GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

APLICATIVO WEB PARA LA GENERACIÓN DE MODELOS DE MACHINE LEARNING Y REALIZACIÓN DE PREDICCIONES SOBRE DATOS ARBITRARIOS.

Autor: Daniel Bicand Fernández

Director: Carlos Morrás Ruiz-Falcó

Madrid

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

APLICATIVO WEB PARA LA GENERACIÓN DE MODELOS DE MACHINE LEARNING Y REALIZACIÓN DE PREDICCIONES SOBRE DATOS ARBITRARIOS

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el

curso académico 2021/2022 es de mi autoría, original e inédito y

no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido

tomada de otros documentos está debidamente referenciada.

Fdo.: Daniel Bicand Fernández Fecha: 12/ 08/ 2022

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Firmado por MORRAS RUIZ-FALCO CARLOS - ***0907** el día 23/08/2022 con un certificado emitido por AC FNMT Usuarios

Fdo.: Carlos Morrás Ruiz-Falcó Fecha: ……/ ……/ ……

GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

APLICATIVO WEB PARA LA GENERACIÓN DE MODELOS DE MACHINE LEARNING Y REALIZACIÓN DE PREDICCIONES SOBRE DATOS ARBITRARIOS.

Autor: Daniel Bicand Fernández

Director: Carlos Morrás Ruiz-Falcó

Madrid

APLICATIVO WEB PARA LA GENERACIÓN DE MODELOS DE MACHINE LEARNING Y REALIZACIÓN DE PREDICCIONES SOBRE DATOS ARBITRARIOS.

Autor: Bicand Fernández, Daniel.
Director: Morrás Ruiz-Falcó, Carlos.
Entidad Colaboradora: ICAI – Universidad Pontificia Comillas

RESUMEN DEL PROYECTO

En este proyecto de fin de grado se ha desarrollado una aplicación web capaz de realizar predicciones haciendo uso de algoritmos de machine learning (ML) entrenados con datos arbitrarios subidos por usuarios a la web.

La aplicación contiene 9 distintos algoritmos de ML, 4 de ellos son de regresión y el resto de clasificación. Además, la aplicación dispone de un módulo de guardado de modelos entrenados para que los usuarios accedan a ellos sin necesidad de reentrenarlos.

Por último, la aplicación incluye de un sistema de limpieza de datos básico que elimina o sustituye los valores vacíos de las bases de datos subidas a la web por los usuarios.

Palabras clave: Machine Learning, Software, WebApp, DataScience, Data Analytics

1. Introducción

A lo largo de la historia, se han producido varias revoluciones industriales que han impulsado la productividad en la humanidad y han mejorado significativamente las condiciones de vida en la tierra. Se dice, que actualmente estamos viviendo la cuarta revolución industrial, la revolución de la Inteligencia Artificial (IA).

Una de las ramas de la Inteligencia Artificial es el Machine Learning. El Machine Learning o ML es un tipo de IA que proporciona a las computadoras capacidad de aprender sin ser explícitamente programadas para ello. Es decir, que, dados unos datos, los algoritmos de ML son capaces de aprender sobre ellos y realizar predicciones con nuevos datos.

Este proyecto tiene como objetivo acercar a los perfiles menos técnicos la utilidad de esta tecnología ya que tarde o temprano, será indispensable tener acceso a herramientas como esta para mantenerse competitivo en el mercado.

2. Definición del proyecto

El diseño del proyecto consiste en crear una aplicación web, que sirva como capa de abstracción para la generación de modelos predictivos. Es decir, embeber la lógica de creación y entrenamiento de modelos predictivos dentro de un entorno web para que los usuarios puedan crear estos modelos con tan solo unos clicks.

Para ello, se debe crear un entorno donde los usuarios puedan subir las bases de datos que quieran utilizar. Después de añadir una capa de limpieza y preparación de datos básica. Los usuarios deben tener acceso a los distintos algoritmos de machine learning y poder editar fácilmente los parámetros y las variables de entrenamiento.

Por último, los usuarios deben ser capaces de realizar predicciones con los modelos entrenados, además de guardar sus modelos para usarlos cuando necesiten.

3. Descripción de la aplicación web

Este proyecto ha sido desarrollado utilizando el framework web Django. Este framework de desarrollo está basado en Python y utiliza una arquitectura llamada MVT (Modelo Vista Template).

La arquitectura MVT es parecida a la arquitectura MVC usada por otros frameworks como SpringBoot. La diferencia es que en la arquitectura MVT el framework se encarga del controlador, facilitando así el desarrollo.

