

PROYECTO DATA SCIENTIST PARA ESTUDIO DE LOS GENES DE LA PLANTA ARABIDOPSIS THAILIANA

**CURSO ACADÉMICO: 2018/2020**

**AUTOR: Miguel Barrios Álvarez**

**DOCENTE ACADÉMICO: Juan Manuel Moreno Lamparero**

# Índice

[1. Introducción 7](#_Toc39756640)

[1.1 Antecedentes 7](#_Toc39756641)

[1.2 Objetivo 8](#_Toc39756642)

[1.3 Descripción 8](#_Toc39756643)

[2. Conceptos previos 8](#_Toc39756644)

[3. Metodología 10](#_Toc39756645)

[4. Entorno de trabajo 11](#_Toc39756646)

[4.1 Equipo de trabajo máquina local 11](#_Toc39756647)

[4.2 Equipo de trabajo en máquina virtual 11](#_Toc39756648)

[Componentes 13](#_Toc39756649)

[5. Datos 13](#_Toc39756650)

[5.1 Descripción de los datos 13](#_Toc39756651)

[5.2 Organización de los datos para el programa 16](#_Toc39756652)

[6. Desarrollo Data Science 17](#_Toc39756653)

[6.1 FASE I: Extracción, tratamiento y carga de los datos 17](#_Toc39756654)

[6.2 FASE II: Análisis exploratorio de los datos 20](#_Toc39756655)

[6.3 Fase III: Ingeniería de características y selección de modelos 24](#_Toc39756656)

[6.3.1 Filter Based 25](#_Toc39756657)

[6.3.2 Wrapper-Based 28](#_Toc39756658)

[6.3.3 Embedded Method: Lasson 30](#_Toc39756659)

[6.3.4 Conclusión análisis de selección de características 32](#_Toc39756660)

[6.4 FASE IV: Implementación de modelos Machine Learning 33](#_Toc39756661)

[6.4.1 Algoritmos Machine Learning 34](#_Toc39756662)

[6.4.2 Scikit-learn 36](#_Toc39756663)

[6.4.3 Overfitting/Underfitting 36](#_Toc39756664)

[6.4.4 Score(R2) 38](#_Toc39756665)

[6.4.5 Code Machine Learning 38](#_Toc39756666)

[6.5 Fase V: Análisis de los resultados 46](#_Toc39756667)

[7. Conclusiones 47](#_Toc39756668)

[ANEXOS I: Desarrollo ETL 48](#_Toc39756669)

[1. Introducción 48](#_Toc39756670)

[2. Anaconda 49](#_Toc39756671)

[2.1 Notebook Jupyter 51](#_Toc39756672)

[3. Librerías Python 52](#_Toc39756673)

[3.1. Pandas 52](#_Toc39756674)

[3.2. Librería Numpy 53](#_Toc39756675)

[3.3. Librería Re 53](#_Toc39756676)

[Librería Zipfile/Os 54](#_Toc39756677)

[3.4. Desarrollo ETL 55](#_Toc39756678)

[Importación de librerías 55](#_Toc39756679)

[Función extract() 56](#_Toc39756680)

[Función tranform load() 57](#_Toc39756681)

[Función clean\_duplicate() 58](#_Toc39756682)

[Anexo II: Ingeniería de características y selección de modelos 60](#_Toc39756683)

[1. Introducción 60](#_Toc39756684)

[2. Desarrollo Ingeniería de características y selección de modelos 60](#_Toc39756685)

[ANEXO III: Implementación de modelos Machine Learning 63](#_Toc39756686)

[1. Introducción 63](#_Toc39756687)

[2. Desarrollo de algoritmos Machine Learning 63](#_Toc39756688)

[Bibliografía 65](#_Toc39756689)

[Referencia 65](#_Toc39756690)

# Índice de Figuras

[Figura 1. Arabidopsis Thaliana 7](#_Toc39756691)

[Figura 2: Página oficial de descargas VMware 12](#_Toc39756692)

[Figura 3: Página oficial de descargas Ubuntu 12](#_Toc39756693)

[Figura 4: Expresión del gen 14](#_Toc39756694)

[Figura 5: Ficheros de expresión 15](#_Toc39756695)

[Figura 6: Organización de los datos 16](#_Toc39756696)

[Figura 7: Programa Machine Learning 17](#_Toc39756697)

[Figura 8: Código Python expresión regular 18](#_Toc39756698)

[Figura 9: Código Python separación de nombre fichero 19](#_Toc39756699)

[Figura 10: Código Python limpieza de valores nulos y duplicados 20](#_Toc39756700)

[Figura 11: Output función ETL Python 20](#_Toc39756701)

[Figura 12: EDA 21](#_Toc39756702)

[Figura 13: Descripción valores estadísticos 22](#_Toc39756703)

[Figura 14: Summary BerlethS1 22](#_Toc39756704)

[Figura 15: Ejemplo gráfica Boxplot 23](#_Toc39756705)

[Figura 16: Gráfica Boxplot Ensayos 23](#_Toc39756706)

[Figura 17: Barplot Mean 24](#_Toc39756707)

[Figura 18: Diferencias entre los métodos Feature Selection 25](#_Toc39756708)

[Figura 19: Matriz de correlación de Pearson 26](#_Toc39756709)

[Figura 20: Cuadro resumen correlación >0,4 27](#_Toc39756710)

[Figura 21: Cuadro resumen correlación < -0,4 28](#_Toc39756711)

[Figura 22: Valor óptimo features RFE 29](#_Toc39756712)

[Figura 23: Output algoritmo RFE 30](#_Toc39756713)

[Figura 24: Método Lasso Scikit-Learn 31](#_Toc39756714)

[Figura 25: Gráfica análisis Lasso 31](#_Toc39756715)

[Figura 26: Variables con coeficiente Lasso 0 32](#_Toc39756716)

[Figura 27: Cuadro resumen resultado coeficientes selección de características 33](#_Toc39756717)

[Figura 28: Red Neuronal Multicapa 35](#_Toc39756718)

[Figura 29: Esquema de los modelos disponibles en Scikit-Learn 36](#_Toc39756719)

[Figura 30: Equilibrio del aprendizaje 37](#_Toc39756720)

[Figura 31: Matriz Output Random Forest I 39](#_Toc39756721)

[Figura 32: Gráfica RFI Score\_train Score\_test 39](#_Toc39756722)

[Figura 33: Matriz Output Random Forest II 40](#_Toc39756723)

[Figura 34: Gráfica RF II Score\_train Score\_test 41](#_Toc39756724)

[Figura 35: Matriz Output Random Forest III Fuente: Elaboración propia 41](#_Toc39756725)

[Figura 36: Gráfica RF III Score\_train Score\_test 42](#_Toc39756726)

[Figura 37: Matriz MLP I Output 43](#_Toc39756727)

[Figura 38: Gráfica MLP I Score\_train Score\_Test 43](#_Toc39756728)

[Figura 39: Matriz MLP II Output 44](#_Toc39756729)

[Figura 40: Gráfica MLP II Score\_train Score\_test 44](#_Toc39756730)

[Figura 41: Matriz MLP III Output 45](#_Toc39756731)

[Figura 42: Gráfica MLP III Score\_train Score\_test 46](#_Toc39756732)

[Figura 43: Cuadro Resumen resultados 47](#_Toc39756733)

[Figura 44: Página principal descarga Anaconda 50](#_Toc39756734)

[Figura 45: Navegador Anaconda 51](#_Toc39756735)

[Figura 46: Jupyter Notebook 51](#_Toc39756736)

[Figura 47: Dataframe 52](#_Toc39756737)

[Figura 48: Principales funcionalidades de librería 'Os' 55](#_Toc39756738)

[Figura 51: Genes solapados 58](#_Toc39756739)

**Resumen**

El aprendizaje automático —o “aprendizaje máquina”, del inglés machine learning— es una rama de la Inteligencia Artificial en la que se recogen las diferentes técnicas para dotar a un ordenador de la capacidad de “aprender” patrones a partir de conjuntos de datos de ejemplo. Para ello, es necesario crear modelos que sean una abstracción de los datos. Estos modelos pueden ser de diferentes tipos: fórmulas matemáticas, conjuntos de reglas o estructuras de conexiones. Una vez obtenidos, los modelos pueden ser utilizados para aumentar la eficacia con la que se resuelven múltiples problemas ya conocidos o para resolver nuevos a través de predicciones, reconocimiento de patrones o buscando grupos en los datos.

Con el continuo aumento de los sistemas Big Data, y gracias al abaratamiento de los costes de almacenamiento y captura de datos, se hace necesario disponer de algoritmos de aprendizaje automático que permitan descubrir la información oculta en estos sistemas para poder ponerla en valor.

En el siguiente proyecto tiene el objetivo de desplegar las diferentes técnicas de un Data Scientist, con el fin de obtener un valor añadido de los datos que son en este caso los averiguar el nivel de relación de los genes de la planta Thailiana.

Para ello se ha requerido combinar los conocimientos de programación con lenguaje Python y R junto la capacidad analítica a través de conocimientos matemáticos y estadísticos.

# ****Introducción****

A lo largo de este proyecto se va a detallar los antecedentes de la empresa que han motivado la realización de este proyecto, la descripción del proyecto, el objetivo que se intenta lograr y finalmente los métodos que se han llevado a cabo para conseguir los mejores resultados.

## Antecedentes

Arabidopsis Thaliana es una planta herbácea de pequeño tamaño (10-30 cm) con hojas en la base del tallo formando una roseta a su alrededor y alguna pequeña hoja aislada a lo largo de la planta. Sus flores presentan 4 pétalos en forma de cruz (de ahí su pertenencia a las crucíferas) y se acumulan en racimos al final de los tallos, terminando por formar silicuas tabicadas repletas de pequeñas semillas (0,5 mm de diámetro) ovoideas.



Figura 1. Arabidopsis Thaliana

Su ciclo de vida dura unas 5-6 semanas en la naturaleza, con una única generación al año, por lo general. Pero en condiciones de laboratorio se pueden conseguir hasta 6 generaciones al año. Además, es una planta que presenta un genoma muy pequeño (130 megabases) distribuido en tan sólo 5 cromosomas, lo cual facilita mucho su utilización en estudios genéticos.

Actualmente se dispone de la secuencia genómica completa y una enorme colección de mutantes que proporcionan un recurso de investigación único para plantas superiores.

Es por ello que actualmente se dispone de numerosos de ensayos y proyectos de investigación sobre la planta para extraer información de la naturaleza vegetal registrados en diferentes formatos disponibles para un usuario.

## Objetivo

El siguiente proyecto tiene como objetivo averiguar el nivel de relación de los genes de la planta Arabidopsis Thailiana en dependencia de los ensayos que se han realizado, obteniendo así diferentes niveles de expresión a través de métodos Data Science.

Como se ha comentado en el apartado anterior, Arabidopsis Thaliana es una planta herbácea que presenta un genoma muy pequeño distribuido 5 cromosomas, lo cual facilita mucho su utilización en estudios genéticos.

Por ello el objetivo del proyecto es:

* En primer lugar, crear un modelo capaz de predecir el nivel de expresión de un gen según los ensayos.
* En segundo lugar, interpretar la magnitud de error asociado a los valores.

De esta manera se espera que, si todos los genes son independientes, maximizaría el procesamiento de la información del sistema y, por lo tanto, esperaríamos un error de predicción muy alto. Al mismo tiempo, los genes se organizan en redes para evitar redundancias (es decir, varios genes pueden participar en las mismas funciones), por lo que este error puede no ser tan alto.

## Descripción

Se pretende seleccionar un conjunto de experimentos, y extraer a su vez un conjunto de genes con el fin de obtener el nivel de expresión.

Posteriormente se quiere aplicar diferentes métodos matemáticos, estadísticos y técnicas Machine Learning con el fin de sacar conclusiones a través de coeficientes y gráficas.

Por ello el proyecto se divide en las siguientes fases:

* FASE I: Extracción, tratamiento y carga de los datos
* FASE II: Análisis exploratorio de datos
* FASE III: Ingeniería de características y selección de modelos
* FASE IV: Implementación de modelos Machine Learning
* FASE V: Análisis de resultados

Tras realizar todos los procesos, las conclusiones de los resultados deben de responder a la pregunta de si los genes condicionados por sus ensayos presentan relación según los valores de correlación de los mismos.

Esto se traduce que los genes dependiendo del tipo de ensayo que se realice, se puede hacer una predicción de las expresiones.

# ****Conceptos previos****

En este apartado se van a explicar los conceptos necesarios para el entendimiento del proyecto:

**Variables continuas**

Los valores pertenecen a un intervalo numérico, esto es, conforman un continuo numérico en un rango de valores posibles. Ejemplos: altura de una persona, tiempo que transcurre hasta que una máquina falla, dinero que gana un agente automático de trading, precio de la electricidad, etc.

**Variables discretas**

Los valores son aislados. Ejemplos: tirada de un dado, número de conexiones de un router, número de veces en que una moneda cae de cara al lanzarla 10 veces, un paciente tiene una enfermedad o no, etc.

**Variables dependientes**

Una variable dependiente, se denomina como la representación emblemática de un determinado evento que no se encuentre especificado, el cual pertenezca a un funcionamiento que se obtenga de diferentes valores.

**Variables independientes**

Es conocida también con el nombre de variable explicativa y la dependiente como variable explicada, lo que significa que las variaciones de esta variable repercuten en las variantes de la variable dependiente.

**EDA**

Las siglas EDA se corresponde a *Análisis exploratorio de datos* se refiere al proceso critico de descubrir patrones, anomalías, mostrar hipótesis y verificar los supuestos a través de análisis estadísticos y representación de gráficas.

**Outliers**

En estadística, tales como muestras estratificadas, un valor atípico (en inglés outlier) es una observación que es numéricamente distante del resto de los datos. Las estadísticas derivadas de los conjuntos de datos que incluyen valores atípicos serán frecuentemente engañosas.

**Features Selection**

Selección de características es el proceso donde automáticamente o manualmente selecciona las características que contribuyen a una mayor predicción.

El proceso de selección de las características que han de ser incluidas en un modelo es fundamental. Antes de la creación de un modelo no se puede saber cuáles son las más significativas, las menos significativas ni tampoco las que simplemente son ruido. Una selección poco exigente puede terminar en un modelo con demasiadas características que puede acabar en sobreajuste. Por otro lado, una eliminación excesiva puede llevar a que no se consigan obtener modelos útiles.

**Modelo supervisado**

Modelos que se caracterizan por ser entrenados mediante conjuntos de datos en los que la solución al problema planteado es conocida, es decir, el conjunto de entrenamiento se divide en una o varias variables independientes y una dependiente (la solución del problema). Estos modelos permiten aprender patrones basándose en la experiencia.

**Modelo no supervisado**

En este tipo de modelos, los conjuntos de datos usados para entrenar no disponen de —o no utilizan— la solución del problema, como sucede en los modelos supervisados, por lo que son especialmente eficaces para la identificación de patrones desconocidos.

**Modelo Conexionista**

El término Deep Learning hace referencia al conjunto de algoritmos que se han desarrollado para el entrenamiento de redes neuronales profundas, también conocidos como sistemas conexionistas. Las redes neuronales profundas son aquellas que tienen múltiples capas ocultas, entendiéndose por capas ocultas aquellas que no forman parte de la entrada y salida del sistema, es decir, aquellas capas que no reconocen directamente los datos de entrada ni de salida.

En los últimos años, el estudio de las redes neuronales se ha desarrollado gracias a la mejora en los algoritmos para el entrenamiento de redes profundas, al aumento de capacidad de cálculo de los ordenadores y a la ampliación de los conjuntos de datos necesarios para el entrenamiento de estos sistemas.

# ****Metodología****

Durante el desarrollo de este proyecto se ha realizado una simulación de metodología ágil a través de Scrum a pesar que no nos encontramos en un proyecto de desarrollo de software, sino en un proyecto o caso de uso de Data Science.

Aun así, con el fin de hacer una planificación para el desarrollo del proyecto de una manera eficaz y teniendo en cuenta la materia en el master, se ha decidido hacer una simulación de metodología ágil para este proyecto para una mayor familiarización.

