:** Estas tres funciones se utilizan para comprobar el buen funcionamiento del archivo *clasificadores.py*

  Se ha utilizado la columna 'type' como variable dependiente y como variables independientes tres conjuntos de variables:

  - 3-12,33,34: Características de frecuencia.
  - 13-22,35,36: Características de energía.
  - 84-92: Características temporales.

  En las 3 pruebas se ha encontrado el mismo error para los 3 clasificadores (SVM, árbol de decisión y clustering): Al cargar la variable separadora de grupos (type) como variable categórica, la función solo reconocía variables numéricas (discretas o continuas) ya que las pruebas se realizaron con este tipo de conjunto de datos.

  Se ha añadido un pequeño fragmento de código para solucionar este problema que se puede ver en la Figura 6.5 , donde se ha comprobado si la variable dependiente es una variable categórica y si es así se ha convertido a una variable discreta numérica.

```
#Si la variable que separa grupos no es numerica, la convertimos en numerica
if (isinstance(y[0], str)):
     y = y.astype('category').cat.codes
```

Figura 6.5: Fragmento de código añadido para solucionar el problema de no aceptación de variables categóricas.

Otra modificación realizada ha sido la eliminación de la función que representaba el clasificador SVM dentro del archivo *funciones_graficas.py* ya que depende del conjunto de datos que se posea, no se puede generalizar para cualquier base de datos.

Por último, se intentó usar la columna 'sex' como variable independiente. Las 3 funciones nos proporcionaban un error, ya que esta variable es categórica. Para poder usarla se convirtió a numérica.

El resto de elementos funcionaban de forma correcta así como las funciones *seleccion_col* y *valores_nan* del archivo *utils.py*.

En las representaciones de los clasificadores se ha añadido el comando *plt.show()* en la representación del clustering, ya que si no se encontraba superposición de ambos gráficos.

- **prueba_correlacion, prueba_concordancia:** Se han creado estas dos funciones para la comprobación del archivo *correlacion_y_concordancia.py*.

  Se han realizado las pruebas con el archivo *fea_gemaps.txt* y con cada grupo de variables numéricas (frecuencia, energía y temporales). Además, para el análisis de concordancia se ha probado también con variables categóricas.

  Al realizar estas pruebas se han encontrado fallos en el archivo Python. En primer lugar, se ha modificado la función convertir_matriz del archivo *utils.py*, para que el *DataFrame* resultante de las funciones correlación y concordancia solo posea aquellas parejas de variables no repetidas (Figura 6.6).

```
#Se eliminan aquellas variables que sean las mismas
corr_mat = corr_mat.drop(corr_mat[corr_mat['level_0']== corr_mat['level_1']]
#Y se eliminan tambien las que se encuentren repetidas
corr_mat.reset_index(drop=True, inplace=True)
corr_mat = corr_mat.drop(list(range(1, corr_mat.shape[0], 2)))
```

Figura 6.6: Fragmento de código añadido para solucionar el problema de repetición de variables.

Asimismo, el coeficiente de correlación Kappa no señalaba bien si las variables eran categóricas, por lo que se le ha añadido una línea de código que lo comprueba correctamente (Figura 6.7).

```
elif ( (data.select_dtypes(include='object').shape[1] == 2) and (n == 2) ):

    kap = metrics.cohen_kappa_score(data.iloc[:, 0], data.iloc[:, 1])
    print('Las variables son categóricas. El índice de Kappa es: ')
    return kap
```

Figura 6.7: Fragmento de código añadido para solucionar la comprobación de variables categóricas.

- **prueba_est_descriptivos:** En esta función se ha probado el archivo *estadisticos_descriptivos.py* para el archivo de datos *fea_gemaps.txt* y sus respectivos conjuntos de variables. Los resultados se corresponden con los esperados y no se ha encontrado ningún error.

- **prueba_ganancia_info:** Se ha creado esta función para probar el archivo *ganancia_info.py*. Se ha mejorado la forma en la que la función mostraba los datos. Se ha creado un *DataFrame* cuyo índice es el nombre de la característica (variable) y la única columna corresponde a la ganancia de información de cada variable ordenadas de mayor a menor, tal y como se puede ver en la Figura 6.8.

Variables	Ganancia informacion
equivalentSoundLevel_dBp	0.5817
StddevVoicedSegmentLengthSec	0.4829
MeanVoicedSegmentLengthSec	0.2011
StddevUnvoicedSegmentLength	0.1457
MeanUnvoicedSegmentLength	0.0600
silencesPerSecond	0.0461
soundingPercentage	0.0454
silencePercentage	0.0454
silencesMean	0.0296

Figura 6.8: Resultado de la aplicación de la función ganancia_info (criterio Gini) a las variables temporales de *fea_gemaps.txt*.

- **prueba_regresion:** Se han realizado varias pruebas con los tres conjuntos de variables del archivo *fea_gemaps.txt* y con las variables numéricas del archivo *partidas_todas.csv*. Además, se ha añadido una función al archivo *funciones_graficas.py* que dibuja el diagrama de dispersión junto a la recta de regresión del modelo ajustado (ver ejemplo en la Figura 6.9).

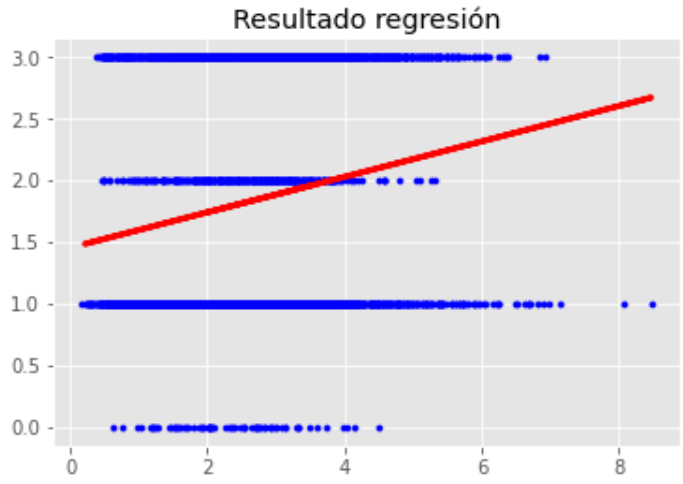


Figura 6.9: Resultado de la regresión lineal de la variable type en función de VoicedSegmentsPerSec del archivo *fea_gemaps.txt*.

	Ganancia información
F0semitoneFrom27.5Hz_sma3nz_percentile20.0	0.582
F0semitoneFrom27.5Hz_sma3nz_percentile50.0	0.582
F0semitoneFrom27.5Hz_sma3nz_percentile80.0	0.582
F0semitoneFrom27.5Hz_sma3nz_pctlrange0-2	0.582
F0semitoneFrom27.5Hz_sma3nz_meanFallingSlope	0.582
loudness_sma3_amean	0.582
loudness_sma3_stddevNorm	0.582
shimmerLocaldB_sma3nz_amean	0.582
shimmerLocaldB_sma3nz_stddevNorm	0.582
F0semitoneFrom27.5Hz_sma3nz_meanRisingSlope	0.581
F0semitoneFrom27.5Hz_sma3nz_stddevRisingSlope	0.57
F0semitoneFrom27.5Hz_sma3nz_stddevFallingSlope	0.569

Tabla 6.2: Tabla en formato LaTeX resultante de la aplicación de la ganancia de información en las variables de frecuencia.

- **prueba_random_forest, prueba_sequential_selection**: En estas dos funciones se ha probado el correcto funcionamiento del archivo *fea_gemaps.txt* y *partidas_todas.csv*, en cuanto a la selección de características. Se han realizado pruebas tanto con Random Forest, como con forward y backward 'selection', con diferentes números de características seleccionadas.

  Mediante esta el test de esta función también se prueba el método que dibuja el número de características en función del score que aporta al modelo.

- **prueba_tablas_latex:** En esta función se ha probado la capacidad de la librería de extraer código para crear tablas LaTeX. Se han creado tablas para LaTeX con algunos de los *DataFrames* resultantes de otras funciones como se puede ver en la Figura 6.2.