El desarrollo ágil de software describe un conjunto de valores y de principios que son aplicables a la creación del software donde los requisitos y la solución final evoluciona a través de un esfuerzo colaborativo de equipos multidisciplinares que se gestionan de forma autónoma. El agilismo hace referencia a procesos de planificación adaptativa, desarrollo evolutivo, entrega pronta y uso de procesos de mejora continua.

Fue desarrollada por Ken Schwaber6, Jeff Sutherland7 y Mike Beedle a finales de los años noventa. Scrum se define como un framework de trabajo para la gestión de proyectos de desarrollo de software que ha demostrado su valía en numerosos proyectos durante los últimos diez años. Es un conjunto de reglas muy sencillas que facilitan trabajar eficientemente de forma colaborativa, así como optimizar la entrega de valor al negocio en el menor tiempo posible.

Es un proceso iterativo e incremental. Permite gestionar el desarrollo de software mediante un sistema flexible donde un equipo de desarrollo trabaja como una unidad que intenta alcanzar una meta común. El equipo trabaja directamente con el cliente de forma autoorganizada y tanto el cliente como el equipo están completamente centrados en hacer el mejor producto posible.

Como se ha comentado, este proyecto no consiste en un proyecto de software con un cliente final, por lo que no se dispone de Product Owner, que es la voz del cliente. Representa a las partes interesadas en el producto: los jefes del negocio, los usuarios y a cualquier otra persona que esté en el lado del cliente.

Tampoco se dispone de una figura única de Scrum Master, que es la persona que, sin formar parte del equipo ni tener responsabilidad alguna en el delivery, está encargada de hacer que todas las partes se muevan y que el sistema produzca lo máximo posible respetando siempre las normas del juego. Es el árbitro del proceso, conocedor como ninguno de los rituales de Scrum, es el máximo responsable de diseminar la ideología Agile entre todas las partes.

Los pasos que ha incluido o se ha basado en las metodologías ágiles para el desarrollo de este proyecto son los siguientes:

* Documentación sistemática de los pasos y ejercicios sucesivos del aprendizaje,
* Uso de Historias de Usuario y definición de sus criterios de aceptación
* Separación en Sprint compuesto por un conjunto de historias de usuario que define las diferentes partes del proyecto, sin exceder de tamaño lo suficiente que pueda ser medible en el tiempo de duración
* Consejos generales sobre mejores prácticas de programación, normas de trabajo en equipo.
* Uso de las herramientas de documentación.

# ****Entorno**** de trabajo

Se tiene en cuenta diferentes entornos donde disponen de las herramientas cumplir con los objetivos del proyecto.

A continuación, se especifica los requerimientos técnicos de software que se han utilizado para ello:

## Equipo de trabajo máquina local

* Procesador: Intel I8
* Memoria instalada (RAM): 8,00 GB (7,85 GB utilizable)
* Tipo de sistema: Sistema operativo de 64 bits, procesador x64
* Edición Windows: Windows 10 Pro
  1. Equipo de trabajo en máquina virtual

Para la instalación y ejecución de la máquina virtual VMware con sistema operativo Ubuntu, se tuvo que activar la tecnología de virtualización en la BIOS ya que se encontraba desactivada por defecto.  
VMware

Instalación Máquina Virtual **VMware Workstation 15** Player for Windows 64-bit Operating Systems a través de la página oficial:

<https://my.vmware.com/en/web/vmware/free#desktop_end_user_computing/vmware_workstation_player/15_0>

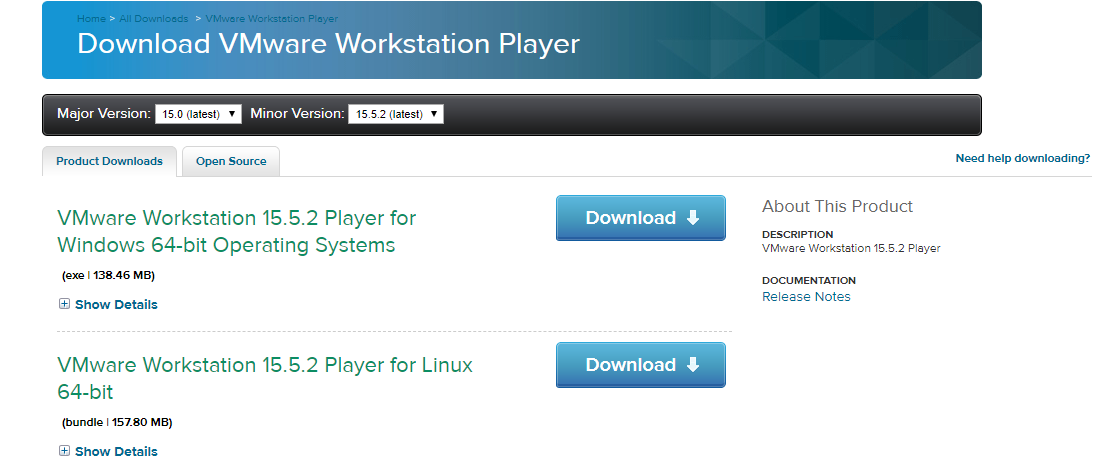


Figura 2: Página oficial de descargas VMware

* 1. **Sistema operativo Ubuntu**

Descarga de sistema operativo Ubuntu 18.04.1 a través de la página oficial:  
<https://www.ubuntu.com/download/desktop>

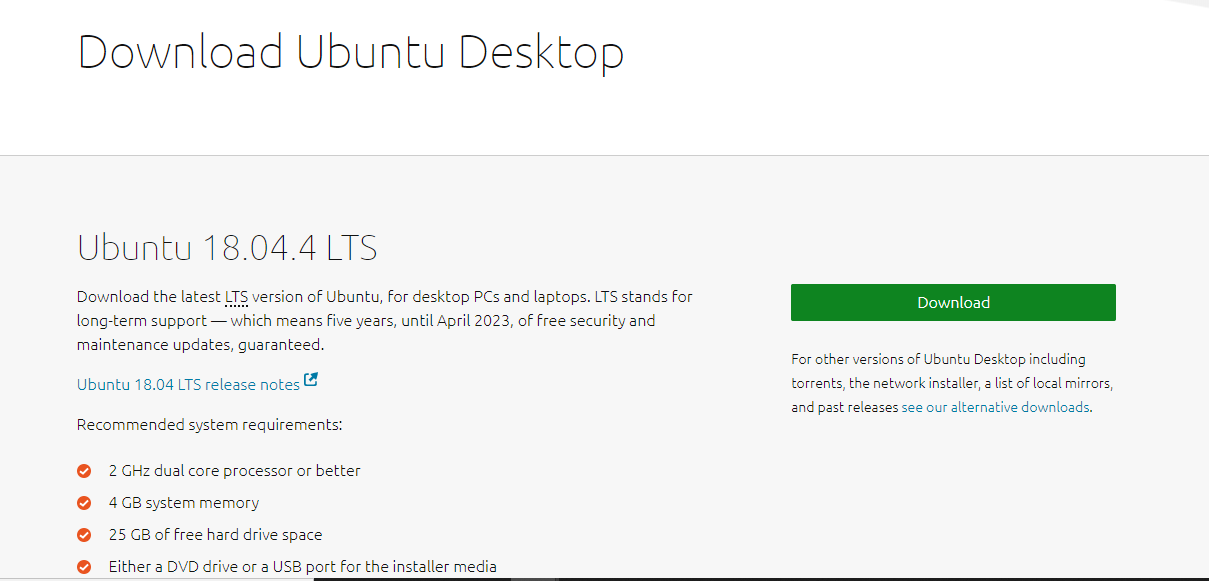


Figura 3: Página oficial de descargas Ubuntu

* 1. **Entorno de Desarrollo**
* No se utilizará un entorno de desarrollo específico, más allá de un editor de textos avanzado, el Editor **Sublime Text** y notebooks de **Jupyter**.
* El lenguaje de desarrollo es **Python** y **R**.
* Se emplea la utilidad **Pip3,** que es un administrador de paquetes Python3. Este es un sistema de gestión de paquetes sencillo utilizado para la instalación y administración de paquetes que pueden ser encontrados en el Python Package Index (PyPI) TBD.

### ****Componentes****

Además de los elementos antes mencionados, las librerías de Python necesarias son las siguientes:

* **Numpy:** Paquete de computación científica. Contiene, entre otras cosas, un potente objeto de tipo array N-dimensional, utilidades de algebra lineal, y capacidades sobre números aleatorios.
* **Pandas**: Librería de código abierto que provee alto rendimiento y facilidad de uso en estructura de datos y análisis de datos para Python.
* **Scikit-learn**: Conjunto de herramientas que permiten de forma simple y eficiente llevar a cabo de análisis de datos y minería de datos. Destaca por su reusabilidad y es de código abierto.
* **Matplotlib**: Librería de dibujo trazado que produce figuras en gran variedad de formatos y entornos.
* **Otras librerías necesarias:**Os, Re, Zipfile, Csv, y Math. Contienen funciones matemáticas simples y utilidades de manejo de ficheros y directorios.

# ****Datos****

Los datos están disponibles a través de un enlace que ha facilitado la Escuela Técnica Superior de Ingenieros Informáticos.

## Descripción de los datos

Consiste en un archivo .RAR y desde el punto de vista de este proyecto, son relevantes los siguientes directorios:

* Genes -> TAIR10\_genome\_release: Este directorio contiene información extraído desde el TAIR (Fuente de información de Arabidopsis).  Existen varias versiones disponibles, en el que se considera el más actualizado la v10.
* TAIR10\_gene\_lists -> TAIR10\_all\_gene\_models: Lista de todos los genes conocidos de la planta codificados a través de su posición en el cromosoma y el locus (e.g. AT1G01010).
* TAIR10\_gene\_lists -> TAIR10\_gene\_type: Tipo de gen conocido, que es aproximadamente la función.
* TAIR10\_chromosome\_files -> NCBI\_Chr1.tbl: Archivo .txt que contiene información más extendida de cada gen, incluyendo las funciones en la célula, ciclo de la planta en el que el gen se expresa, y similitud con los genes de otras especies (solo está disponible el número de apariciones, e.g. 2502 in other plants ) Note: Hay disponibles 5 archivos, uno por cada cromosoma.
* Microarrays -> Datasets: Este directorio contiene los resultados numéricos en el conjunto de experimentos, cada uno de ellos organizados en sub-directorios y comprimido en un archivo .ZIP. Cuando se abre, cada experimento contiene 2 importantes archivos:
* ExperimentName\_README.txt: Información general sobre el experimento, incluyendo el nombre y la descripción. La estructura del archivo ReadMe es constante en todos los experimentos.
* ExperimentName\_Tair.txt: Contiene información sobre el nivel de expresión del conjunto de genes. Lo relevante para este proyecto son lo siguientes campos:
* Locus: ID del gen.
* IS\_EXPRESSED: Valor booleano definiendo si el gen se expresa en el experimento.
* AVG\_FOLD\_CHANGE: Cuanto se expresa el gen, o no se expresa. Consiste en un valor flotante positivo o negativo.

Con el fin de cumplir el objetivo que se busca en el proyecto, se va hacer uso de única y exclusivamente el archivo con los archivos de los diferentes experimentos:

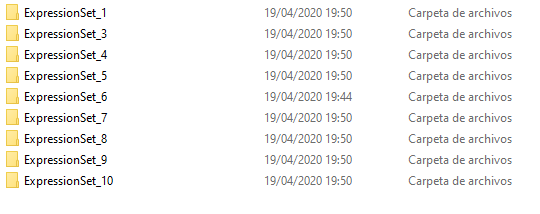


Figura 4: Expresión del gen

Fuente: Elaboración propia

En cada fichero Expression\_Set\_X se encuentra los diferentes nombres de los experimentos que se han realizado en archivos.zip, al que a cada uno le corresponde un tipo de experimento y contiene lo siguiente:

* Experimento\_original.txt: Se encuentran valores estadísticos de los ensayos
* Experimento\_README.txt: Se explica los diferentes parámetros que compone el fichero de expresión del gen
* Experimento\_tair.txt: Archivo objetivo en donde se quiere extraer la información de la expresión del gen:
  + ARRAY\_ELEMENT: Name of the element on an array as given by the manufacturer.
  + **LOCUS : Mapped element that corresponds to a transcribed region in Arabidopsis genome, or a genetic locus that segregates as a single genetic locus or quantitative trait.**
  + SUID: SMD unique identifier for a SMD sequence.
  + FLAG: Type of flag associated with an array element; set by either image processing software or manually; 0 - good, other than 0 - bad.
  + IS\_EXPRESSED : Qualitative measure of gene expression as a function of signal intensity; 'yes' - if intensity values >= 350 in both channels after background subtraction; 'no' - if intensity values < 350 in any channel after background subtraction; 'absent' - if the data were flagged as unreliable either by software or manually.
  + **FOLD\_CHANGE**: **How many times the expression signal for a given transcript increased or decreased compared to the control.**
  + STD\_ERR: The standard deviation of the sampling distribution of the mean.
  + AVG\_FOLD\_CHANGE: Averaged fold change for all replicate hybridizations.
  + AVG\_STD\_ERR: The standard deviation of the sampling distribution of the mean for the replicate set.

Como conclusión, la información que se quiere extraer para poder aplicar los métodos de Data Scientist es el atributo **FOLD\_CHANGE** y el nombre del gen **LOCUS** de los siguientes experimentos:

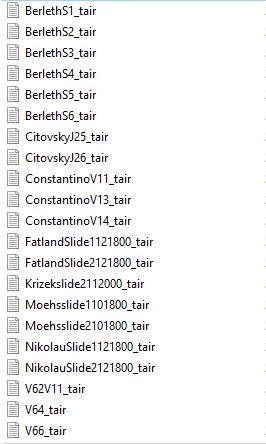


Figura 5: Ficheros de expresión

Fuente: Elaboración propia

El hecho que se repitan los nombres de los experimentos, quiere decir que se corresponde a una repetición del mismo ensayo a la planta Arabidopsis.

## Organización de los datos para el programa

Para poder realizar un análisis exploratorio de datos e implementar los algoritmos en Machine Learning junto con la información de los genes con el fin de averiguar si existe relación entre ellos, en primer lugar, se deben de organizar de tal manera que incluya el nombre del gen, y su nivel de expresión (Fold\_change) en el ensayo.

Para ello se ha optado por crear dataframe donde cada fila es un ensayo y cada columna un gen, cada celda contiene el fold\_change de un gen en un ensayo concreto:

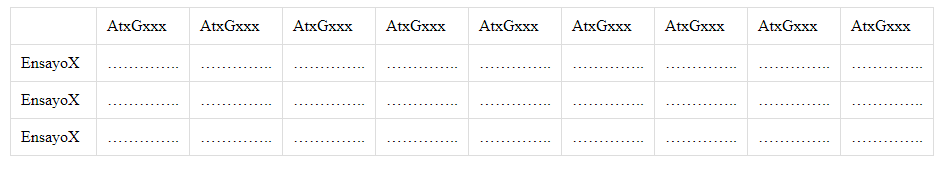


Figura 6: Organización de los datos

Fuente: Elaboración propia

Con esta organización de los datos lo que se pretende es hacer “loop” de los algoritmos de manera que divida los datos en grupos para entrenar y otro para hacer los test.

En el primer bucle (externo) buscamos dividir los datos en dos grupos, un grupo para entrenar y otro para hacer el test.

En el segundo bucle (interno), como necesitamos tener nuestro grupo de datos ‘validadores’ sobre los que vamos a comprobar si la predicción es buena.

De esta forma, según avanzamos en el bucle externo, iremos cogiendo filas, y en el bucle interno vamos cogiendo columnas, así podemos predecir cada uno de los fold\_change para cada gen en cada uno de los ensayos.

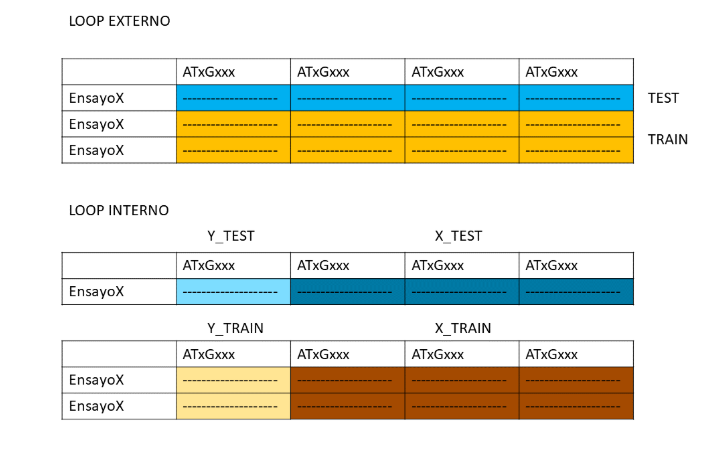


Figura 7: Programa Machine Learning

Fuente: Elaboración propia

# Desarrollo Data Science

Una vez que se ha definido las diferentes partes en el que se va a dividir el proyecto Data Science en planta Arabidopsis, se va a estudiar y desarrollar los datos para almacenamiento y organización en un dataset.

De esta manera facilita un análisis exploratorio de los datos, en el que se pretende conocer la naturaleza de los mismos, identificar valores nulos o valores fuera de rangos que pueden influir de manera negativa en el entrenamiento de los modelos de Machine Learning.

A continuación, se realizará métodos con el que se permita conocer que modelos de algoritmos son interesantes para conseguir el objetivo del proyecto, además de sus parámetros para conseguir un mayor ajuste en los resultados.

Por último, con los resultados permitirá realizar una evaluación y definir conclusiones finales a través de coeficientes y gráficas que identifique el nivel de relación entre los genes.

## FASE I: Extracción, tratamiento y carga de los datos

ETL es el proceso en el que es responsable de la extracción de datos y de su limpieza, conformación y localización en el almacén de datos.

Las herramientas o métodos de ETL recopilan datos de varias fuentes (tablas de bases de datos, archivos planos, ERP, Internet, etc.) y les aplican transformaciones complejas.

Para este caso solo va hacer falta una única fuente de datos.

**Extracción**

El objetivo de esta fase es la extracción de los datos de cada una de las fuentes identificadas, realizando una copia exacta (sin procesamiento) de los datos de origen.

En esta fase se realiza el acceso a los datos de origen de una forma eficiente, periódica y automatizada, definiendo la conexión/integración con las fuentes de datos internas o externas que se hayan identificado.

Para el proyecto presente la extracción de los datos se hace a través de un enlace de AWS facilitado por la Universidad Politécnica, por lo que no requiere de extracción de diferentes fuentes operacionales y no se considera necesario identificar y diseñar las ventanas óptimas de carga sobre los orígenes.

**Desarrollo de Extracción de los Datos**

Como se ha comentado anteriormente, no se va a proceder el uso de herramientas que permite diseñar ETLs mediante transformaciones y trabajos que pueden ser ejecutadas por las herramientas de Spoon, Pan y Kitchen, o en su defecto Pentahoo que es un set de herramientas que reúne las ya citadas.

A través del lenguaje de programación Python se define una función en el que extrae los ficheros de interés para ser almacenados en una ubicación común.

De esta manera se reduce el espacio de aquellos datos que no interesan, y facilita a la hora de definir la función desde una misma ubicación.

Para la identificación de los ficheros de interés se definen expresiones regulares, también conocidas como 'regex' o 'regexp', que son patrones de búsqueda definidos con una sintaxis formal. Siempre que se siga sus reglas, se puede realizar búsquedas simples y avanzadas que, utilizadas en conjunto con otras funcionalidades, las vuelven una de las opciones más útiles e importantes de cualquier lenguaje.



Figura 8: Código Python expresión regular

Fuente: Notebook Fase I ETL Elaboración propia

Con este patrón de expresión regular indica aquellas palabras que contengan cualquier letra en mayúscula o minúscula y números, con extensión “\_tair.txt”.

Tras su identificación simplemente queda cambiar la ubicación de los ficheros al fichero común.

**Transformación y carga**

La transformación de los datos en el proceso de ETL es el segundo de los procesos y consiste principalmente en limpiar y conformar la información extraída de la fuente. Este paso es el más laborioso y en el que ETL agrega más valor.

En esta fase se realizan los procesos de limpieza de datos para aportar calidad a los mismos, eliminando para ello, los datos erróneos y complementando con otras fuentes, se enriquecen los datos para que estos sean fiables.

En esta fase, también se realiza el conformado y unificación de fuentes y datos, lo que aporta unicidad a la información, veracidad y genera maestros.

La carga es el paso final del proceso y su objetivo es realizar de forma eficiente, la carga de datos en el almacén de datos y, más concretamente, en cada uno de los modelos multidimensionales definidos.

**Desarrollo de transformación y carga**

Para esta fase se ha definido una función a través de código Python en el que se realice de manera concatenada la transformación de los datos y la carga.

Para ello se vuelve hacer uso de las expresiones regulares para identificar los archivos con extensión \_tair.txt.

Los archivos .txt están nombrados según el nombre del ensayo que se ha realizado a la planta Arabidopsis como se puede ver en la [figura 5](#_Descripción_de_los).

Los nombres de los ensayos son:

* Berleth
* Citovsky
* Constantino
* FatlandSlide
* Krizekslide
* Moehsslide
* NikolauSlide
* V62V11
* V64

Tras localizar los ficheros, se extrae el nombre del ensayo para añadirlo en el dataframe donde se van a cargar todos los datos que se van a utilizar para las siguientes fases del proyecto.

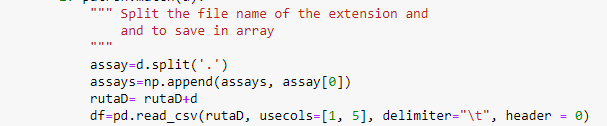


Figura 9: Código Python separación de nombre fichero

Fuente: Notebook Fase I ETL Elaboración propia

Por otro lado, en la misma iteración de extracción del nombre del ensayo, se procesa el mismo fichero para extraer todos los genes que se han aplicado en el ensayo, de manera que se registra en el dataframe el valor de expresión (Fold\_Change).

Aprovechando el proceso de transformación y carga, se omite cualquier valor nulo para evitar futuros errores en la implementación de los algoritmos.

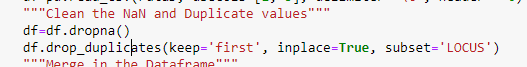


Figura 10: Código Python limpieza de valores nulos y duplicados

Fuente: Elaboración propia

Se obtiene un dataframe de 7359 filas y 21 columnas con todos los genes que se les ha realizado los ensayos y sus respectivos valores de expresión que han manifestado durante los proyectos de investigación.

El hecho de tenerlo registrado en un dataframe, lo hace más manejable y versátil a la hora de poder ser manipulado, analizado o almacenado en otros formatos para tratamientos de los datos en otros lenguajes de programación como R.

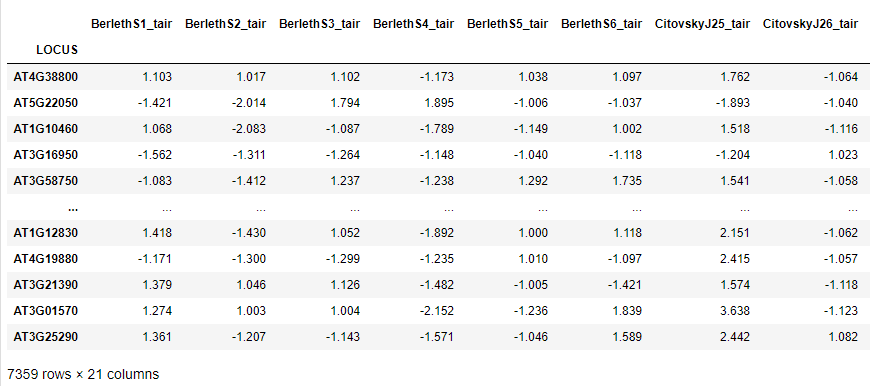


Figura 11: Output función ETL Python

Fuente: Elaboración propia

## FASE II: Análisis exploratorio de los datos

El análisis exploratorio de los datos o también conocido como EDA, consiste en el análisis estadístico para entender las variables del dominio de negocio o científicas, sus dependencias y relaciones.

Aporta una explicación científica a través de gráficas y estadísticos a la incertidumbre de los datos y ayuda a entender mejor los mismos dotándonos de estructura.

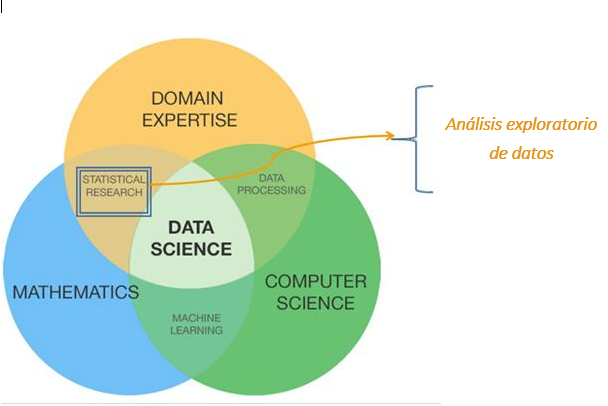


Figura 12: EDA

Fuente: Documentación Máster en Business Analytics y Big Data

Uno de los problemas para el análisis exploratorio de los datos que muestran las expresiones de los genes organizado en el Dataframe de la figura 11, es que todos pertenecen al mismo tipo de variable.

Solo les diferencia el tipo de ensayo, que solo influye en el valor de expresión del gen.

Por ello se ha querido enfocar únicamente en conocer cuál es la distribución de los valores de los genes, y averiguar si hay valores Outliers[[1]](#footnote-1).

Para esta fase se ha desarrollado a través del lenguaje de programación R, que es un lenguaje diseñado para el tratamiento de datos y la aplicación de procesos y modelado estadístico. Este tiene muchas facilidades incorporadas en su sintaxis, multitud de librerías específicas para distintas tareas estadísticas, que a su vez incorporan una colección de datos copiosa. Además, se enmarca en lo que se conoce como código abierto: esto significa que todas las librerías, herramientas y código base se pueden usar gratuitamente.

El desarrollo de R se hará a través de RStudio, que es un IDE (integrated development environment), programa capaz de cargar el lenguaje R, comprenderlo y ejecutar sus instrucciones.

**Desarrollo con RStudio**

Para poder extraer información del dataframe de la fase del ETL desarrollado en lenguaje Python, se incluyó en la función de transformación y carga de los datos, otra función que se guardará los valores de expresión del dataframe en un archivo **AGDA.csv**.

De esta manera se puede extraer la información de los resultados a través de otros lenguajes de programación, como en este caso, a través del lenguaje de programación R.

En el análisis exploratorio de los datos se puede observar lo siguiente:

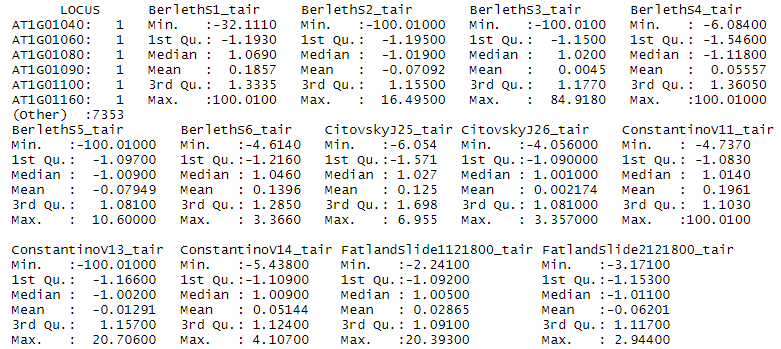


Figura 13: Descripción valores estadísticos

Fuente: Elaboración propia

A través de este análisis, se puede apreciar la media, mediana, cuartiles, valor mínimo y valor máximo, para variables cuantitativas y la frecuencia absoluta para variables cualitativas.

Se puede observar que los valores responden a un rango negativo como positivo con intervalos entre 10 y -10 por lo general.

Se consideran variables continuas, porque puede adoptar cualquier valor en el marco de un intervalo que ya está predeterminado, por lo que es válido tener en cuenta el valor “Mean ” para conocer el valor medía de cada ensayo.

Aun así, se puede observar que hay ensayos que tienen valores máximos que se diferencian del comportamiento general de los valores estadísticos.



Figura 14: Summary BerlethS1

Fuente: Elaboración propia

Esto conlleva identificar la existencia de valores atípicos a través de gráficas boxplot.

Las gráficas boxplot es un tipo de gráfico que muestra un resumen de una gran cantidad de datos en cinco medidas descriptivas, además de intuir su morfología y simetría.

Este tipo de gráficos nos permite identificar valores atípicos y comparar distribuciones. Además de conocer de una forma cómoda y rápida como el 50% de los valores centrales se distribuyen.

Se puede detectar rápidamente los siguientes valores:

* Primer cuartil: el 25% de los valores son menores o igual a este valor (punto 2 en el gráfico anterior).
* Mediana o Segundo Cuartil: Divide en dos partes iguales la distribución. De forma que el 50% de los valores son menores o igual a este valor (punto 3 en el gráfico siguiente).
* Tercer cuartil: el 75% de los valores son menores o igual a este valor (punto 4 en el gráfico siguiente).
* Rango Intercuartílico (RIC): Diferencia entre el valor del tercer cuartil y el primer cuartil.

Las dimensiones de la caja están determinadas por la distancia del rango intercuartílico, que es la diferencia entre el primer y tercer cuartil, es decir, que en nuestro gráfico vemos que para la filial central.

El segmento que divide la caja en dos partes es la mediana, que facilitará la comprensión de si la distribución es simétrica o asimétrica.

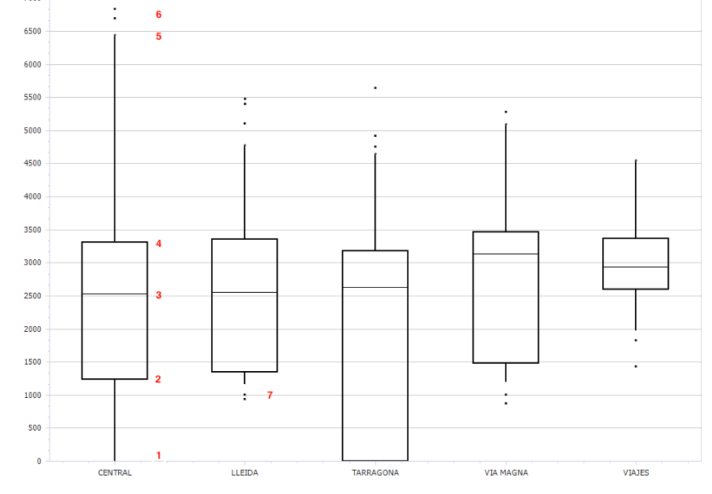


Figura 15: Ejemplo gráfica Boxplot

Fuente: Elaboración propia

Para el presente proyecto se ha extraído todos los ensayos que presenten valores máximos o mínimos con rangos muy alejados, y se ha representado en la siguiente gráfica:

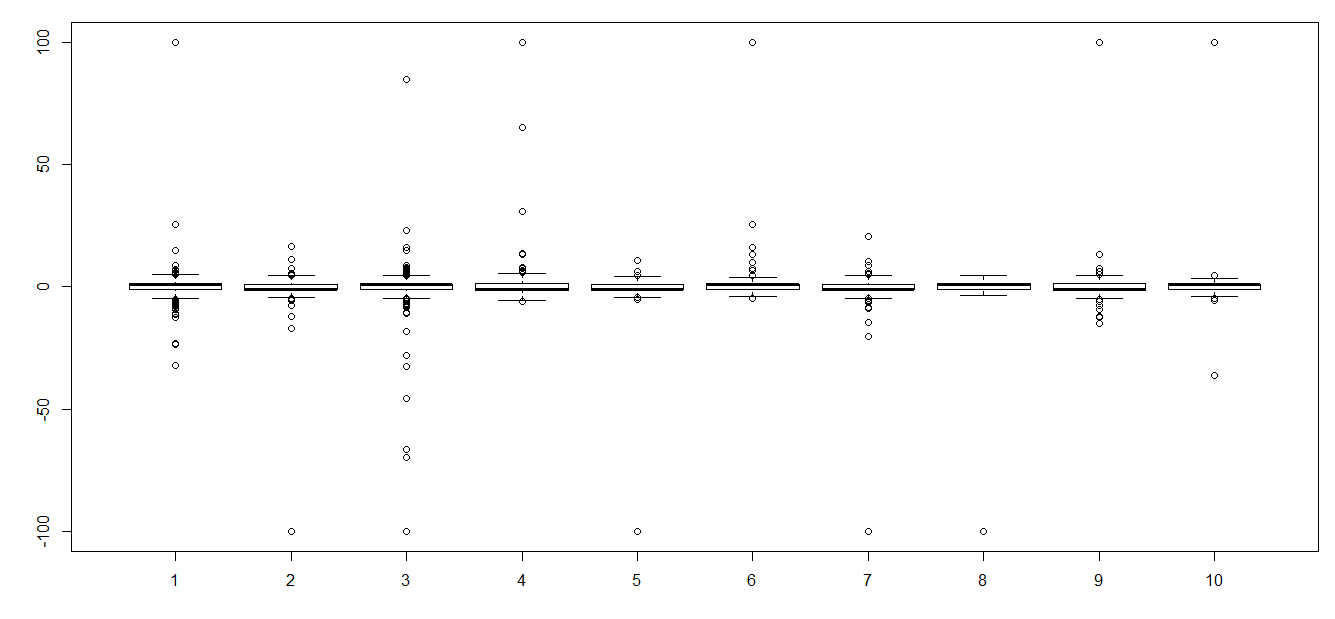


Figura 16: Gráfica Boxplot Ensayos

Fuente: Elaboración propia

No se puede apreciar en esta gráfica el comportamiento de las medias de los todos los ensayos, por lo que se construye un Barplot (gráfico de barras), para complementar la información EDA.