  Esta tabla se ha creado con Python y se le han añadido los bordes.

- **prueba_chi_square_test:** Se ha probado el test chi-cuadrado para determinar la dependencia de diferentes pares de variables.

  Al comprobar la relación entre las variables *type* y *sex* se ha obtenido la tupla de la Figura 6.10 donde se puede observar que, para todos los test, se rechaza H0 y se asume que entre las variables existe algún tipo de asociación.

```
[(sex            F            M
  type
  C      30.119310    38.880690
  D     938.936751  1212.063249
  O     159.763297   206.236703
  T     693.180642   894.819358,
  sex     F     M
  type
  C      29    40
  D     833  1318
  O     294    72
  T     666   922,
                  test    lambda        chi2  dof          pval    cramer  \
  0            pearson  1.000000  223.338332  3.0  3.811104e-48  0.231316
  1      cressie-read  0.666667  223.397737  3.0  3.700058e-48  0.231347
  2     log-likelihood  0.000000  230.474767  3.0  1.091853e-49  0.234982
  3      freeman-tukey -0.500000  242.862556  3.0  2.288040e-52  0.241215
  4  mod-log-likelihood -1.000000  262.860764  3.0  1.081325e-56  0.250950
  5             neyman -2.000000  335.533993  3.0  2.022163e-72  0.283525

     power
  0    1.0
  1    1.0
  2    1.0
  3    1.0
  4    1.0
  5    1.0  )]
```

Figura 6.10: Resultado de la aplicación del test Chi-Cuadrado entre dos variables del archivo *fea_gemaps.txt*.

- **prueba_mann_whitney:** Se ha probado la función que realiza un test de Mann Whitney con tres conjuntos de datos.

  El primer conjunto de datos es *fea_gemaps.txt*, donde la variable que separa los grupos tiene 4 niveles. Este test no se puede realizar con variables que no sean dicotómicas, por lo tanto, la función devuelve un mensaje de error advirtiendo al usuario de que debe introducir una variable separadora de grupos con dos niveles (*'La variable separadora de grupos solo puede tener dos niveles. Debe ser una variable binaria'*.

  El segundo conjunto de datos utilizado será *partidas_todas.csv.* La columna 'completa' se usará como variable separadora de grupos y se comprobará si la distribución de partida de ambos grupos es la misma. La función señala que el p-valor es 0 para las dos variables continuas comprobadas, por lo existen diferencias significativas. Sin embargo, se encuentra inexistencia de homocedasticidad en los datos, por lo que el uso de este test no sería válido.

  Por último, se ha creado un conjunto de datos con dos variables: una cualitativa que separa la variable numérica en dos categorías (SI o NO) y la otra con números con la misma varianza (Figura 6.11a).

  El test establece que existe igualdad de varianzas y de distribución de los datos y calcular el p-valor del test (0.54762) que no rechaza la hipótesis nula y asume que no existen diferencias significativas entre los datos (Figura 6.11b).

- **prueba_wilcoxon:** Se ha vuelto a probar este test con tres conjuntos de datos, dos de los cuales han sido usados anteriormente:

  Para el conjunto de datos *fea_gemaps.txt* la función devolverá la frase *'La variable que separa los*

	num	grupo
0	2.2	SI
1	4.1	SI
2	5.0	SI
3	6.1	SI
4	2.9	SI
5	6.2	NO
6	5.1	NO
7	2.0	NO
8	4.0	NO
9	6.5	NO

(a) Conjunto de datos creado

```
               W      pval  equal_var
levene  0.051095  0.826837       True
Es recomendable usar este test, ya que se supone igualdad de distribuciones de los datos.

[     U-val alternative     p-val   RBC  CLES
 MWU    9.0   two-sided  0.547619  0.28  0.36]
```

(b) Resultados del test

Figura 6.11: Test Mann Whitney

grupos solo puede tener dos niveles. Debe ser una variable binaria' ya que 'type' está formado por cuatro etiquetas.

Para una variable numérica del archivo *partidas_todas.csv* se realiza Wilcoxon y la función devuelve *'No se puede realizar el test, ya que los grupos no tienen la misma longitud'*, ya que el número de 0's y 1's de la variable 'completa' no es el mismo.

Para finalizar, se usa el conjunto de datos homocedástico creado anteriormente, ya que además contiene el mismo número de observaciones en cada grupo. El test asume homogeneidad de varianzas y de distribuciones de los datos pareados. Sin embargo, no se puede comprobar la normalidad de los datos si el número de muestras es menor que 8, por lo que se lanza una excepción que pide un dataset con mayor número de observaciones (*'No se puede comprobar la normalidad del test si tiene menos de 8 muestras.'*).

Como este último conjunto de datos no ha funcionado, se crea otro con 20 observaciones, números aleatorios y la misma longitud de grupos (Figura 6.12) y se obtiene un p-valor de 0.909722 no pudiendo rechazar la hipótesis nula y aceptando que las muestras proceden de poblaciones con la misma distribución de probabilidad.

```python
#Crear un conjunto de datos homocedastico
col1 = np.random.uniform(low=0, high=6, size=10).tolist()
col2 = np.random.uniform(low=0, high=6, size=10).tolist()
col3 = np.repeat(['SI', 'NO'], repeats = 10)
data3 = pd.DataFrame({'num': np.concatenate([col1, col2]),
                      'grupo': col3
})
```

Figura 6.12: Código para crear un conjunto de datos pareados y con la misma longitud de grupos.

- **prueba_kruskal_wallis:** Se ha probado esta función con los tres conjuntos de datos anteriores:

  Como primer conjunto de variables se han usado las variables temporales (*fea_gemaps.txt*) con 'type' como variable separadora de grupos. Ninguna de estas variables cumple los supuestos de homocedasticidad y misma distribución de los datos por lo que no es recomendable el uso de este test para estas variables. Aún así, la función imprime el resultado del test y de la prueba T por pares post-hoc.

  En segundo lugar, se han probado las variables 'puntos_creador' y 'posicion_creador' con la variable 'completa'. Ocurre lo mismo que en la anterior prueba, se realiza el test, pero sin el cumplimiento de los supuestos del modelo.

  Por último, se emplea el conjunto de datos creado mediante la librería *random* con n = 20 observaciones. En este caso si se cumplen los supuestos del modelo (homocedasticidad y distribuciones parecidas) y se obtienen los resultados de la Figura 6.13.

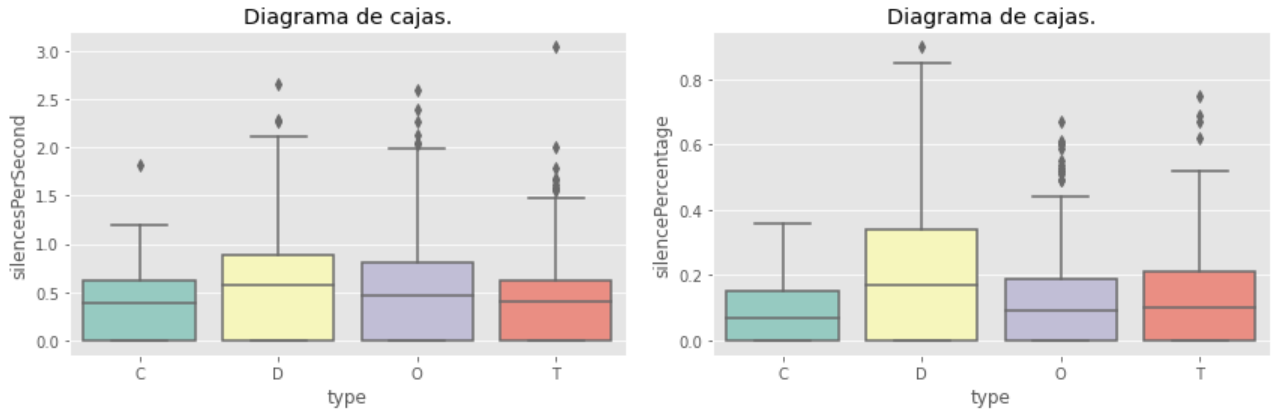
```
Columna num :
               W      pval  equal_var
levene  0.003927  0.950726       True
Es recomendable usar este test, ya que se supone igualdad de distribuciones de los datos.

([        Source  ddof1         H     p-unc
  Kruskal  grupo      1  0.142857  0.705457],
 [  Contrast   A   B  Paired  Parametric  U-val alternative    p-unc    hedges
  0    grupo  NO  SI   False       False   55.0   two-sided  0.73373  0.202852])
```

Figura 6.13: Resultado de la aplicación del test Kruskal-Wallis al *DataFrame* creado con la librería *random*.