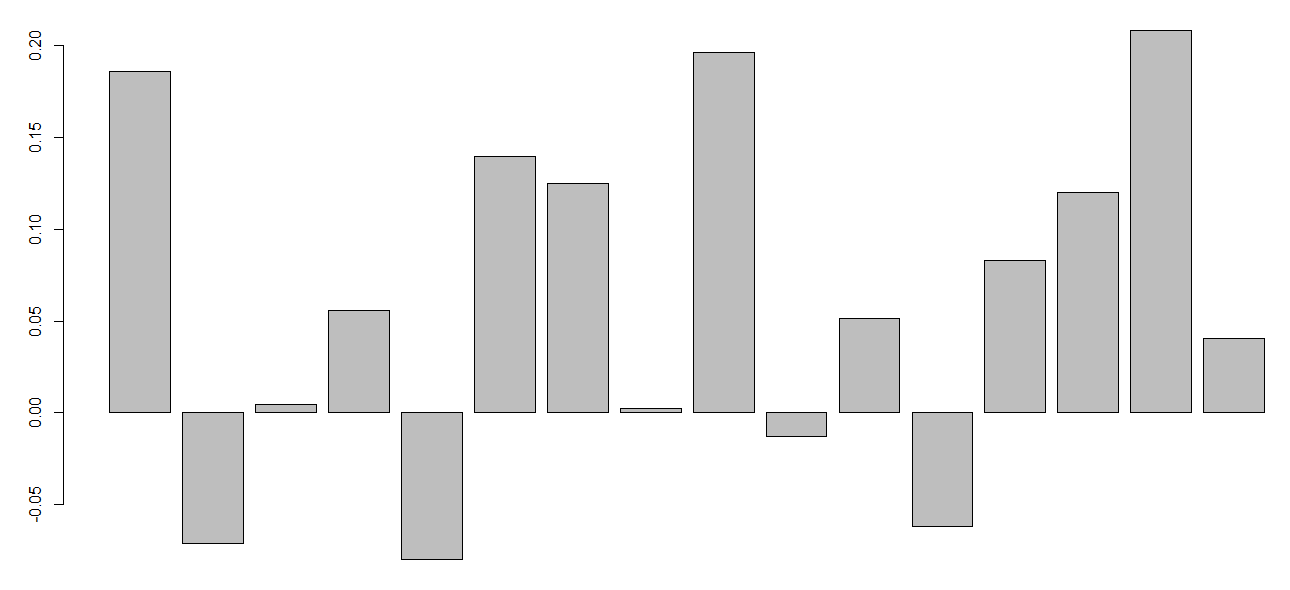


Figura 17: Barplot Mean

Fuente: Elaboración propia

Como se sospechaba, el conjunto de valores se encuentra cercanos a cero, siendo positivos o negativos. Por otro lado, se observa que los valores atípicos que pueden superar el valor Fold\_Change más de 50.

Tras consultar a los responsables de las investigaciones del proyecto la naturaleza de estos resultados, se ha decidido no realizar a priori ninguna limpieza de estos datos con valores por encima de 10 o debajo de -10, ya que pertenece a los posibles resultados de expresión del gen.

## Fase III: Ingeniería de características y selección de modelos

En un proyecto Data Science, por lo general no se conocen a priori en un conjunto de datos las características que son más adecuadas para identificar el modelo empleado. Además, durante el proceso de entrenamiento, en la mayoría de los modelos es necesario indicar parámetros, los cuales determinarán un mejor o peor resultado.

Para el presente proyecto se va a realizar **Feature Selection**, o también conocido como Selección de Características.

Como se ha comentado anteriormente, el presente proyecto consta de un conjunto de datos de la misma naturaleza, mismo tipo de características y mismo tipo de variables.

Por ello se considera interesante aplicar el método Feature Selection, ya que es el proceso de seleccionar las más importante y/o relevantes características de un conjunto de datos, con el objetivo de mejorar el rendimiento de predicción de los predictores, proporcionar predictores más rápidos y más rentables y proporcionar una mejor comprensión del proceso subyacente que generó los datos.

Para este proyecto se van aplicar los 3 siguientes métodos:

* Filter based
* Wrapper-based
* Embedded Lasso

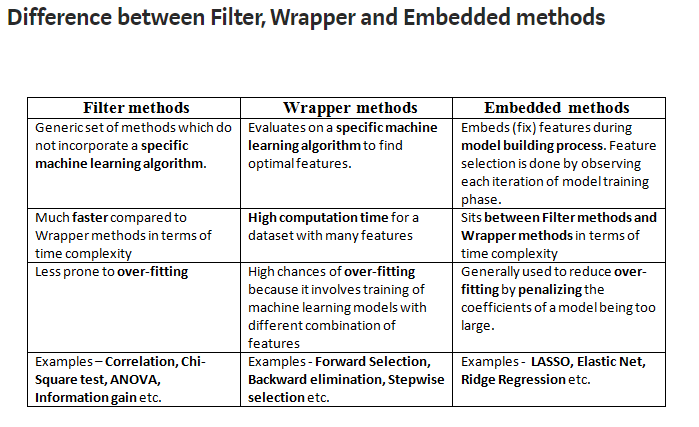


Figura 18: Diferencias entre los métodos Feature Selection

Fuente: <https://towardsdatascience.com/feature-selection-using-wrapper-methods-in-python-f0d352b346f>

### Filter Based

Como el nombre sugiere, en este método se filtra solo el subconjunto de características relevantes.

El filtro se hace a través de una matriz de correlación de Pearson[[2]](#footnote-2).

Los valores de correlación de Pearson entre variables pueden haber de tres tipos:

* Si r < 0 Hay correlación negativa: las dos variables se correlacionan en sentido inverso. Cuánto más próximo a -1 esté el coeficiente de correlación más patente será esta covariación extrema.
* Si r= -1 hablaremos de correlación negativa perfecta lo que supone una determinación absoluta entre las dos variables (en sentido inverso): Existe una relación funcional perfecta entre ambas (una relación lineal de pendiente negativa).
* Si r > 0 Hay correlación positiva: las dos variables se correlacionan en sentido directo. A valores altos de una le corresponden valores altos de la otra e igualmente con los valores bajos.

Cuánto más próximo a +1 esté el coeficiente de correlación más patente será esta covariación.

* Si r = 1 hablaremos de correlación positiva perfecta lo que supone una determinación absoluta entre las dos variables (en sentido directo):Existe una relación lineal perfecta ( con pendiente positiva).
* Si r = 0 se dice que las variables están “incorrelacionadas”: no puede establecerse ningún sentido de covariación.

**Code Pearson Correlation**

Para desarrollar esta fase del proyecto se hace a través de lenguaje de programación Python, con la ayuda de librerías como Pandas [[3]](#footnote-3)y Matplotlib[[4]](#footnote-4).

A través de ella se lee los datos ya limpiados y almacenados en el fichero AGDA.csv, y posteriormente se construye una matriz de correlación para conocer que variables presentan altos valores.

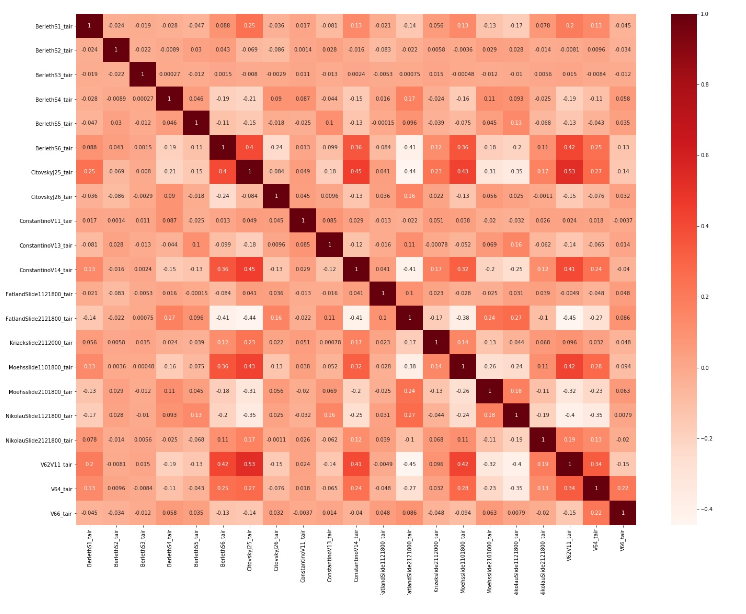


Figura 19: Matriz de correlación de Pearson

Fuente: Elaboración propia

A través de la figura 19 se puede apreciar que valores presentan mayor correlación.

Se puede apreciar que en término generales no presentan índices altos de relación entre variables, ya que apenas superan los resultados más de 0,6.

Se debe de recordar que el objetivo de este proyecto es conocer la relación de las variables, de manera que es factible un resultado negativo como positivo, es decir, el hecho de que no presenten relación entre variables, es información útil para el objetivo del proyecto, al igual que si presentan resultados positivos de relación.

A continuación, se desarrolla una función en el que se extrae aquellos valores que presenten valores de correlación mayor de 0,4 para identificar que ensayos pueden presentar correlación directa.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | BerlethS6 | CitovskyJ25 | Constantino  V14 | Moehsslide  1101800 | V62V11 |
| BerlethS6 | 1 | **0.404618** | <0,4 | <0,4 | **0.415634** |
| CitovskyJ25 | **0.404618** | 1 | **0.450922** | **0.430111** | **0**.**525892** |
| Constantino  V14 | <0,4 | **0.450922** | 1 | <0,4 | **0.405755** |
| Moehsslide  1101800 | <0,4 | **0.430111** | <0,4 | 1 | **0.424311** |
| **V62V11** | **0.415634** | **0.525892** | **0.405755** | **0.424311** | 1 |

Figura 20: Cuadro resumen correlación >0,4

Fuente: Elaboración propia

Como se puede comprobar, en el cuadro resumen en el que se ha extraído a través de la función desarrollado con Python los resultados que presentaban una correlación directa mayor de 0,4, los ensayos que más correlación directa presenta en las iteraciones entre ellos son **BerlethS6, CitovskyJ25, ConstantinoV14, FatlandSlikde2121800, Moehssile2101800, V62V11**.

En cambio, también interesa conocer cuales presentan mayor correlación inversamente directa por debajo de -0,4.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | CitovskyJ25 | Constantino  V14 | FatlandSlide  2121800 | NikolauSlide  1121800 | V62V11 | BerlethS6 |
| CitovskyJ25 | 1 | <0,4 | **-0.435509** | <0,4 | <0,4 | <0,4 |
| ConstantinoV14 | <-0,4 | 1 | **-0.412606** | <0,4 | <0,4 | <0,4 |
| FatlandSlide  2121800 | **-0.435509** | **-0.412606** | 1 | <0,4 | **-0.445044** | **-0.406343** |
| NikolauSlide  1121800 | <-0,4 | <0,4 | <0,4 | 1 | **-0.404036** | <-0,4 |
| V62V11 | <-0,4 | <0,4 | **-0.445044** | **-0.404036** | 1 | <0,4 |
| BerlethS6 | <-0,4 | <0,4 | **-0.406343** | <0,4 | <0,4 | 1 |

Figura 21: Cuadro resumen correlación < -0,4 Fuente: Elaboración propia

En la figura 21 presenta las variables **BerlethS6, CitovskyJ25, ConstantinoV14, FatlandSlikde2121800, NikolauSlide1121800, V62V11**, en su iteración con mayor correlación negativo, es decir, se correlacionan en sentido contrario.

Como conclusión, las variables indican que no destacan por su gran interrelación de manera positiva o negativa, ya que apenas superan el 0,5 entre ellas.  
De todos modos es importante tener en cuenta cuales presentan mejores correlaciones de cara a la elección de modelos y su implantación.

### Wrapper-Based

Proceso basado en un algoritmo de Machine Learning que entrena un dataset dado, para posteriormente aplicar criterios de evaluación y tomar decisiones en eliminar o añadir features.

Este método es más costoso computacionalmente que el método de filtrado con el que se ha ejecutado a través de la correlación de Pearson, pero al no tratarse de un dataset excesivamente grande, se puede aplicar sin problemas.

Como algoritmo se utilizará RFE (Recursive Feature Elimination)[[5]](#footnote-5), que funciona utilizando la métrica de precisión para clasificar la característica según su importancia. El método RFE toma el modelo a utilizar y la cantidad de características requeridas como entrada. Luego da la clasificación de todas las variables, siendo 1 la más importante.

**Code RFE**

Para la aplicación del RFE se ha hecho importando el algoritmo a través de Scikit Learn, en el que se ha divido en dos funciones; la primera función para encontrar el número óptimo con el fin de conocer que rango de clasificación a la hora de escoger que variables ayudan a obtener mayor precisión. Teniendo en cuenta que a priori no se conoce que variables es la variable más dependiente, se ha creado una función donde a través de una iteración se obtiene todos los valores individuales de cada variable, obteniendo los siguientes resultados:

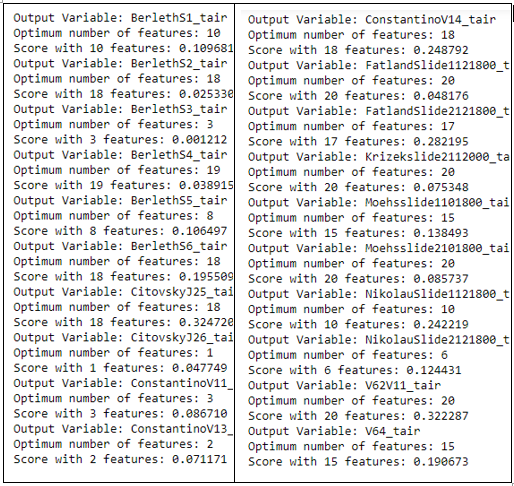


Figura 22: Valor óptimo features RFE

Fuente: Elaboración propia

Como se observa, **CitovskyJ25** presenta mayor score que los demás, por lo que se toma la decisión de escoger **18** como número óptimo de variables como parámetro en el algoritmo RFE.

La segunda función donde se implementa el algoritmo RFE con el parámetro de la primera función, que en este caso es 18 con el que se muestran las variables a tener en cuenta:

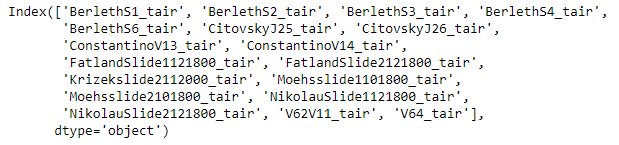


Figura 23: Output algoritmo RFE

Fuente: Elaboración propia

### Embedded Method: Lasson

Los métodos embebidos son iterativos en un sentido que se encarga de cada iteración del proceso de capacitación del modelo y extrae cuidadosamente las características que más contribuyen a la capacitación para una iteración en particular. Son los métodos integrados más utilizados que penalizan una característica dado un umbral de coeficiente.

Se realizará a través del método Lasso, que es la función que indica el error cometido por el modelo en función de los parámetros y es la que se minimiza utilizando el algoritmo del gradiente descendente para la obtención de los parámetros del modelo.

Lasso afrontan el problema de la multicolinealidad. Esta se presenta cuando existen variables predictoras que son casi dependientes entre sí, lo que produce un efecto de inestabilidad del resultado numérico del modelo, si no se trata adecuadamente.

La regularización consiste en añadir una penalización a los coeficientes de regresión que sean demasiado grandes para reducir el sobreajuste. Por lo tanto, si la característica es irrelevante, le aplicará un coeficiente de 0.

**Code Lasso**

La implantación de la regresión LASSO en scikit-learn es similar a la regresión lineal y comparte una gran cantidad de opciones. El constructor para este tipo de modelos es Lasso, que se encuentra dentro de sklearn.linear\_model. Entre las opciones más interesantes, se pueden enumerar:

* alpha: es el constante que multiplica al parámetro del modelo para regularizarlo. El valor por defecto es 1 y, en caso de indicar 0, el modelo es uno lineal simple.
* fit\_intercept: es un valor lógico que indica si el modelo tiene término independiente, por defecto, su valor es cierto.
* normalize: es un valor lógico que indica si las características se normalizan antes de la regresión. En caso de que se seleccione esta opción, los resultados suelen ser más robustos. Por defecto, este valor es falso.

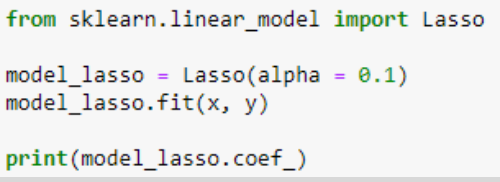


Figura 24: Método Lasso Scikit-Learn

Fuente: Elaboración propia

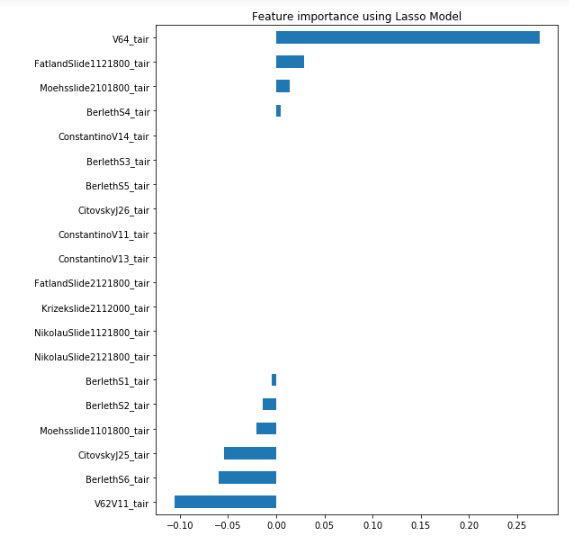


Figura 25: Gráfica análisis Lasso

Fuente: Elaboración propia

El parámetro de regularización determina la cantidad de características que serán utilizadas en el modelo final. A medida que el valor se acerca a la unidad, la penalización en la regresión se hace más estricta. Por otro lado, a medida que se reduce la penalización, empieza a ser menos importante hasta llegar a una regresión lineal cuando este se hace cero.

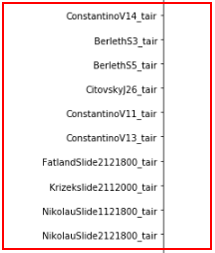


Figura 26: Variables con coeficiente Lasso 0

Fuente: Elaboración propia

Las variables que según el método Lasso los considera como irrelevantes son: **Constantinov14, BerlethS3, BerlethS5, Citvsky26, ConstantinoV11, Constantinov13, FatlandSlide2121800, Krizeksilde2112000, NikolauSlide1121800, NikolauSlide2121800.**

### Conclusión análisis de selección de características

A lo largo de *la FASE III: Ingeniería de características y selección de modelos*, se ha podido conocer la naturaleza y comportamiento de las variables, de manera que se ha obtenido más información cuales representan mayor correlación entre ellas, o presentan mayor valor a la hora de poder aportar información a través de la implementación de modelos Machine Learning que se realizará en la siguiente fase.

A continuación, se muestra un cuadro resumen donde se exponen todas las variables que han estado presente en los resultados de los métodos de Pearson, RFE y Lasso, de manera que se puede identificar que variables muestran mejores resultados.



Figura 27: Cuadro resumen resultado coeficientes selección de características

Fuente: Elaboración propia

**BerlethS6, CitovskyJ25, Moeshslide110180 y V62V11** son las variables que presentan mayor mejores resultados según los métodos aplicados para selección de características.

## FASE IV: Implementación de modelos Machine Learning

En la siguiente fase del proyecto tiene como objetivo averiguar el nivel de relación de los genes de una planta Arabidopsis Thailiana a través de métodos Machine Learning, teniendo en cuenta todas las variables que los anteriores métodos se han calificado como importantes.

De esta manera si todos los genes son independientes, maximizaría el procesamiento de la información del sistema y, por lo tanto, esperaríamos un error de predicción muy alto. Al mismo tiempo, los genes se organizan en redes para evitar redundancias (es decir, varios genes pueden participar en las mismas funciones), por lo que este error puede no ser tan alto.

Por cada ensayo de gen, se prepara el conjunto de datos de la siguiente manera:

* Features o los datos de entradas del modelo es el nivel de expresión de todos los genes, y en todos los experimentos.
* Target o el valor a predecir, es el nivel de expresión del gen objetivo en todos los experimentos.

El modelo debe de ser entrenado usando los valores "target", y posteriormente la predicción del nivel de expresión en el experimento seleccionado.

Por último, el proceso se repite con el fin de conseguir un error promedio.

Finalmente se obtendrá un listado de en el que se podrá conocer a través de coeficientes que variables se consideran dependientes de una manera fiable, consiguiendo el objetivo final del proyecto.

Debido al tipo de variables que presenta, se van implementar modelos supervisados de tipo regresión, por lo que los algoritmos a utilizar son Regresion Forest, Multilayer Perceptron(MLP) y Artificial Neural Network (Keras).

Además, se pretende también aplicar los métodos para evitar overfitting o underfitting aumenta fiabilidad en los resultados de los algoritmos.

## Algoritmos Machine Learning

El aprendizaje automático (Machine Learning) abarca múltiples técnicas que se utilizan para la identificación de patrones en los conjuntos de datos. Para esto se usan algoritmos con los que se construyen modelos. Un modelo es una representación parcial de la realidad, generalmente una simplificación de la misma, que puede ser utilizado para la realización de predicciones. Los modelos pueden ser tanto una fórmula matemática o un conjunto de fórmulas matemáticas (por ejemplo, una ecuación lineal), conjuntos de reglas (árboles de decisión), estructuras conectadas (las redes neuronales) o sistemas de memorización.

Para el presente proyecto se va hacer uso de algoritmo supervisado y perceptrón multicapa, con la finalidad de averiguar el nivel de relación entre ellas asignando una variable etiquetada en cada iteración.

Los modelos supervisados se caracterizan por ser entrenados mediante conjuntos de datos en los que la solución al problema planteado es conocida, es decir, el conjunto de entrenamiento se divide en una o varias variables independientes y una dependiente (la solución del problema). Estos modelos permiten aprender patrones basándose en la experiencia.

Los modelos elegidos para su implementación son:

* **Random Forest**: Técnica con la que se pretende unir varios árboles de decisión para crear un clasificador más preciso.

Un modelo Random Forest se estima utilizando el siguiente algoritmo:

* Creación de un conjunto de muestras de los datos de entrenamiento. Esta selección se realiza aleatoriamente con reemplazo de los datos, es decir, los registros seleccionados pueden volver a serlo
* Creación de un modelo de árbol de decisión para cada uno de los conjuntos de entrenamiento seleccionados previamente
* Se pueden repetir los procesos 1 y 2 varias veces
* Agregación de los resultados mediante votación
* **Mlp (Multi- layer Perceptron):** Según la definición que muestra la página oficial de [Scikit-Learn](https://scikit-learn.org/stable/modules/neural_networks_supervised.html), MLP es un algoritmo supervisado que aprende la función f(): Rm  -> Ro para entrenamiento de conjunto de datos, donde *m* es el número de dimensiones para una entrada y *o* es el número de dimensiones de salida.

Dado un conjunto de features X=x1,x2,… xm y la variable etiquetada *y,* puede aprender una función no lineal para la clasificación o regresión.

Es una red neuronal artificial (RNA) formada por múltiples capas, de tal manera que tiene capacidad para resolver problemas que no son linealmente separables, lo cual es la principal limitación del perceptrón (también llamado perceptrón simple). El perceptrón multicapa puede estar totalmente o localmente conectado. En el primer caso cada salida de una neurona de la capa "i" es entrada de todas las neuronas de la capa "i+1", mientras que en el segundo cada neurona de la capa "i" es entrada de una serie de neuronas (región) de la capa "i+1".

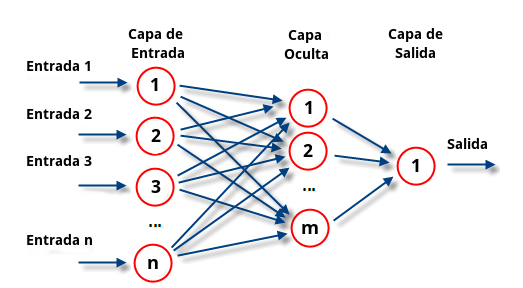


Figura 28: Red Neuronal Multicapa

Fuente: https://es.wikipedia.org/wiki/Perceptr%C3%B3n\_multicapa

La implementación de los algoritmos Machine Learning se va a realizar a través de Python, que se considera un lenguaje de programación de propósito general, a pesar de su popularidad en entornos de Data Science, las funciones de aprendizaje automático no se incluyen en las librerías estándar. Por lo cual es necesario instalar librerías para estas tareas. Se utilizarán las librerías scikit-learn para la mayoría de modelos supervisados y modelos convolucionales. Además, para la unidad de reglas de asociación, debido a que este tipo de modelos no se encuentran actualmente dentro de los que implementa scikit-learn, se utilizarán otras librerías que se importarán mediante los procedimientos estándar de Python.

## Scikit-learn

Scikit-learn es una librería de open source para el lenguaje de programación Python que implementa diferentes métodos de aprendizaje automático.

Scikit-learn cuenta con diversos algoritmos para la implementación de los modelos más utilizados en aprendizaje automático, entre los que se incluyen:

* Clasificación.
* Regresión.
* Clúster-
* Reducción de la dimensionalidad.

En la figura 29 se muestra un esquema reproducido de la propia web de scikit-learn, en el que se pueden ver todos los algoritmos disponibles en esta librería y un proceso para seleccionar el más adecuado, en función de los datos disponibles y del análisis que se desea realizar.

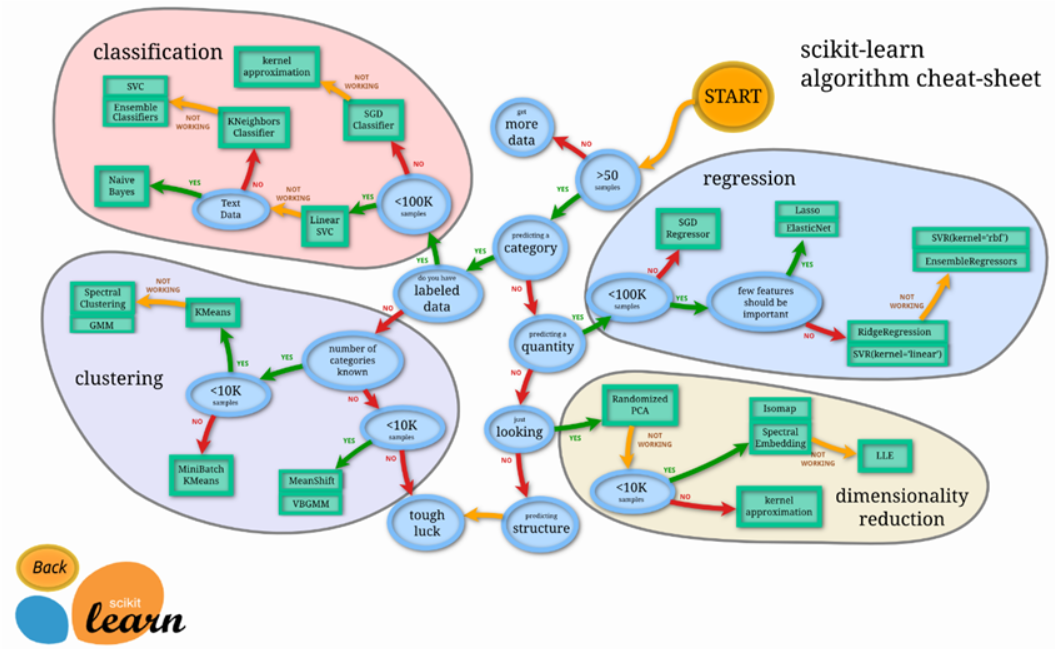


Figura 29: Esquema de los modelos disponibles en Scikit-Learn

Fuente: <http://scikit-learn.org/stable/_static/ml_map.png>

## Overfitting/Underfitting

Las principales causas al obtener malos resultados en Machine Learning son el overfitting o el underfitting de los datos. Cuando se entrena el modelo se intenta ajustar los datos de entrada entre ellos y con la salida. Tal vez se pueda traducir overfitting como «sobreajuste» y underfitting como «sub-ajuste» y hacen referencia al fallo del modelo al generalizar.

Por lo que overfitting se reproduce cuando el modelo tiene demasiados grados de libertad y “memoriza” los datos en lugar de obtener la tendencia que existe en los mismos. En próximas secciones se analizarán algunas técnicas para identificar si un modelo ha sido sobreajustado o no.

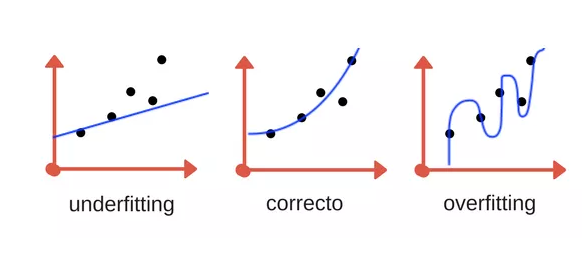


Figura 30: Equilibrio del aprendizaje

Fuente: Aprende Machine Learning

El sobreajuste en un modelo de regresión lineal que se puede detectar fácilmente representado las predicciones realizadas por modelo frente a los datos. Este método presenta un problema cuando existen más de dos variables independientes, ya que no es posible hacer una representación gráfica en estos casos, por lo que es necesario utilizar otra aproximación para identificar si un modelo muestra o no sobreajuste.

Una aproximación es realizar el entrenamiento con un subconjunto de los datos (denominado grupo de entrenamiento) y, al mismo tiempo, guardar otro diferente para la validación (denominado grupo de test), de este modo, si el modelo “memoriza” los resultados del conjunto de entrenamiento, las predicciones en el conjunto de test no serán correctas. Esta técnica se conoce como validación fuera de muestra.

Para intentar que estos problemas afecten los menos posible, se ha llevado a cabo las siguientes recomendaciones:

* Cantidad mínima de muestras tanto para entrenar el modelo como para validarlo.
* Clases variadas y equilibradas en cantidad
* Conjunto de Test de datos. Siempre subdividir nuestro conjunto de datos y mantener una porción del mismo «oculto» a nuestra máquina entrenada
* Parameter Tunning o Ajuste de Parámetros
* Reducción de variables que puedan afectar al sobreajuste/sub-ajuste
* No añadir demasiadas capas ocultas a la red neuronal

Si el modelo entrenado con el conjunto de train tiene un 90% de aciertos y con el conjunto de test tiene un porcentaje muy bajo, esto señala claramente un problema de overfitting.