- **prueba_anova_unidireccional:**

  Se han realizado tres pruebas:

  - Se ha realizado una prueba ANOVA unidireccional con dos variables dependientes (silencesPerSecond y silencePercentage) y una variable que separa los grupos (type). Para ambas variables no se cumple la normalidad ni la homocedasticidad, se realiza un ANOVA heterocedástico de Welch, el cual indica que existen diferencias significativas para ambas variables dentro de la variable 'type'. Se observan estas diferencias mediante un test T post-hoc. La función también devuelve dos gráficos de cajas de las variables dependientes en función del tipo de audio (Figura 6.14).
  - El siguiente conjunto de datos empleado ha sido el creado con la librería *random* con 20 observaciones. Estos datos cumplen la normalidad y la homocedasticidad. Se ha realizado un ANOVA y un test post-hoc con los resultados. El diagrama de cajas resultante se puede observar en la Figura 6.15.
  - La siguiente prueba realizada se ha llevado a cabo con el conjunto de datos *fea_gemaps.txt* donde se ha usado la variable 'sex' como variable sujeto, ya que se quiere realizar un ANOVA **dentro** de grupos. Para este ANOVA no es necesario que se cumpla el supuesto de homo-



(a) Diagrama de caja de la variable silencesPerSecond según el tipo de audio.

(b) Diagrama de caja de la variable silencePercentage según el tipo de audio.

Figura 6.14: Diagramas de caja resultado de un ANOVA delas variables *silencesPerSecond* y *silencePercentage* según el tipo.

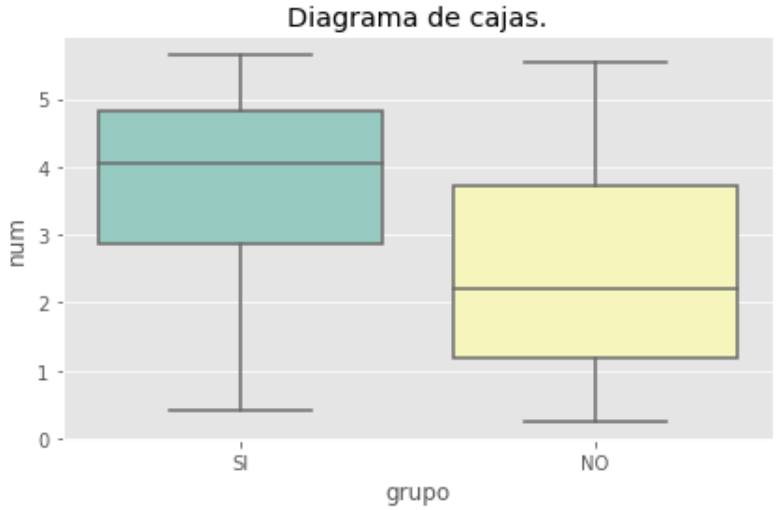


Figura 6.15: Diagrama de caja de la variable num según el grupo.

c

cedasticidad, pero sí el de esfericidad. Esta función devuelve el resultado del ANOVA dentro de grupos y del test post-hoc, así como el diagrama de caja (Figura 6.14).

- **prueba_anova_bidireccional:** Se han llevado a efecto varias pruebas:

  - Primeramente, se ha realizado un ANOVA bidireccional **entre** grupos (between) para el conjunto de datos *fea_gemaps.txt*, específicamente para los factores 'sex' y 'type'. En estos datos no existe normalidad ni homocedasticidad, por lo que la función ha efectuado un ANOVA de Welch, con sus posteriores comprobaciones post-hoc (Games-Howell). La función devuelve el ANOVA producido y las comprobaciones a posteriori, así como los diagramas de caja de cada uno de los factores por separado (Figura 6.14 y juntos (Figura 6.16).

  - Para el conjunto de datos homocedástico, se ha creado una nueva variable, 'grupo2', categórica con dos niveles ('F' y 'M') y se ha realizado un ANOVA bidireccional de estos dos

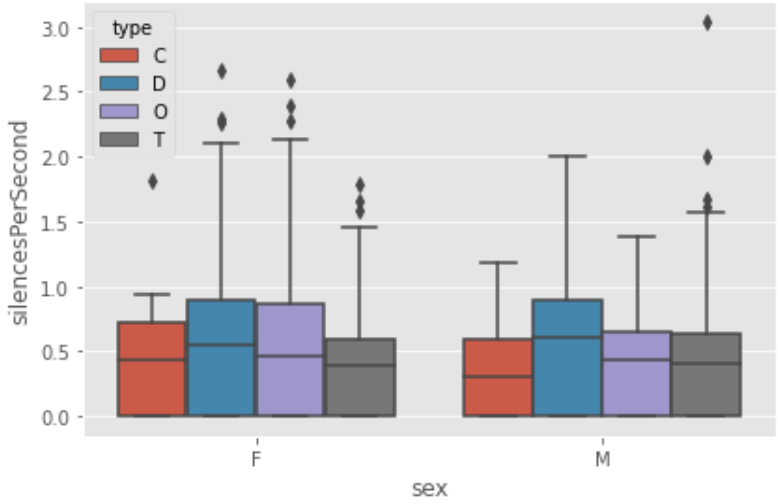


Figura 6.16: Diagrama de cajas de silencesPerSecond vs (sex and type).

factores sobre la variable numérica. Los datos son normales y con la misma varianza y se ha obtenido el resultado del ANOVA y de las comprobaciones post-hoc, asimismo los diagramas de cajas individuales y de la interacción.