## Score(R2)

El coeficiente de determinación, se define como la proporción de la varianza total de la variable explicada por la regresión. El coeficiente de determinación, también llamado R cuadrado, refleja la bondad del ajuste de un modelo a la variable que pretender explicar

## Code Machine Learning

Para el proyecto estudio Data Scientist al código genómico Arabidopsis, se ha querido implementar un algoritmo que se ajuste a la tipología de los datos que se presenta, además de construir una red neuronal básica, con el que se pueda hacer comparaciones de resultados entre ambos, y aplicar las buenas prácticas para evitar posibles sobreajustes/sub-ajustes como se ha comentado en el anterior apartado.

Como algoritmo supervisado, se ha implementado **Random Forest**, con el que se ha querido modificar los siguientes parámetros:

* **n\_estimators**: A la hora de crear un modelo Random Forest es importante indicar el número de estimadores que se van a utilizar mediante el comando n\_estimators. En general, cuantos más árboles, menos probabilidades es que el algoritmo se sobreajuste.
* **max\_features:** Reducir este número (probar 30-50% del número de características). Esto determina cuántas entidades se asigna aleatoriamente cada árbol. Cuanto más pequeño, menos probable de sobreajuste
* **max\_depth:** Esto reducirá la complejidad de los modelos aprendidos, reduciendo el riesgo de ajuste excesivo
* **min\_samples\_leaf:** Intentar establecer valores >1. Esto tiene un efecto similar al parámetro max\_depth, significa que la rama dejará de dividirse una vez que las hojas tengan ese número de muestras cada una

Se define la función llamada “random\_forest(data)”, en el que introduciendo el parámetro data, con el que se corresponde los datos al que se quiere implementar el algoritmo, se divide una cantidad de datos para entrenamiento, y otra para test.

Con ello se puede comprobar si alguna de las variables presenta dependencia hacia otras variables, teniendo siempre en cuenta posibles sobreajustes/sub-ajustes a través de los parámetros.

En las siguientes figuras se muestran el tiempo en segundos que ha tardado el algoritmo en entrenar el modelo, posteriormente una matriz compuesto por un conjunto de array [variable dependiente, score\_train y score de valores test ], y finalmente una gráfica donde se compara si los valores score de entrenamiento y test son semejantes o diferentes.

El hecho de conocer gráficamente si los score son valores parecidos, nos ayuda a identificar si el modelo ha aprendido a generalizar.

A continuación, se expone los resultados de 6 implementaciones: 3 implementaciones Random Forest y 3 implementaciones a MLP, en el que se diferenciaran los valores de los parámetros para descubrir mejoras en la predicción.

1. **Random Forest**

\*Parámetros por defecto

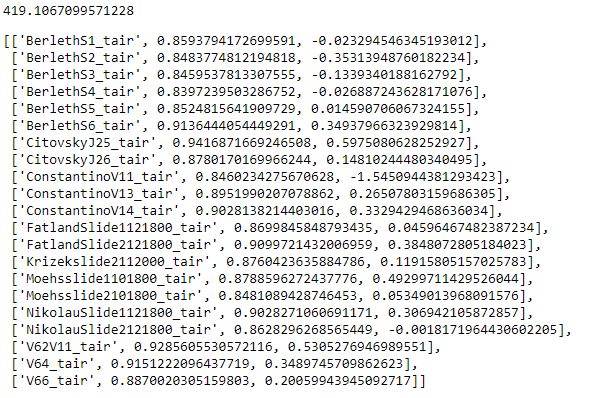


Figura 31: Matriz Output Random Forest I

Fuente: Elaboración propia

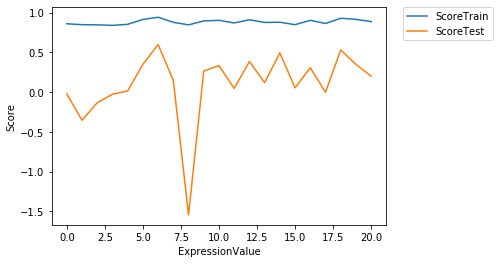


Figura 32: Gráfica RFI Score\_train Score\_test

Fuente: Elaboración propia

1. **Random Forest**

* n\_estimators= 150
* max\_depth= 10
* min\_samples\_leaf=1

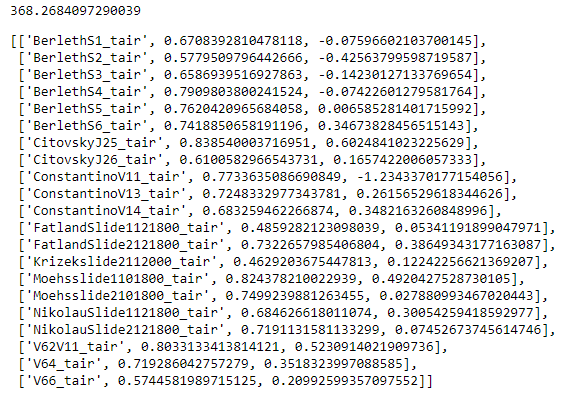


Figura 33: Matriz Output Random Forest II

Fuente: Elaboración propia

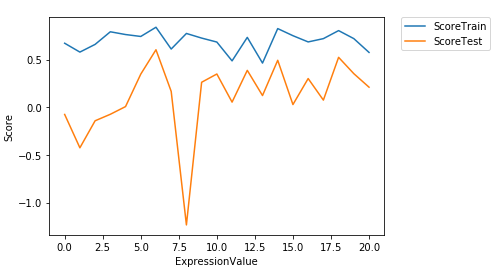


Figura 34: Gráfica RF II Score\_train Score\_test

Fuente: Elaboración propia

1. **Random Forest**

* n\_estimators= 250
* max\_depth= **1**0
* min\_samples\_leaf=10
* max\_features= 20

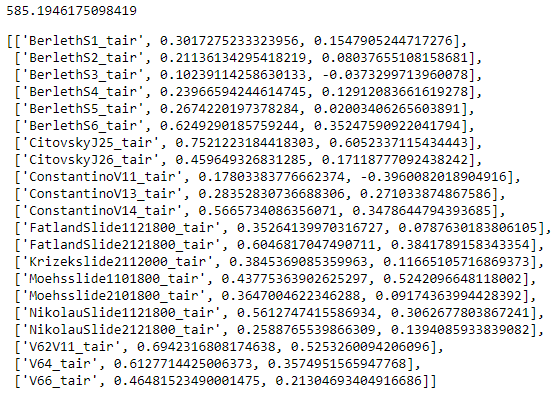


Figura 35: Matriz Output Random Forest III  
Fuente: Elaboración propia

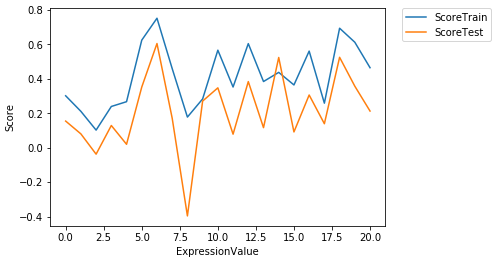


Figura 36: Gráfica RF III Score\_train Score\_test

Fuente: Elaboración propia

1. **MLP**

\*Parámetros por defecto



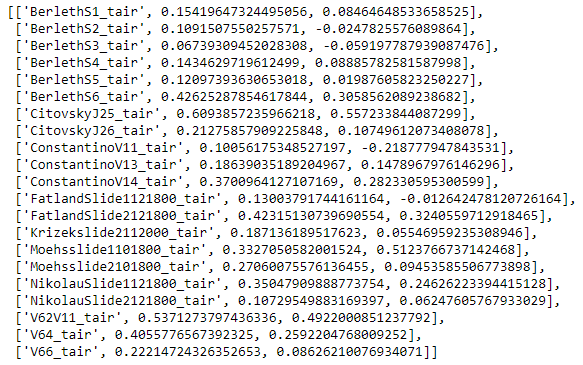


Figura 37: Matriz MLP I Output

Fuente: Elaboración propia

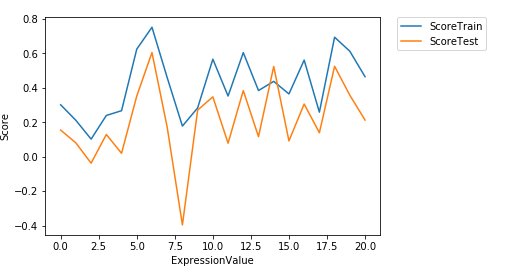


Figura 38: Gráfica MLP I Score\_train Score\_Test

Fuente: Elaboración propia

1. **MLP**

* hidden\_layer\_sizes=(15,10,10,15)
* activation = 'tanh'



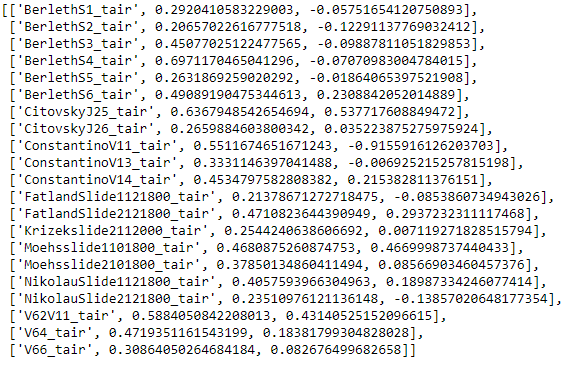


Figura 39: Matriz MLP II Output

Fuente: Elaboración propia

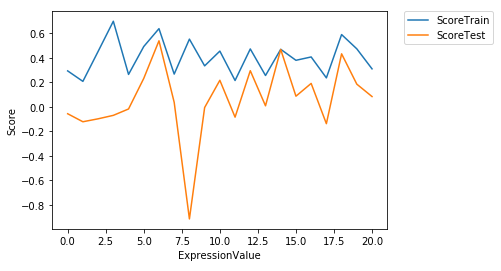


Figura 40: Gráfica MLP II Score\_train Score\_test

Fuente: Elaboración propia

1. **MLP**

* hidden\_layer\_sizes=(15,)



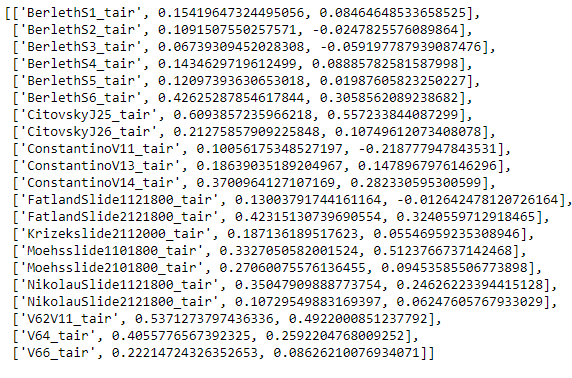


Figura 41: Matriz MLP III Output

Fuente: Elaboración propia

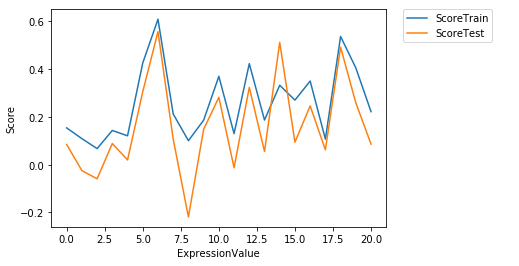


Figura 42: Gráfica MLP III Score\_train Score\_test

Fuente: Elaboración propia

## Fase V: Análisis de los resultados

Como fase final del proyecto, se ha querido a través de dos algoritmos (Random Forest y MLP), conocer el nivel de relación de los genes modificando únicamente los parámetros con el fin de evitar overfitting y aumentar la calidad de predicción.

En el siguiente cuadro se expone de manera resumida los resultados que se han obtenido:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Parámetros por defecto | n\_estimators= 150  max\_depth= 10  min\_samples\_leaf=1 | n\_estimators= 250  max\_depth= **1**0  min\_samples\_leaf=10  max\_features= 20 |
| **Random Forest** |  |  |  |
|  | Parámetros por defecto | hidden\_layer=(15,10,10,15)  activation = 'tanh' | hidden\_layer\_sizes=(15,) |
| **MLP** |  |  |  |

Figura 43: Cuadro Resumen resultados

Fuente: Elaboración propia

Se puede observar que a través de los parámetros el modelo **Random Forest III** y **MLP III** no presentan sobreajuste ya que sus coeficientes de determinación son similares para el conjunto de datos de Train y Test.

Es por ello que conviene tener en cuenta las matrices de los resultados de los algoritmos indicados anteriormente, para poder sacar las conclusiones necesarias sobre los resultados del proyecto.

# Conclusiones

A lo largo del proyecto Data Science se ha elaborado diferentes métodos matemáticos, estadísticos y de programación, capaz de extraer los datos desde su origen, filtrar la información deseada y organizarlos de tal manera que a través de análisis exploratorio de los datos y de los algoritmos implementados se obtengan resultados en las predicciones. Posteriormente tras la visualización de los resultados en todas las fases del proyecto, se ha podido apreciar valores altos de correlación los siguientes ensayos en los genes.

* **BerlethS6**
* **CitovskyJ25**
* **MoeshSlide 110180**
* **V62V11**

Estos resultados significan que, de todos los ensayos realizados en genes, son aquellos que puede existir una relación a la hora de que un gen se exprese.

A pesar que los coeficientes de determinación no han sido demasiado altos para concretar que son variables relacionadas, sí que han sido los que han expresado mejores resultados

Aun así, existen otros enfoques que se puede realizar en proyectos futuros como la disposición de los datos tras su ETL, en el que se aplica los métodos Machine Learning directamente a los genes como atributos.

También sería interesante construir una red neuronal densificada a través de Keras junto con Cross Validation para conocer si los resultados en la predicción mejoran.

Finalmente me gustaría incluir que este proyecto se enmarca dentro del Máster Business Analytics and Big Data, y las asignaturas de dicho máster que han aportado una gran de conocimientos y herramientas que han facilitado el desarrollo del proyecto.

Su desarrollo ha sido muy satisfactorio además de interesante, ya que me ha permitido adentrarme en el tratamiento de los datos, análisis a través de métodos matemáticos y estadísticos y posteriormente la implementación de algoritmos Machine Learning y Deep Learning.

# Bibliografía

**Bootstrapping Machine Learning**. Dorard, L. [En línea] URL disponible en <http://www.louisdorard.com/machine-learning-book/>

**Hands-On Machine Learning with Scikit-Learn & TensorFlow.** Aurélien Géron. O’REILLY; 2017

**Python for Data Analysis.** Wes Mckinney O’REILLY

**Data Science and Predictive Analytics** Ivo D.Dinov . Srpinger 2018

**scikit-learn**

<https://scikit-learn.org/stable/>

**Feature Selection with sklearn and Pandas** Abhini Shetye, Feb 11, 2019

<https://towardsdatascience.com/feature-selection-with-pandas-e3690ad8504b>

**Feature selection using Wrapper methods in Python.** Vikashraj Luhaniwal, Oct 4, 2019

<https://towardsdatascience.com/feature-selection-using-wrapper-methods-in-python-f0d352b346f>

**Sets de Entrenamiento, Test y Validación**

<https://www.aprendemachinelearning.com/sets-de-entrenamiento-test-validacion-cruzada/>

**rico-schmidt.name,** 2019 Ernesto Rico Schmidt

<https://rico-schmidt.name/pymotw-3/index.html>

**Machine Learning Supervisado: Fundamentos de la Regresión Lineal,** Victor Roman, Feb 27, 2019

<https://medium.com/datos-y-ciencia/machine-learning-supervisado-fundamentos-de-la-regresi%C3%B3n-lineal-bbcb07fe7fd>

**Recursive Feature Elimination,** 2016-2019, The scikit-yb developers

<https://www.scikit-yb.org/en/latest/api/model_selection/rfecv.html#recursive-feature-elimination>

# ANEXOS I: Desarrollo ETL

## Introducción

Actualmente, tanto las empresas como las organizaciones invierten en proyectos de inteligencia de negocio con el objetivo de optimizar sus procesos, su actividad y ser capaces de medir su desempeño.

Su objetivo es mejorar el proceso de toma de decisiones generando un repositorio único de todas las fuentes que ofrezca una visión unificada y veracidad del dato.