6.3. Pruebas doctest

Para finalizar la validación de la aplicación se utiliza el módulo de Python ***doctest*** para la generación de pruebas basadas en el intérprete estándar de este lenguaje. Este módulo también se utiliza para escribir documentación para cada función, ilustrada generosamente con ejemplos de entrada y salida [58]. En la siguiente Figura 6.17 se puede ver un ejemplo de la función clasificador_cluster y cómo funciona en ella el módulo *doctest*.

```
def clasificador_cluster(data, col_x, col_y, size, k, estratg= 'k-means++',
                         n_init = 10, max_iter = 300, ignorar_nan = True):

    """El término clustering hace referencia a un amplio abanico de técnicas cuya finalidad es encontrar patrones o grupos
    (clusters) dentro de un conjunto de observaciones. En este caso vamos a utilizar Kmeans como algoritmo para la separación
    de los grupos, ya que es de los más rápidos y sencillos de implementar.

    Las variables introducidas como variables independientes deben ser variables numéricas, ya sean continuas o discretas,
    sino el algoritmo no lo soportará. En caso de tener variables categóricas, convertirlas primero a numéricas.

    Args:
        data (Dataframe): Conjunto de datos del que se quiere obtener la clasificación.
        col_x (str o all):Columnas de las variables que se van a utilizar como variables independientes.
        Las columnas seleccionadas se deben introducir entre comillas, separado entre comas ,
        tanto si se quiere un intervalo de columnas, como varias columnas individuales. Ejemplo: '3:8,2,9'.
        Si se desean todas las columnas, se introducirá la palabra reservada all.
        col_y (int): Columna en la que se encuentra la variable dependiente.
        size (float): Tamaño en el que se quiere dividir el conjunto de entrenamiento. Debe estar entre 0 y 1.
        k (int): Determina el número  K  de clusters que se van a generar.
        estratg (str): estrategia para asignar los centroides iniciales. Por defecto se emplea 'k-means++', una estrategia que
        trata de alejar los centroides lo máximo posible facilitando la convergencia. Sin embargo, esta estrategia puede
        ralentizar el proceso cuando hay muchos datos, si esto ocurre, es mejor utilizar 'random'.
        n_init (int): determina el número de veces que se va a repetir el proceso, cada vez con una asignación aleatoria
        inicial distinta. Es recomendable que este último valor sea alto, entre 10-25, para no obtener resultados subóptimos
        debido a una iniciación poco afortunada del proceso. Por defecto es 10.
        max_iter (int): número máximo de iteraciones permitidas. Por defecto es 300.
        ignorar_nan (bool): Parámetro que permite al usuario decidir si quiere ignorar los valores nan o sustituirlos por
        la media de cada grupo ( si existiesen )

    Return:
        La función devuelve el grafico que dibuja los clusters para cada una de las observaciones con el centroide
        de cada grupo.
        param(list): parámetros de la estimación
        predicciones(list): predicciones del modelo.

    PRUEBAS
    >>> clasificador_cluster(data, 3, 12, 2/3, 4)
    'Introduzca un valor válido de columnas'

    >>> clasificador_cluster(data, '3,8', -2, 2/3, 4)
    ('La variable respuesta no esta dentro del data set. Introduzca un valor entre 0 y', 12)

    >>> clasificador_cluster(data, '3,8', 12, 1.5, 4)
    'Introduza un valor de tamaño para el entrenamiento entre 0 y 1'

    >>> clasificador_cluster(data, '3,8', 12, 2/3, -2)
    'Introduzca una k mayor que 0 y que sea un número entero'

    >>> clasificador_cluster(data, '3,8', 12, 2/3, 4, 'kmeans++')
    'Introduzca una estrategia válido : k-means++ o random.'

    >>> clasificador_cluster(data, '3,8', 12, 2/3, 4, 'k-means++', 15, -3)
    'Introduzca un repetición/iteración válida'

    >>> data = pd.DataFrame()
    >>> clasificador_cluster(data, '3,8', 12, 2/3, 4)
    'No puede introducir un dataset vacío'

    """
```

Figura 6.17: Documentación y pruebas doctest en la función clasificador_cluster.

Tal y como se puede observar en la Figura 6.17 se realiza una pequeña explicación de lo que es el estadístico/clasificador/test que se va a llevar a cabo (en este caso clustering) y las restricciones que se necesitan para ejecutar la función. Después se explica cada el tipo y la definición de cada uno de los valores de entrada de la función, *Args*. Asimismo, se exponen los valores de salida (tipo y definición), y el *Return*. Finalmente, se realizan pruebas de correcto funcionamiento del método, sobre todo aquellas relacionadas con la validación de los argumento de entrada.

Capítulo 7

Conclusiones

En este capítulo, se exponen las diferentes ideas llevadas a cabo durante el desarrollo del proyecto, así como las posibles mejoras que se podrían llegar a realizar en un futuro.

En este TFG se ha desarrollado una aplicación que ayuda a los usuarios de Python a realizar análisis estadísticos. Mediante la implementación de diferentes funciones se ha logrado poner en práctica varias ramas de la estadística, desde la descriptiva a la inferencial pasando por el Machine Learning, todo ello unido a la visualización gráfica de los resultados de dichas funciones.

El trabajo ejecutado ha resultado útil para el aprendizaje de diversas herramientas de desarrollo que anteriormente no se dominaban, como el uso de repositorios (GitLab) o el lenguaje de programación *Python* con su software *Anaconda*. Al desarrollar código en Anaconda el alumno se ha dado cuenta de su facilidad de uso y rápida ejecución respecto a la programación de código directa. Además, esta herramienta permite la escritura de algoritmos complejos, proporciona un alto soporte para usuarios, contiene una interfaz de uso (GUI) bastante sencilla (Jupyter Notebook), pero potente, y simplifica la realización de proyectos relacionados con el análisis de datos [79].

El desarrollo del proyecto ha sido más largo y difícil de lo que se esperó en un primer momento, ya que han ido surgiendo más problemas de los esperados, tanto en la escritura del código como en las pruebas realizadas al final del proyecto. Sin embargo, gracias a ello se ha descubierto la importancia de desarrollo de una buena planificación de un proyecto a largo plazo, la elección de una buena metodología (iterativa e incremental) y el análisis de los riesgos que debe acarrear un proyecto.

A nivel personal, el proyecto ha resultado muy interesante desde su planificación hasta su implementación, pruebas y escritura de la memoria. Asimismo, se han aplicado conocimientos aprendidos durante el grado, como el desarrollo de código en Python o el desarrollo software enfocado al análisis estadístico. Todo ello ha ayudado al crecimiento personal y especialmente al profesional, ya que dentro del mundo laboral esta experiencia será de mucha utilidad.

7.1. Trabajo futuro

Después de la finalización del proyecto y de analizarlo de forma general, así como las ideas aportadas por los tutores, se han planteado algunas de las posibles mejoras para dar mayor calidad al proyecto:

- **Creación de interfaz:** Desarrollo de una interfaz gráfica de usuario (GUI) para que a los usuario inexpertos les resulte más sencillo el uso de la librería.
- **Mayor cantidad de gráficos:** Mejora de la experiencia visual de los resultados de los análisis de datos.
- **Integración de nuevas funcionalidades y test:** Implementación de nuevas y más complejas funcionalidades y análisis estadísticos, junto a sus pruebas pertinentes.

Apéndices

Apéndice A

Acrónimos

- **ANOVA**: ANalysis Of VAriance.
- **API**: Application Programming Interface.
- **GUI**: Interfaz Gráfica de Usuario.
- **ITACyL**: Instituto Tecnológico Agrario de Castilla y León.
- **IVA**: Impuesto sobre el Valor Añadido.
- **NaN**: Not a Number.
- **PC**: Personal Computer.
- **RF**: Requisito Funcional.
- **RFN**: Requisito No Funcional.
- **SO**: Sistema Operativo.
- **TFG**: Trabajo de Fin de Grado.
- **URL**: Uniform Resource Locator (Localizador de Recursos Uniforme).
- **Uva**: Universidad de Valladolid.

Apéndice B

Manual de despliegue

En esta sección de la memoria se van a mostrar todos los pasos para cada una de las instalaciones requeridas para el uso de la aplicación.

B.1. Instalación de Anaconda

1. Abrir el navegador que se esté utilizando, en este caso se ha llevado a cabo con *Google Chrome*.
2. Escribir en la barra de búsqueda la página oficial de descarga de Anaconda: `https://www.anaconda.com/products/individual` .
3. Descargar la versión de Anaconda que soporte el PC que se va a utilizar.
4. Ejecutar el archivo de instalación que se ha descargado en el ordenador e instalarlo en el PC que se ha empleado.
5. Abrir Anaconda Navigator y ejecutar Jupyter Notebook.

B.2. Instalación de Anaconda con más detalle

En primer lugar, es necesario entrar en la página oficial de descarga de Anaconda: `https://www.anaconda.com/products/individual` y elegir el sistema operativo compatible con nuestro ordenador así como la última versión de Python que soporte nuestro ordenador. Se guardará en el disco duro el archivo de instalación.

Se ejecutará el archivo y se notificará que Anaconda se ayuda de una suite de herramientas gráficas llamada “Anaconda Navigator”. En estas herramientas gráficas, se puede abrir un terminal para ejecutar los archivos, actualizar versiones, etcétera. En la Figura B.1, se puede ver el menú principal (GUI) de Anaconda Navigator una vez ya completada la instalación. En esta página se puede acceder y ejecutar

a diferentes IDE sin necesidad de usar comandos en una terminal. Para acceder a alguno de ellos, solo es necesario hacer clic en *launch* en la ventana correspondiente.

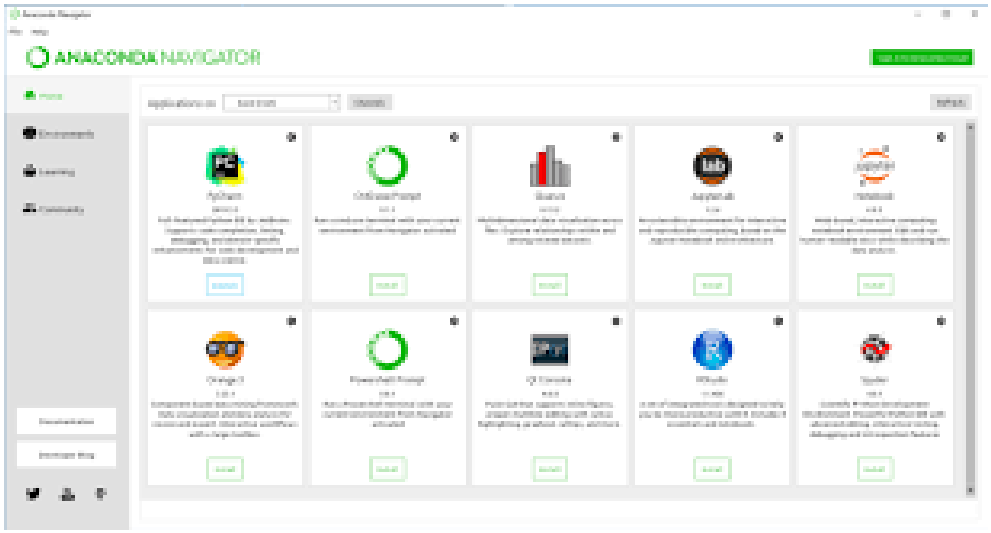


Figura B.1: Menú Principal de Anaconda Navigator

B.3. Instalación de Git y clonación del repositorio

1. Abrir una terminal en Linux, ya sea desde el propio SO o desde una máquina virtual.
2. Actualizar cualquier dependencia existente con *"sudo apt update"*
3. Instalar git mediante el comando *"apt-get install git"*.
4. Con el comando "*cd «directorio_proyecto»*" situarse sobre el directorio en el que se desea clonar el proyecto.
5. Ejecutar el comando *"git clone «gitlab_url_proyecto»"* (ver enlace URL en Apéndice D) para la clonación del proyecto.

B.4. Descarga del repositorio mediante la web GitLab

1. Abrir el navegador que se esté utilizando, en este caso se ha llevado a cabo con *Google Chrome*.
2. Escribir en la barra de búsqueda la URL donde se encuentra alojado el repositorio: `https://gitlab.inf.uva.es/elvdelg/integracion-de-herramientas-estadisticas.git` .
3. En la parte superior derecha de la página, descargar la librería en formato .zip.
4. Una vez guardada la librería en el PC, se puede ejecutar cualquier archivo.

B.5. Instalación de las librerías requeridas

1. Una vez clonado el repositorio del proyecto, entrar en él.
2. Situarse en el directorio en el que se encuentra la aplicación y comprobar si el archivo *requirements.txt* se encuentra en el entorno.
3. Usar el comando *"pip install -r /path/to/requirements.txt"* para la instalación de las librerías requeridas por el proyecto.
4. En caso de que el fichero *requirements.txt* no funcione en el entorno de su ordenador, instalar el fichero *requirements_alternative.txt* con el comando *"pip install -r /path/to/requirements_alternative.txt"*.
5. Ejecutar el comando *"pip list"* para verificar las librerías y su versión instaladas.

B.6. Funcionamiento de la biblioteca

1. Una vez instalado Anaconda y el repositorio, abra una terminal de Anaconda.
2. Ejecute el comando *"jupyter notebook –NotebookApp.iopub_data_rate_limit=1.0e10"*, para que el archivo *tablas_latex.py* funcione correctamente.

Apéndice C

Manual de uso

En este apéndice del TFG se va a describir los pasos necesarios para el uso de la librería y la ejecución de las pruebas.

En primer lugar, es necesario la instalación y descarga de los softwares, dependencias y repositorios reflejados en el Apéndice B.

Una vez realizada la instalación de GitLab y Anaconda (con sus respectivas librerías); y clonado el repositorio se abrirá una aplicación web de Jupyter Notebook. Los pasos a seguir serán los siguientes, con el SO Windows:

1. En el menú de Inicio busque y ejecute *'Anaconda Prompt'* (Figura C.1). Esta dependencia se instalará directamente con la instalación de Anaconda.

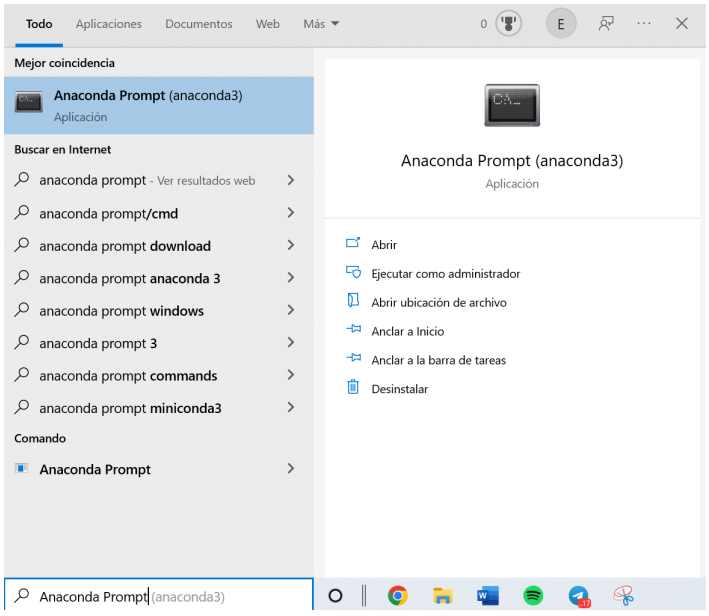


Figura C.1: Primer paso para la utilización de la librería

2. Una vez dentro de Anaconda Prompt escriba en la terminal *jupyter notebook* y pulse *enter* (Figura

C.2).

Figura C.2: Segundo paso para la utilización de la librería

3. Dentro de la página inicial de Jupyter entre en el repositorio que ha clonado o descargado.
4. Una vez dentro del proyecto, ya puede ejecutar el archivo que desee y observar sus resultados (Figura C.3)

Figura C.3: Cuarto paso para la utilización de la librería

5. En el caso de que quiera ejecutar, por ejemplo, las pruebas de proyecto, pulse y entre en el fichero

pruebas (Figura C.4) y en el archivo *pruebas.ipynb*.

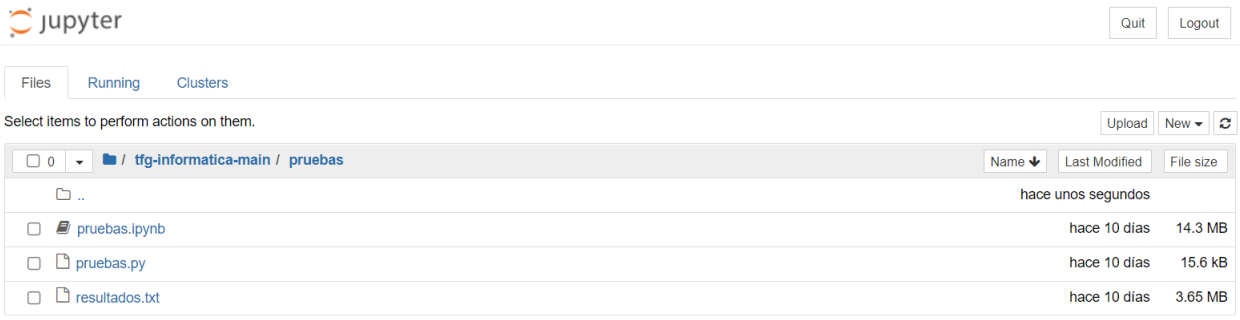


Figura C.4: Quinto paso para la utilización de la librería

6. Una vez dentro del archivo, el resultado de las pruebas se reflejará directamente debido a que ya han sido ejecutadas con anterioridad. Sin embargo, si quisiese volver a ejecutarlas, colóquese encima de la función que desee ejecutar y pulse *Run*. El resultado aparecerá a continuación (Figura C.5).