Este anexo se profundizará en el estudio y desarrollo de una de las tres piezas más importantes de una solución de inteligencia de negocio: los procesos ETL. Estos van a constituir el motor de procesamiento de información y son los responsables de garantizar la calidad del dato.

Como se ha explicado en la memoria, el proceso ETL responde a las siglas de extracción, transformación y carga de los datos.

La **extracción** tiene el objetivo de esta fase es la extracción de los datos de cada una de las fuentes identificadas, realizando una copia exacta (sin procesamiento) de los datos de origen.

La **transformación** de los datos en el proceso de ETL es el segundo de los procesos y consiste, principalmente, en limpiar y conformar la información extraída de la fuente.

La **carga** es el paso final del proceso y su objetivo es realizar de forma eficiente, la carga de datos en el almacén de datos y, más concretamente, en cada uno de los modelos multidimensionales definidos.

Para ello se van a definir funciones a través del lenguaje de programación Python.

Python es un lenguaje de programación que cada vez se utiliza más por las empresas y programadores que trabajan con datos (Business Intelligence, Integración de datos, Data Science, Machine Learning, Big Data…). El motivo de que cada vez cobra más importancia en su uso es en la gran cantidad de librerías existentes para realizar prácticamente todo y más aún si el objetivo es trabajar y gestionar datos, también por lo optimizado que está Python.

Se ha desarrollado un proceso ETL con Python desde cero en el que se va hacer uso de las principales librerías más conocidas en la manipulación y tratamiento de los datos (Pandas) y aplicación de fórmulas matemáticas a matrices multidimensionales (Numpy), implementado y desarrollados en notebooks de Jupyter a través de la instalación de Anaconda.

* ETL
* Anaconda
* Pandas
* Numpy
* Código explicado
* Resultado

## Anaconda

Anaconda es una distribución libre disponible tanto para Windows, macOS como Linux del lenguaje de programación Python para el procesamiento a gran escala de datos, análisis predictivo y la computación científica, que tiene como objetivo simplificar la gestión y distribución de paquetes. Para esto utiliza el sistema de gestión llamado Conda.

La última versión de esta distribución se puede encontrar en la página web de [Anaconda](https://www.anaconda.com/products/individual).

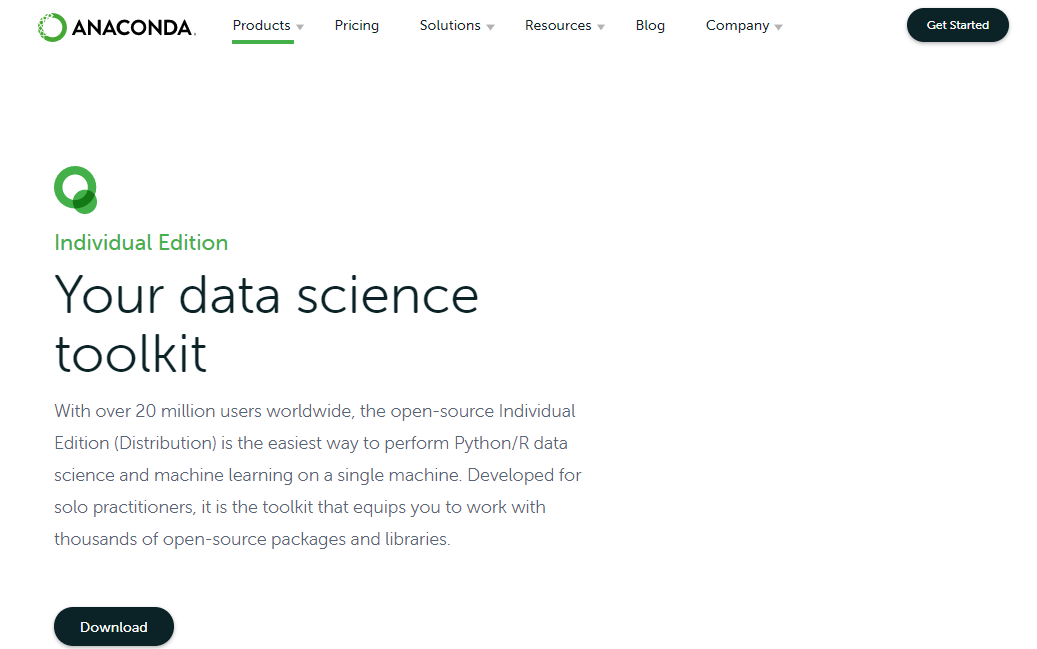


Figura 44: Página principal descarga Anaconda

La instalación de Anaconda es sencilla: para cada uno de los sistemas operativos existe un instalador que realiza todas las tareas de forma automática. En los sistemas UNIX (macOS y Linux) no es necesaria una cuenta de administrador, ya que la instalación se puede realizar en la cuenta de usuario. El instalador de Anaconda configura un entorno con Python, Jupyter Notebook, pandas, scikit-learn y otras aplicaciones y librerías, dejando el sistema listo para trabajar. Además, instala el programa Anaconda Navigator, a través del cual se pueden lanzar las aplicaciones, al mismo tiempo que instalar y actualizar paquetes.

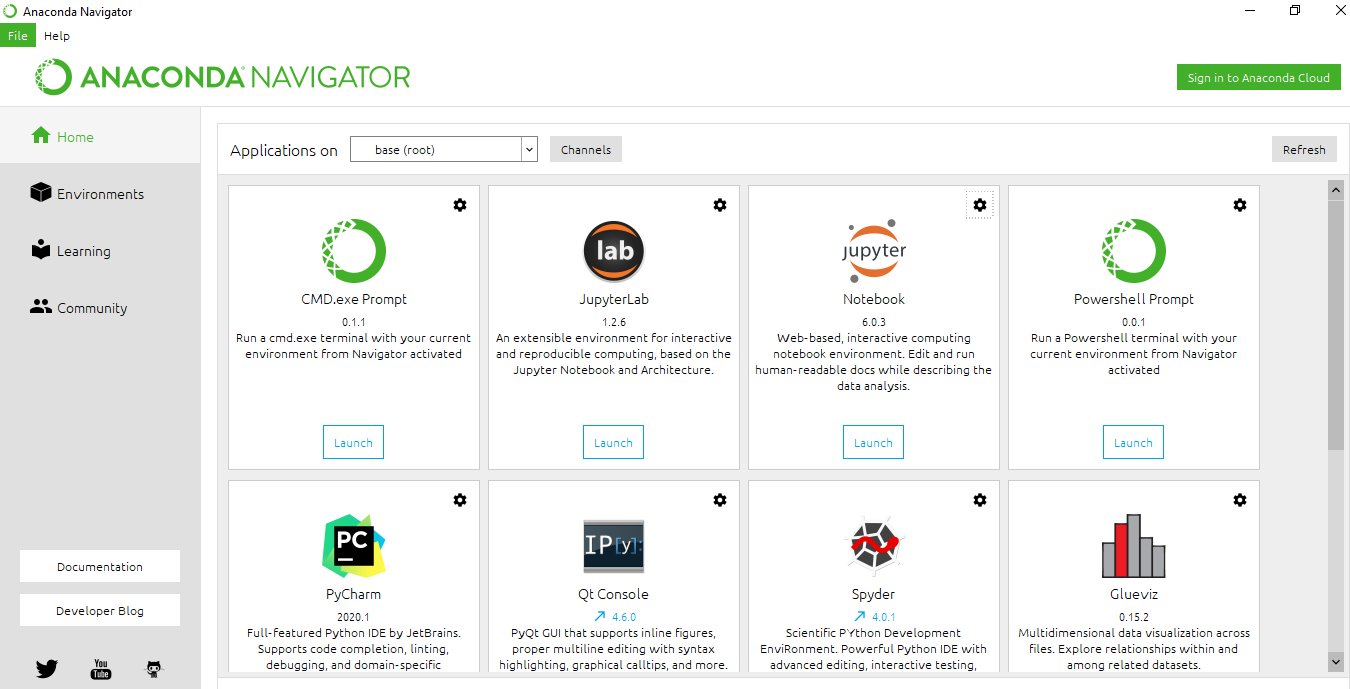


Figura 45: Navegador Anaconda

# Notebook Jupyter

A través del programa Anaconda Navigator se puede lanzar, en el navegador Jupyter Notebook, el editor que se utilizará para ejecutar el código Python durante el proyecto. Alternativamente se puede utilizar en la línea de comandos la instrucción:

jupyter notebook

Esto posibilitará lanzarlo en el navegador Jupyter Notebook. En este caso, aparecerá en el navegador una página como la que se muestra en la Figura 3, donde se puede navegar por el sistema en busca de los archivos.

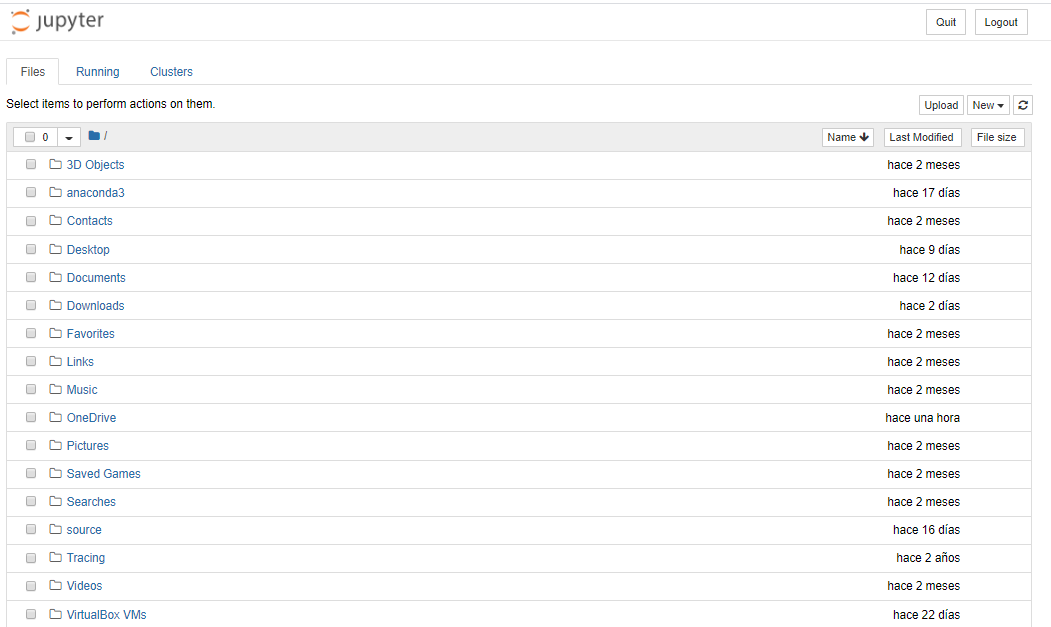


Figura 46: Jupyter Notebook

## Librerías Python

En programación, una librería es un archivo o conjunto de archivos que se utilizan para facilitar la programación.

A continuación, se van a nombrar las librerías que se han hecho uso para el proceso ETL

# Pandas

La librería de Pandas proporciona estructuras de datos fáciles de usar como funciones diseñados para hacer el trabajo con estructuras de datos más rápido, fácil y expresivo.

Esto es una de las razones por lo que Python es una poderosa y productiva herramienta en un entorno de análisis de datos.

Pandas funciona en las estructuras de datos a través de dataframe, que son datos tabulares de tipo heterogéneo con etiquetas en columnas y filas.

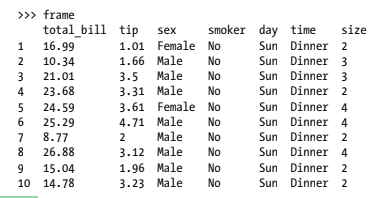


Figura 47: Dataframe

Pandas combina las características de computación de matrices en alto rendimiento de Numpy con la flexibilidad de manipulación de datos en las hojas de cálculo y base de datos relacionales (como SQL).

Proporciona una sofisticada funcionalidad de indexación para facilitar la remodelación, cortar y cortar, realizar agregaciones y seleccionar subconjuntos de datos.

Para los usuarios financieros, los pandas cuentan con una rica funcionalidad de series de tiempo de alto rendimiento y herramientas adecuadas para trabajar con datos financieros. De hecho, inicialmente se diseñó pandas como una herramienta ideal para aplicaciones de análisis de datos financieros.

Para los usuarios del lenguaje R para computación estadística, el nombre del Dataframe será familiar, ya que el objeto recibió el nombre del objeto similar R data.frame. No son lo mismo, sin embargo; la funcionalidad proporcionada por data.frame en R es esencialmente estricta subconjunto del proporcionado por el pandas Dataframe.

El nombre de los pandas en sí se deriva de los datos del panel, un término econométrico para multi conjuntos de datos estructurados mensuales y el análisis de datos de Python en sí.

Por otro lado, también se encuentra otro tipo de objeto llamado Serie. Una serie es un objeto, como un array, que está formado por un array de datos y un array de etiquetas denominado índice.

Las series actúan de forma similar a ndarray, siendo un argumento válido de funciones de NumPy, como un diccionario de tamaño fijo en el que se pueden gestionar los valores a través de los índices y Sobre las series se pueden realizar operaciones vectoriales.

La principal diferencia entre series y ndarray es que las operaciones entre series alinean automáticamente los datos basados en las etiquetas, de forma que pueden realizarse cálculos sin tener en cuenta si las series sobre las que se operan tienen las mismas etiquetas.

# Librería Numpy

NumPy, abreviatura de Numerical Python, es el paquete fundamental para la investigación científica. poniendo en Python. La mayoría de este libro se basará en NumPy y en las bibliotecas construidas. encima de NumPy. Proporciona, entre otras cosas:

* Un objeto de matriz multidimensional rápido y eficiente ndarray
* Funciones para realizar cálculos de elementos sabios con matrices o matemáticas operaciones entre matrices
* Herramientas para leer y escribir conjuntos de datos basados ​​en matrices en el disco
* Operaciones de álgebra lineal, transformación de Fourier y generación de números aleatorios.
* Herramientas para integrar código de conexión C, C ++ y Fortran a Python

Más allá de las capacidades rápidas de procesamiento de matriz que NumPy agrega a Python, uno de sus los propósitos principales con respecto al análisis de datos es como el contenedor primario para que los datos ser pasado entre algoritmos. Para datos numéricos, las matrices NumPy son mucho más forma eficiente de almacenar y manipular datos que los otros datos integrados de Python estructuras

Por ello su principal ventaja es la eficiencia para manipular vectores y matrices. En este sentido, proporciona funciones que operan sobre ndarrays.

# Librería Re

Las expresiones regulares son patrones de coincidencia de texto descritos con una sintaxis formal. Los patrones se interpretan como un conjunto de instrucciones, que luego se ejecutan con una cadena como entrada para producir un subconjunto de coincidencia o una versión modificada del original. El término «expresiones regulares» con frecuencia se acorta en conversación a «regex» o «regexp». Las expresiones pueden incluir correspondencia de texto literal, repetición, composición de patrones, ramificación y otras reglas sofisticadas. Una gran cantidad de problemas de análisis son más fáciles de resolver con una expresión regular que mediante la creación de un analizador léxico de propósito especial y un analizador sintáctico.

Las expresiones regulares se usan normalmente en aplicaciones que implican procesamiento de una gran cantidad de texto. Por ejemplo, se usan comúnmente como patrones de búsqueda en los programas de edición de texto utilizados por los desarrolladores, incluido vi, emacs e IDEs modernos. También son una parte integral de utilidades de Unix de línea de comandos como sed, grep y awk. Muchos lenguajes de programación incluyen soporte para expresiones regulares en el sintaxis (Perl, Ruby, Awk y Tcl). Otros lenguajes, como C, C ++, y Python, admiten expresiones regulares a través de bibliotecas de extensión.

Existen múltiples implementaciones de código abierto de expresiones regulares, cada una compartiendo una sintaxis común, pero con diferentes extensiones o modificaciones a sus características avanzadas. La sintaxis utilizada en el módulo ***re*** en Python se basa en la sintaxis utilizada para las expresiones regulares en Perl, con algunas mejoras específicas de Python.