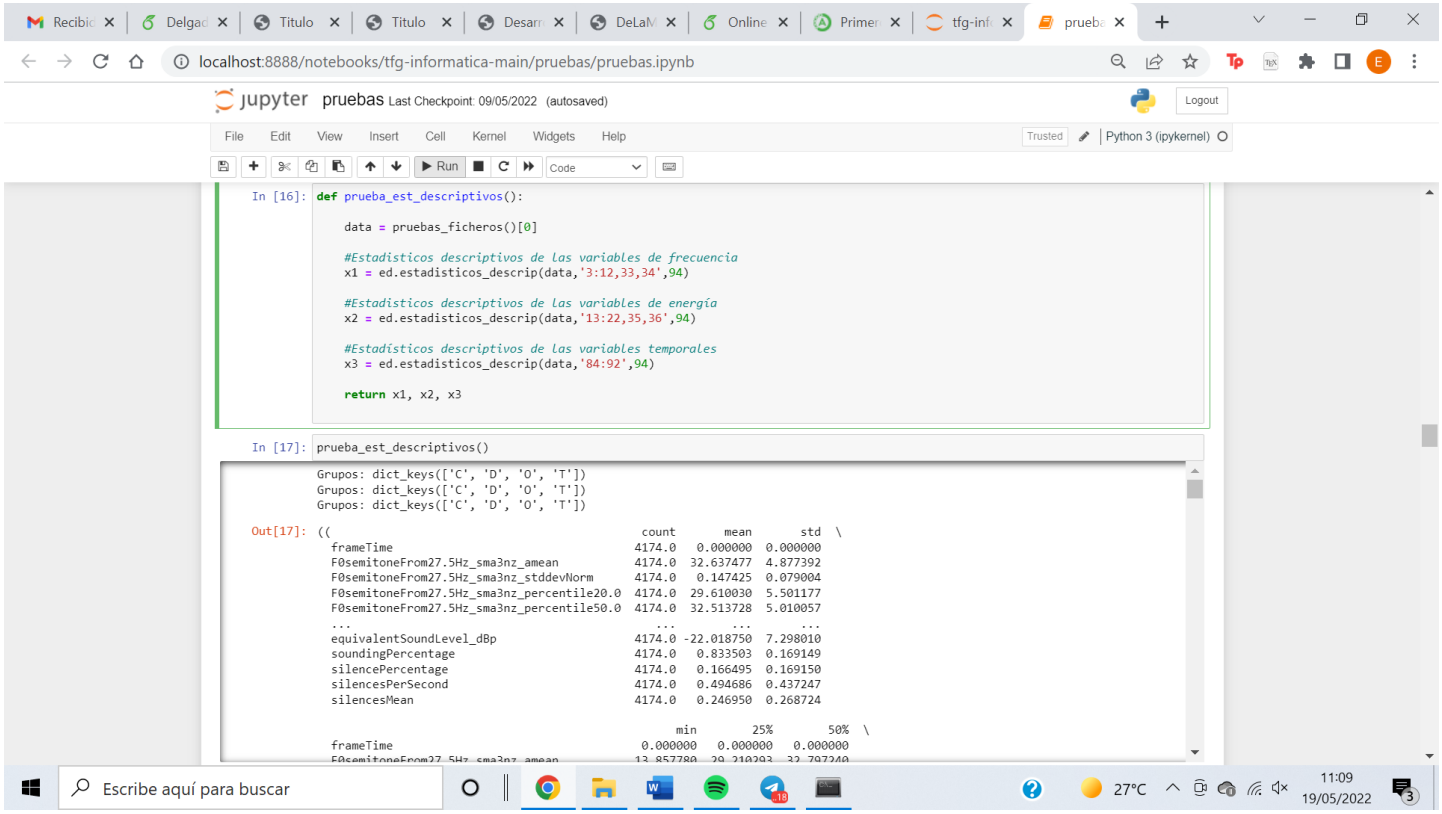


Figura C.5: Sexto paso para la utilización de la librería

C.1. Cómo ejecutar un test concreto

En esta sección se va a explicar como ejecutar un test concreto, en este caso se quiere utilizar un algoritmo para inducir un Random Forest en varios pasos:

1. En primer lugar, después de ejecutar todos los pasos del Manual de Despliegue (Apéndice B) se realizarán los pasos del 1 al 4 de la lista enumerada anteriormente C).

2. Entrará dentro del directorio *seleccion_caracteristicas* en el archivo *seleccion_caracteristicas.ipynb*.

    a) También se podría ejecutar el archivo *seleccion_características.py* desde la terminal de Linux con el archivo que se desee.

3. Una vez dentro del archivo, colocarse en una celda anterior a la función que se desea utilizar y llamar al conjunto de datos que se quiere emplear, en este caso el conjunto de datos *fea_gemaps.txt* que se encuentra dentro del directorio *datos* (Figura C.6).

```
data1 = fi.cargar_fichero("../datos/fea_gemaps.txt", ";", 0, 0)
```

Figura C.6: Tercer paso para la utilización de un test concreto.

4. Una vez cargado el conjunto de datos, llamar a la función que realiza un random forest desde otra celda con los argumentos que interesen (Figura C.7). En este caso, la variable dependiente se encontrará en la columna 94 y las variables independientes de las columnas 3 a la 12, la 33 y la 34. Se utilizará un tamaño de entrenamiento de 2/3, 100 estimadores y como criterio de selección *entropía*.

```
#Seleccion de caracteristicas de variables de frecuencia, con 100 estimadores y criterio entropía.
random_forest(data1, '3:12,33,34', 94, 2/3, 100, 'entropy')
```

Figura C.7: Cuarto paso para la utilización de un test concreto.

5. Por último, Anaconda mostrará el resultado de la función tal y como se puede ver en la Figura C.8. En este caso, esta función devuelve medidas de precisión del modelo: *accuracy*, *matriz de confusión*, *precisión*, *recall* y *f1*; en orden; un array de predicciones del modelo y por último, un *DataFrame* con la importancia de cada atributo ordenados de mayor a menos importancia (según el criterio entropía).

```
((0.8318965517241379,
  Predicción  0    1   2    3
  Real
  0           9    7   1    0
  1           4  671  10   62
  2           2   39  63    9
  3           0   98   2  415,
  0.7765424292193029,
  0.6977547092016888,
  0.7293731063537402),
 array([1, 1, 1, ..., 1, 3, 1], dtype=int8),
                                                 importancia
 atributos
 F0semitoneFrom27.5Hz_sma3nz_pctlrange0-2           0.152783
 F0semitoneFrom27.5Hz_sma3nz_percentile20.0         0.117820
 F0semitoneFrom27.5Hz_sma3nz_percentile50.0         0.105145
 loudness_sma3_amean                               0.097809
 F0semitoneFrom27.5Hz_sma3nz_percentile80.0         0.095366
 shimmerLocaldB_sma3nz_amean                       0.093985
 shimmerLocaldB_sma3nz_stddevNorm                  0.085538
 loudness_sma3_stddevNorm                          0.072446
 F0semitoneFrom27.5Hz_sma3nz_meanRisingSlope        0.051325
 F0semitoneFrom27.5Hz_sma3nz_stddevRisingSlope      0.044610
 F0semitoneFrom27.5Hz_sma3nz_meanFallingSlope       0.043385
 F0semitoneFrom27.5Hz_sma3nz_stddevFallingSlope     0.039789)
```

Figura C.8: Quinto paso para la utilización de un test concreto.

Apéndice D

Contenido del TFG

Este proyecto consta de la memoria aquí presentada en forma de PDF con el nombre *DelgadoDePazElvira_2022*.

Además, se adjunta el código del proyecto en Python que se encuentra en el **repositorio de la UVa de forma pública**, junto con la memoria en la siguiente URL:

`https://gitlab.inf.uva.es/elvdelg/integracion-de-herramientas-estadisticas.git`

Para la utilización de las pruebas del repositorio es necesario solicitar al grupo ECA-SIMM de la Universidad de Valladolid los ficheros de datos cuyo acceso es privado. Para ello contactar con `gir.ecasimm@uva.es`.

Bibliografía

[1] Deimos Estadística, "Análisis Estadísticos," https://www.deimosestadistica.com/analisis-estadisticos-de-datos/, (Accessed on 05/09/2022).

[2] KeepCoding Tech School, "¿Conoces los beneficios de Python y el Big Data?" https://keepcoding.io/aprender/python-y-el-big-data/#:~:text=Por%20ejemplo%2C%20la%20forma%20m%C3%A1s,y%20conjuntos%20de%20datos%20complejos., (Accessed on 05/09/2022).

[3] "Metodología iterativa o incremental en la gestión de proyectos," https://www.mundoerp.com/blog/metodologia-iterativa-o-incremental-gestion-proyectos/, (Accessed on 25/04/2022).

[4] Wikipedia, "Desarrollo iterativo y creciente," https://es.wikipedia.org/wiki/Desarrollo_iterativo_y_creciente, (Accessed on 04/25/2022).

[5] N. Jerez, "Diseño iterativo, la metodología que perfeccionará tus proyectos." https://medium.com/sue%C3%B1os-graficos/dise%C3%B1o-iterativo-la-metodolog%C3%ADa-que-perfeccionar%C3%A1-tus-proyectos-21034b0d277e, (Accessed on 25/04/2022).

[6] S. M. Ross, *Introducción a la estadística*. Reverté, 2018.

[7] A. V. Sabadías, *Estadística descriptiva e inferencial*. Univ de Castilla La Mancha, 1995, vol. 8.

[8] Wikipedia, "Estadística descriptiva," https://es.wikipedia.org/wiki/Estad%C3%ADstica_descriptiva#:~:text=La%20estad%C3%ADstica%20descriptiva%20es%20la,tablas%2C%20medidas%20num%C3%A9ricas%20o%20gr%C3%A1ficas., 2022, (Accessed on 04/25/2022).

[9] Wikipedia, "Correlación," https://es.wikipedia.org/wiki/Correlaci%C3%B3n, (Accessed on 05/09/2022).

[10] M. Attal, "Machine learning: definición, funcionamiento y usos," https://datascientest.com/es/machine-learning-definicion-funcionamiento-usos, 2022, (Accesed on 11/04/2022).

[11] M. de Asuntos Económicos y Transformación Digital, "¿Cómo aprenden las máquinas? Machine Learning y sus diferentes tipos," https://datos.gob.es/es/blog/como-aprenden-las-maquinas-machine-learning-y-sus-diferentes-tipos, 2022, (Accesed on 11/04/2022).

[12] Wikipedia, "Aprendizaje supervisado," http://es.wikipedia.org/w/index.php?title=Aprendizaje%20supervisado&oldid=133248401, 2022, (Accessed on 11/04/2022).

[13] Wikipedia, “Máquinas de vectores de soporte,” https://es.wikipedia.org/wiki/Scikit-learn, 2022, (Accessed on 11/04/2022).

[14] R. D. C. Huacac, “Uso de máquina de soporte vectorial para predicción de resistencia a la compresión de concreto,” http://repositorio.unsa.edu.pe/bitstream/handle/UNSA/9092/UPcrhurd.pdf?sequence=1&isAllowed=y, 2019, (Accessed on 05/03/2022).

[15] Wikipedia, “Árbol de decisión,” http://es.wikipedia.org/w/index.php?title=%C3%81rbol%20de%20decisi%C3%B3n&oldid=141428484, 2022, (Accessed on 11/04/2022).

[16] Wikipedia, “Aprendizaje basado en árboles de decisión,” http://es.wikipedia.org/w/index.php?title=Aprendizaje%20basado%20en%20%C3%A1rboles%20de%20decisi%C3%B3n&oldid=131532386, 2022, (Accessed on 11/04/2022).

[17] P. A. José Luis Bortolini, “Árboles de clasificación de potimirim mexicana,” https://www.scielo.cl/scielo.php?script=sci_arttext&pid=S0718-560X2013000400010, 2012, (Accessed on 05/23/2022).

[18] Wikipedia, “Regresión lineal,” http://es.wikipedia.org/w/index.php?title=Regresi%C3%B3n%20lineal&oldid=142521701, 2022, (Accessed on 11/04/2022).

[19] H. Chitarroni, “La regresión logística,” *IDICSO*, vol. 1, 2002.

[20] L. Gonzalez, “Aprendizaje no supervisado - Aprende IA,” https://aprendeia.com/aprendizaje-no-supervisado-machine-learning/, 2020, (Accessed on 11/04/2022).

[21] J. T. Cáceres, “Reconocimiento de patrones y el aprendizaje no supervisado,” *Universidad de Alcalá, Madrid*, 2007.

[22] R. Moya, “K-means en Python y Scikit-learn,” https://jarroba.com/k-means-python-scikit-learn-ejemplos/, 2016, (Accessed on 05/23/2022).

[23] L. Gonzalez, “Métodos de selección de características,” https://aprendeia.com/metodos-de-seleccion-de-caracteristicas-machine-learning/#:~:text=La%20Selecci%C3%B3n%20de%20Caracter%C3%ADsticas%20es,una%20mejor%20comprensi%C3%B3n%20del%20proceso, 2019, (Accessed on 18/04/2022).

[24] C. H. Enterprise, “Importancia y métodos de selección de características en Inteligencia Artificial,” https://forum.huawei.com/enterprise/es/importancia-y-m%C3%A9todos-de-selecci%C3%B3n-de-caracter%C3%ADsticas-en-inteligencia-artificial/thread/743403-100757, 2021, (Accessed on 05/03/2022).

[25] L. Breiman, “Random forests,” *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[26] A. Fernández, *Python 3 al descubierto*. Alfaomega Grupo Editor, 2013.

[27] Wikipedia, “Python,” https://es.wikipedia.org/wiki/Python, (Accessed on 05/09/2022).

[28] K. T. School, “Ventajas y Desventajas de Python,” https://keepcoding.io/blog/ventajas-y-desventajas-de-python/, 2022, (Accessed on 05/09/2022).

[29] Wikipedia, “Python Software Foundation License,” https://es.wikipedia.org/wiki/Python_Software_Foundation_License, (Accessed on 05/09/2022).

[30] Wikipedia, “R (lenguaje de programación),” https://es.wikipedia.org/wiki/R_(lenguaje_de_programaci%C3%B3n), (Accessed on 05/10/2022).

[31] Wikipedia, “R studio,” https://es.wikipedia.org/wiki/RStudio, (Accessed on 05/10/2022).

[32] J. Santaella, “¿Qué es la programación en R y cómo funciona?” https://talently.tech/blog/programacion-en-r/, 2022, (Accessed on 05/10/2022).

[33] A. Gil, “R–Ingenio en marcha,” https://agiltools.com/blogsp/analytics/r/, (Accessed on 05/10/2022).

[34] Wikipedia, “SPSS,” https://es.wikipedia.org/wiki/SPSS, (Accessed on 05/10/2022).

[35] Y. E. C. Vera, “SPSS: Conceptos, ventajas y desventajas,” http://correavera95.blogspot.com/2017/04/19-spss-conceptos-ventajas-y-desventajas.html, 2017, (Accessed on 05/10/2022).

[36] J. Samueza, “SPSS Conceptos Básicos,” https://sites.google.com/site/ticsyfinanzasjs21/manejo-de-datos-con-spss, 2016, (Accessed on 05/10/2022).

[37] Wikipedia, “SAS (lenguaje de programación),” https://es.wikipedia.org/wiki/SAS_(lenguaje_de_programaci%C3%B3n), (Accessed on 05/10/2022).