A continuación, se exponen algunos ejemplos de las finalidades de las expresiones regulares:

* Encontrar patrones en el texto
* Compilar expresiones
* Coincidencias múltiples
* Sintaxis de patrones
* Repetición
* Conjunto de caracteres

# Librería Zipfile/Os

Estas dos últimas librerías es para uso más específico. Se puede usar para manipular archivos comprimidos ZIP.

Librería ‘Os’ permite acceder a funcionalidades dependientes del Sistema Operativo. Sobre todo, aquellas que nos refieren información sobre el entorno del mismo y nos permiten manipular la estructura de directorios.

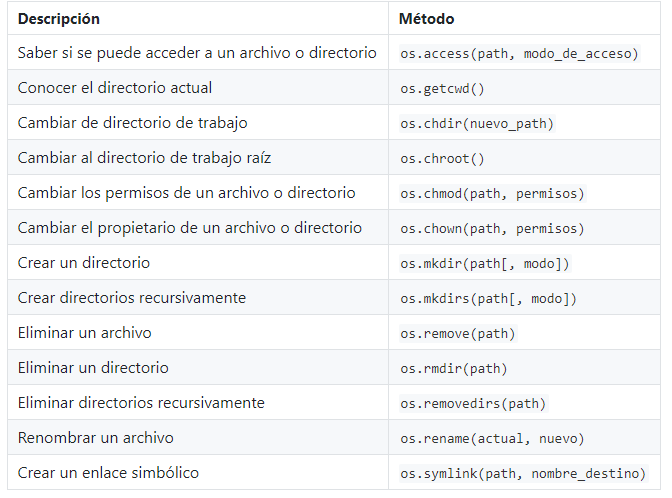


Figura 48: Principales funcionalidades de librería 'Os'

Fuente: <https://rico-schmidt.name/pymotw-3/re/index.html>

# Desarrollo ETL

A través del lenguaje de programación Python se define diferentes funciones con el que se realiza de la manera más optimizada la extracción, transformación y carga de los datos que posteriormente se van a implementar en modelos de machine Learning.

El notebook FASE I Extracción, tratamiento y carga de los datos, se ha querido definir tres tipos de funciones de manera que se consiga modularizar la función ETL. De esta manera se consigue una mayor facilidad de lectura y comprensión para identificar la finalidad de cada proceso y, por otro lado, no interferir entre ellas en el caso que se produzca un error.

### Importación de librerías

En un primer lugar se procede a la importación de las librerías necesarias:



Tras importar las funciones, se definen as funciones que se han definido se corresponde a extract(), tranform\_load() y clean\_duplicate().

### Función extract()





La función extract() tiene como entrada dos argumentos( la dirección de los ficheros comprimidos, y la dirección donde se quiere guardar los ficheros descomprimidos).

En primer lugar, la función extract() es definir las expresiones regulares con el que se quiere encontrar los ficheros que interesan descomprimir según un patrón común entre ellos.

El patrón que se cumple son aquellos ficheros que están compuestos por la extensión “\_tair.txt”.

Una vez esto, se formula un bucle en el que recorra todos los ficheros, y aquellos que cumplan con el patrón definido por la expresión regular, se descomprime con la función zipfile() y se guarda en el fichero definido en el argumento de entrada de la función.

De esta manera se realiza la extracción de los ficheros en el proceso de ETL.

### Función tranform load()

La función tranform\_load() tiene como finalidad limpiar y almacenar los datos que interesa para el proyecto en una estructura de datos, que en este caso es en un dataframe. Alternativamente también se almacena en un archivo .CSV para en procesos posteriores poder trabajar con los datos ya sea en diferentes lenguajes de programación, como puede ser R.





La primera parte de la función consiste en crear un dataframe vacío, con el que se posteriormente se va a ir incluyendo los datos que interesan.

A continuación, se vuelven a definir los patrones de las expresiones regulares, con el que se va a extraer el nombre del gen para ya incluirlo al dataframe tras una serie de funciones “filtro”, con el que se van a limpiar los valores duplicados y valores nulos.

Finalmente, estos datos limpiados se incluyen en el dataframe.

### Función clean duplicate()

Mientras se realizaba el código, se encontró con que algunos genes estaban solapados, esto quiere decir que el formato de presentación de los datos en los ficheros, en ocasiones presentaba la información asociada a dos genes no por separado, sino asociada a una lista de genes. Se consideró que esa información era la misma para cada uno de los genes, por lo que se procedió a separar la lista de genes y asociar a cada uno de los genes dicha información. Un ejemplo de ello:

Es necesario este paso intermedio, porque si no el programa reconoce la lista de genes como un gen diferente, cuando en realidad se trata de dos genes, se perdería información.

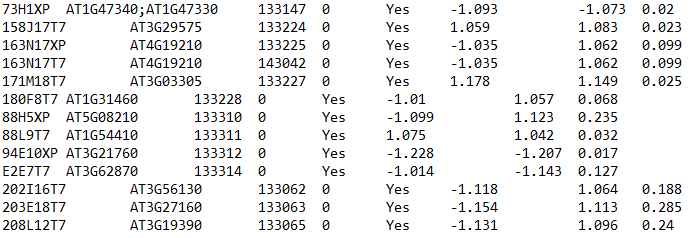


Figura 51: Genes solapados

Fuente: Datos AGDA .txt

Con el fin de evitar futuros errores y cumplir con un formato uniforme en la elaboración del dataframe, se define la función clean\_duplicate() con el que separa los genes solapados, y se les asigna el valor de expresión que se le corresponde.



La clave de esta función identificar que genes presentan el carácter “;” para a través de un bucle “for”, los separe de manera que almacene su valor de expresión, y posteriormente lo añada al dataframe.

# Anexo II: Ingeniería de características y selección de modelos

# Introducción

El objetivo de esta fase del proyecto, como se ha indicado en la memoria, es identificar en un conjunto de datos las características más adecuadas, proceso conocido como Features Selection.

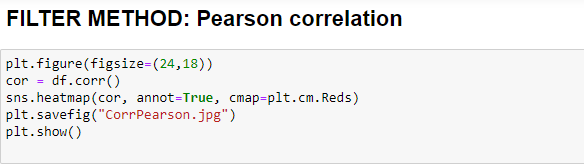
Feature Selection es el proceso de seleccionar las más importante y/o relevantes características de un conjunto de datos, con el objetivo de mejorar el rendimiento de predicción de los predictores, proporcionar predictores más rápidos y más rentables y proporcionar una mejor comprensión del proceso subyacente que generó los datos.

# Desarrollo Ingeniería de características y selección de modelos

A continuación, se expone como se ha desarrollado los métodos de Features Selection compuesto por un método Filter Based, Wrapper-based y Embedded.

**FILTER BASED**

Para este método se ha llevado a cabo la función de correlación que ofrece la librería de Pandas **corr(),** en el que calcula la correlación entre las variables y junto con la librería matplotlib se representa en una gráfica ([Figura 19](#_Filter_Based))



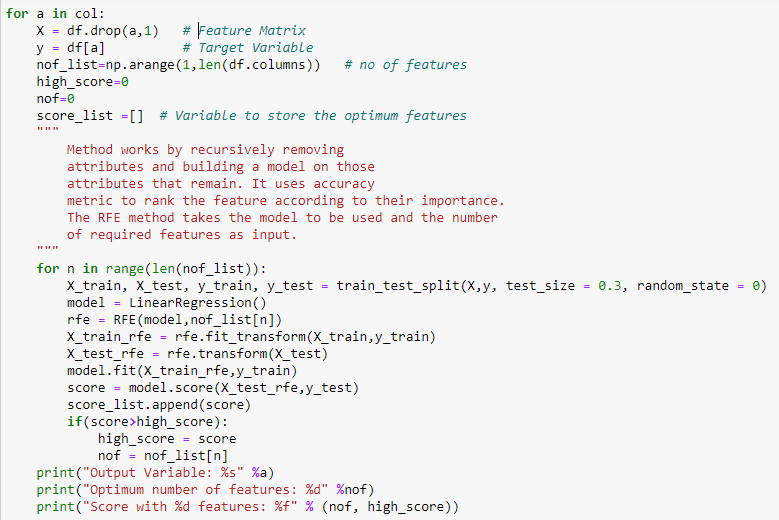
Tras obtener la matriz de correlación de Pearson, se filtra aquellas variables que presenten una correlación mayor de 0,4 y menor de -0,4 para conocer cuales están directamente e indirectamente relacionados.



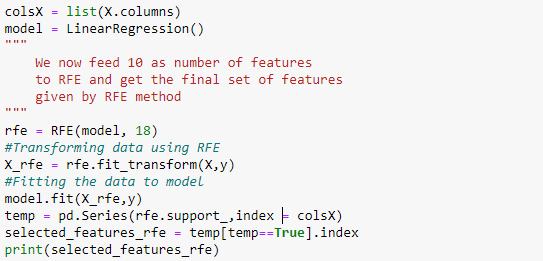


**WRAPPED METHOD: RFE**

El siguiente notebook tiene el objetivo de encontrar el número óptimo con el fin de conocer que rango de clasificación a la hora de escoger que variables ayudan a obtener mayor precisión.

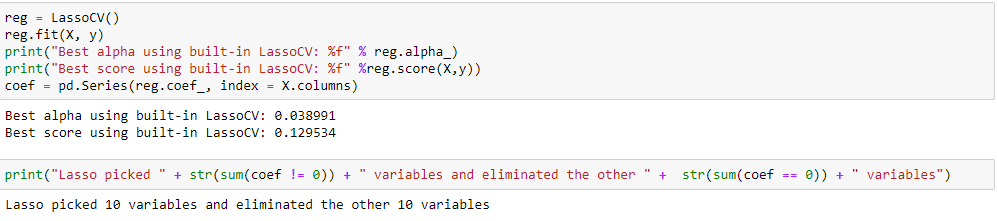


Tras conocer el resultado del número óptimo de rango de clasificación, que en este caso es 18, se desarrolla método RFE.

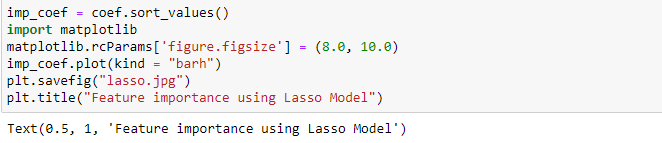


**METHOD EMBEDDED: LASSON**

La función que indica el error cometido por el modelo en función de los parámetros y es la que se minimiza utilizando el algoritmo del gradiente descendente para la obtención de los parámetros del modelo.



Desarrollo Python a través de librería **matplotlib** para representación gráfica de los resultados en las variables ([Figura 25](#_Embedded_Method:_Lasson)).



# ANEXO III: Implementación de modelos Machine Learning

# Introducción

En la siguiente fase del proyecto tiene como objetivo averiguar el nivel de relación de los genes de una planta Arabidopsis Thailiana a través de métodos Machine Learning, teniendo en cuenta todas las variables que los anteriores métodos se han calificado como importantes.

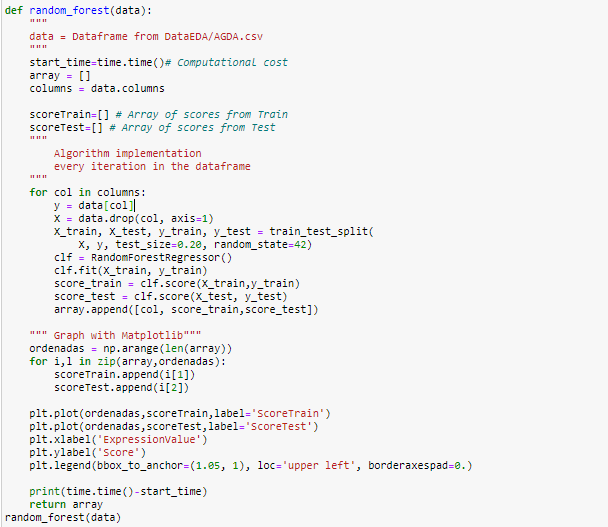
Durante este apartado se va a exponer las implementaciones de algoritmos Machine Learning de aprendizaje supervisado, es decir, se va a etiquetar como variable dependiente a través de iteraciones cada variable del conjunto de datos para posteriormente a través de los resultados de los algoritmos conocer cuales presentan mejores resultados como variable dependiente.

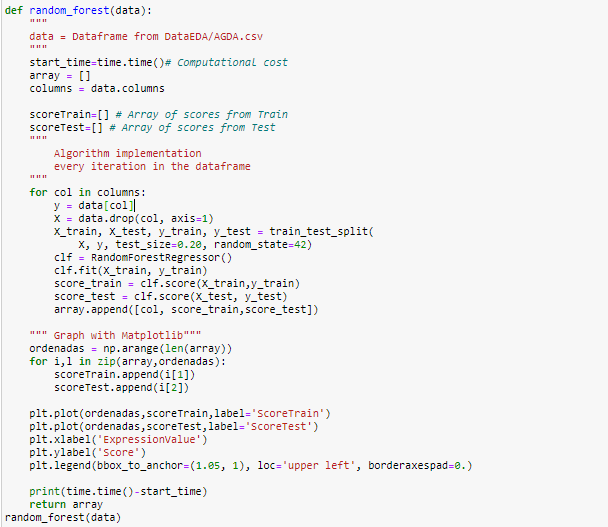
Para ello se ha querido seleccionar un algoritmo de aprendizaje supervisado: Random Forest, y un algoritmo multilayer perceptrón: MLP a través de la librería scikit-learn.

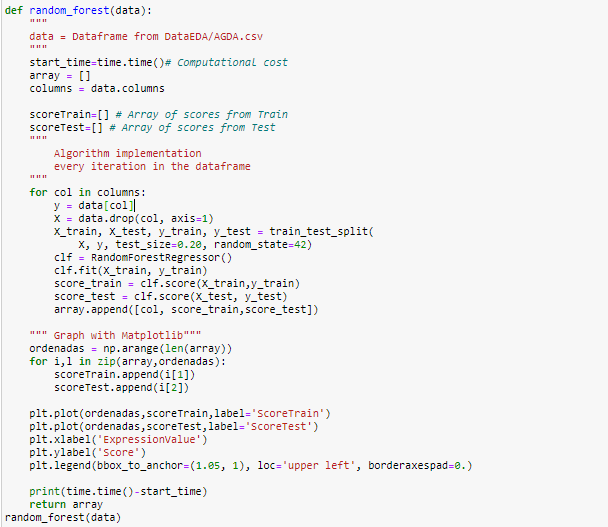
Con estos dos algoritmos se ha experimentado a través de que parámetros responde a mejores resultados evitando sobreajuste y mayor coeficiente de determinación en el aprendizaje.

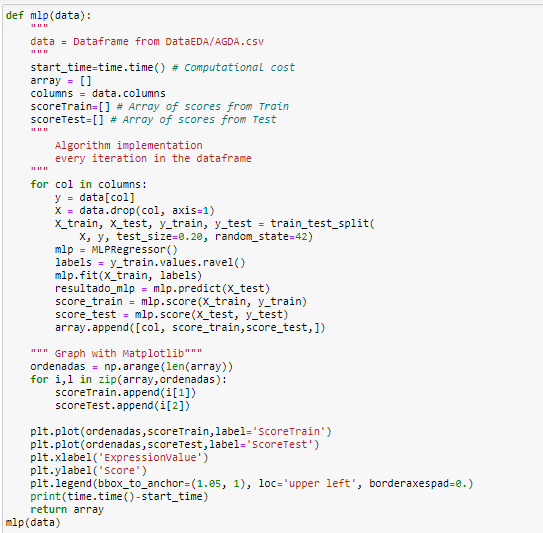
# Desarrollo de algoritmos Machine Learning

A continuación, se expone como se ha desarrollado e implementado los algoritmos machine Learning Random Forest y MLP a través de los datos procesados, en el que se ha querido modificar los parámetros para visualizar los resultados.









1. Valor atípico es una observación que es numéricamente distante del resto de los datos [↑](#footnote-ref-1)
2. [Pearson](https://en.wikipedia.org/wiki/Pearson_correlation_coefficient) [↑](#footnote-ref-2)
3. Librería de Python para manipulación de datos [↑](#footnote-ref-3)
4. Librería de Python para representación de gráficas [↑](#footnote-ref-4)
5. [RFE](https://www.scikit-yb.org/en/latest/api/model_selection/rfecv.html) [↑](#footnote-ref-5)