[38] “Revisión de SAS: Qué es, pros, contras e idoneidad,” https://www.newgenapps.com/es/blogs/sas-review-what-is-it-pros-cons-and-suitability/, 2018, (Accessed on 05/10/2022).

[39] S. Institute, “SAS logo,” https://es.m.wikipedia.org/wiki/Archivo:SAS_logo_horiz.svg, (Accessed on 05/10/2022).

[40] A. S. Romero, “Aplicación web para el análisis estadístico de datos de la calidad del agua y del aire,” https://reunir.unir.net/bitstream/handle/123456789/6096/SANCHEZ%20ROMERO%2C%20ALBERTO.pdf?sequence=1&isAllowed=y, 2017, (Accessed on 04/25/2022).

[41] A. de Osset Greño, “Desarrollo de una herramienta para el análisis de rendimiento del alumnado del Grado en Ingeniería Informática de la ETSINF,” https://m.riunet.upv.es/bitstream/handle/10251/126079/Osset%20-%20Desarrollo%20de%20una%20herramienta%20para%20el%20an%C3%A1lisis%20de%20rendimiento%20del%20alumnado%20del%20Grado%20en%20I....pdf?sequence=1&isAllowed=y, 2019, (Accessed on 04/25/2022).

[42] A. Schlegel, “Hypothesis and statistical testing in python,” https://github.com/aschleg/hypothetical/tree/master/hypothetical, 2020, (Accessed on 04/25/2022).

[43] C. Fonnesbeck, “Statistical data analysis in python,” https://github.com/fonnesbeck/statistical-analysis-python-tutorial, 2015, (Accessed on 05/04/2022).

[44] S. Patel, “Statistics for Data Science using Python,” https://github.com/suneelpatel/Statistics-for-Data-Science-using-Python, 2019, (Accessed on 05/04/2022).

[45] “Anaconda Navigator,” https://docs.anaconda.com/anaconda/navigator/index.html, 2022, (Accessed on 25/04/2022).

[46] Astah, “The power of software modeling,” https://astah.net/, 2022, (Accessed on 11/04/2022).

[47] G. Chrome, “Navegador Web Google Chrome,” https://www.google.com/intl/es/chrome/, 2022, (Accessed on 05/25/2022).

[48] M. 365, “Microsoft Excel,” https://www.microsoft.com/es-es/microsoft-365/excel, (Accessed on 05/25/2022).

[49] “Gitlab, Escuela de Ingeniería Informática,” https://gitlab.inf.uva.es/users/sign_in, 2022, (Accessed on 05/25/2022).

[50] Google, “Plataforma de almacenamiento personal en la nube y uso compartido de archivos - Google Drive,” https://www.google.com/intl/es_es/drive/, 2022, (Accessed on 05/25/2022).

[51] L. S. Inc., “Software de Diagramas Online | Lucidchart,” https://www.lucidchart.com/pages/es, 2022, (Accessed on 05/25/2022).

[52] M. Firefox, “Descarga Navegador Firefox,” https://www.mozilla.org/es-ES/firefox/new/, 2022, (Accessed on 05/03/2022).

[53] J. Hammersley, “Overleaf, Editor de LaTeX Online,” https://es.overleaf.com/project, 2022, (Accessed on 05/25/2022).

[54] N. y Pável Dúrov, “Telegram web,” https://web.telegram.org/k/, 2022, (Accessed on 05/25/2022).

[55] M. y. R.-M. Rolon-Mérette, Damien y Ross, “Introduction to Anaconda and Python: Installation and setup,” *Python for Research in Psychology*, vol. 16, no. 5, pp. S5–S11, 2016.

[56] Talent.com, “Salario programador web en españa,” https://es.talent.com/salary?job=programador+web#:~:text=Descubre%20cu%C3%A1l%20es%20el%20salario%20medio%20para%20Programador%20Web&text=El%20salario%20programador%20web%20promedio,hasta%20%E2%82%AC%2033.300%20al%20a%C3%B1o., 2022, (Accessed on 05/23/2022).

[57] “¿Qué son los requisitos funcionales? Especificación, tipos, EJEMPLOS,” https://ebooksonline.es/que-es-un-requisito-funcional-especificacion-tipos-ejemplos/, (Accessed on 05/15/2022).

[58] P. S. Fundation, “doctest – Prueba ejemplos interactivos de Python,” https://docs.python.org/es/3/library/doctest.html, 2022, (Accessed on 05/25/2022).

[59] NumFOCUS, “pandas - Python Data Analysis Library,” https://pandas.pydata.org/, (Accessed on 05/25/2022).

[60] NumFOCUS, “Numpy,” https://numpy.org/, 2022, (Accessed on 05/25/2022).

[61] NumFOCUS, “Scipy,” https://scipy.org/, 2022, (Accessed on 05/25/2022).

[62] S. learn Developers, “Installing scikit-learn,” https://scikit-learn.org/stable/install.html, 2022, (Accessed on 05/25/2022).

[63] W. McKinney, *Python for Data Analysis*. O’Reilly Media, Inc., 2012.

[64] M. D. Team, “Installation—Matplotlib,” https://matplotlib.org/stable/users/installing/index.html, 2012, (Accessed on 05/25/2022).

[65] N. Ari and M. Ustazhanov, “Matplotlib in python,” in *2014 11th International Conference on Electronics, Computer and Computation (ICECCO)*. IEEE, 2014, pp. 1–6.

[66] P. S. Fundation, “seaborn · PyPI,” https://pypi.org/project/seaborn/, 2022, (Accessed on 05/25/2022).

[67] M. L. Waskom, “Seaborn: statistical data visualization,” *Journal of Open Source Software*, vol. 6, no. 60, p. 3021, 2021.

[68] P. S. Fundation, “collection · PyPI,” https://pypi.org/project/collection/, 2022, (Accessed on 05/25/2022).

[69] P. S. Fundation, “pingouin · pypi,” https://pypi.org/project/pingouin/, 2022, (Accessed on 05/25/2022).

[70] P. S. Fundation, “xdoctest · pypi,” https://pypi.org/project/xdoctest/, 2022, (Accessed on 05/25/2022).

[71] P. S. Fundation, “mlxtend · pypi,” https://pypi.org/project/mlxtend/, 2022, (Accessed on 05/25/2022).

[72] S. Raschka, “Mlxtend: Providing machine learning and data science utilities and extensions to python’s scientific computing stack,” *Journal of open source software*, vol. 3, no. 24, p. 638, 2018.

[73] P. S. Fundation, “tabulate · pypi,” https://pypi.org/project/tabulate/, 2022, (Accessed on 05/25/2022).

[74] P. S. Fundation, “linearmodels · pypi,” https://pypi.org/project/linearmodels/, 2022, (Accessed on 05/25/2022).

[75] P. S. Fundation, “rpy2 · pypi,” https://pypi.org/project/rpy2/, 2022, (Accessed on 05/25/2022).

[76] P. S. Fundation, “random2 · pypi,” https://pypi.org/project/random2/, 2022, (Accessed on 05/25/2022).

[77] D. Escudero-Mancebo, M. Corrales-Astorgano, V. Cardeñoso-Payo, L. Aguilar, C. González-Ferreras, P. Martínez-Castilla, and V. Flores-Lucas, “Prautocal corpus: A corpus for the study of down syndrome prosodic aspects,” *Language Resources and Evaluation*, pp. 1–34, 2021.

[78] C. Tejedor-García, D. Escudero-Mancebo, V. Cardeñoso-Payo, and C. González-Ferreras, “Using challenges to enhance a learning game for pronunciation training of English as a second language,” *IEEE Access*, vol. 8, no. 1, pp. 74 250–74 266, 4 2020.

[79] T. School, “Ciencia de datos: Anaconda Python,” https://www.tokioschool.com/noticias/ciencia-datos-anaconda-python/#:~:text=Facilita%20la%20escritura%20de%20algoritmos,proyectos%20de%20Ciencia%20de%20Datos., 2022, (Accessed on 05/19/2022).