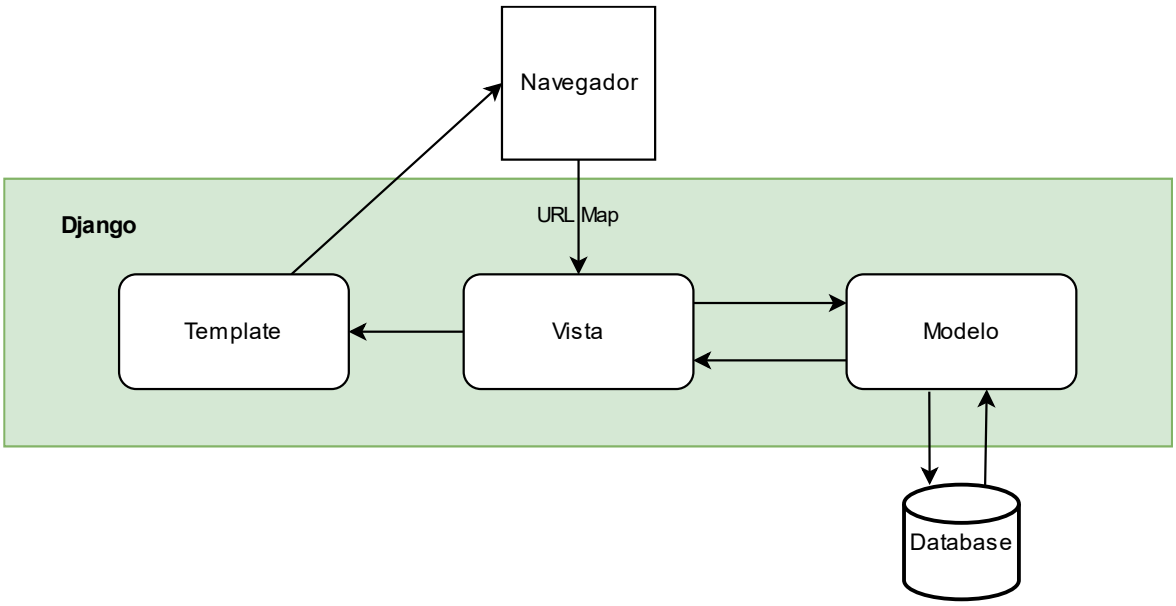


Ilustración 1 - Arquitectura MVT

Un proyecto de Django se divide en varias aplicaciones, cada una con una funcionalidad especifica. Este proyecto contiene 4 aplicaciones distintas:

1. SandboxApp
2. Databases
3. ML
4. Models

La primera es la aplicación principal, su funcionalidad es cargar el template de inicio y redirigir a los usuarios al resto de aplicaciones.

La aplicación de Databases contiene la lógica de subida, borrado y limpieza de bases de datos. Desde aquí, los usuarios seleccionan la base de datos que quieran usar para generar los modelos predictivos.

La aplicación ML es el núcleo del proyecto y contiene toda la lógica de creación y entrenamiento de modelos predictivos. Desde esta aplicación se puede seleccionar cualquiera de los 9 algoritmos disponibles y generar un modelo básico. También, para los usuarios más avanzados, existe la opción de tunear los hyper-parametros del modelo.

Por último, la aplicación Models permite al usuario guardar los modelos entrenados y realizar predicciones sobre ellos.

El diagrama de flujo para la generación de un modelo es el siguiente:

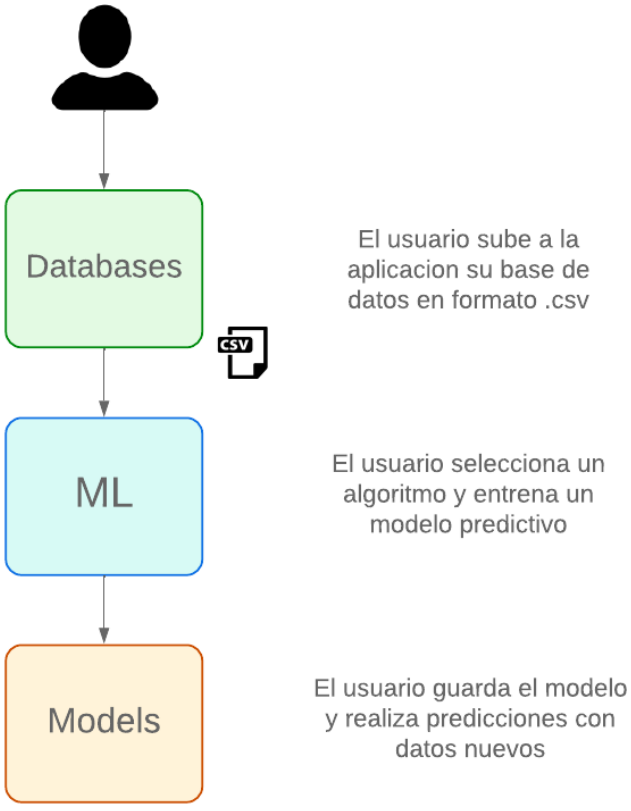


Ilustración 2 - Diagrama de flujo de la generación y uso de un modelo

4. Resultados

Finalmente, se ha desarrollado una aplicación que cumple con los objetivos definidos en el apartado 2. La aplicación es capaz de generar modelos y realizar predicciones sobre cualquier fichero .csv sin tener que escribir ni una sola línea de código.

En las siguientes ilustraciones se puede observar el proceso desde que el usuario selecciona los datos para crear un modelo a su gusto, hasta que está de acuerdo con su precisión, guarda el modelo y realiza una predicción.

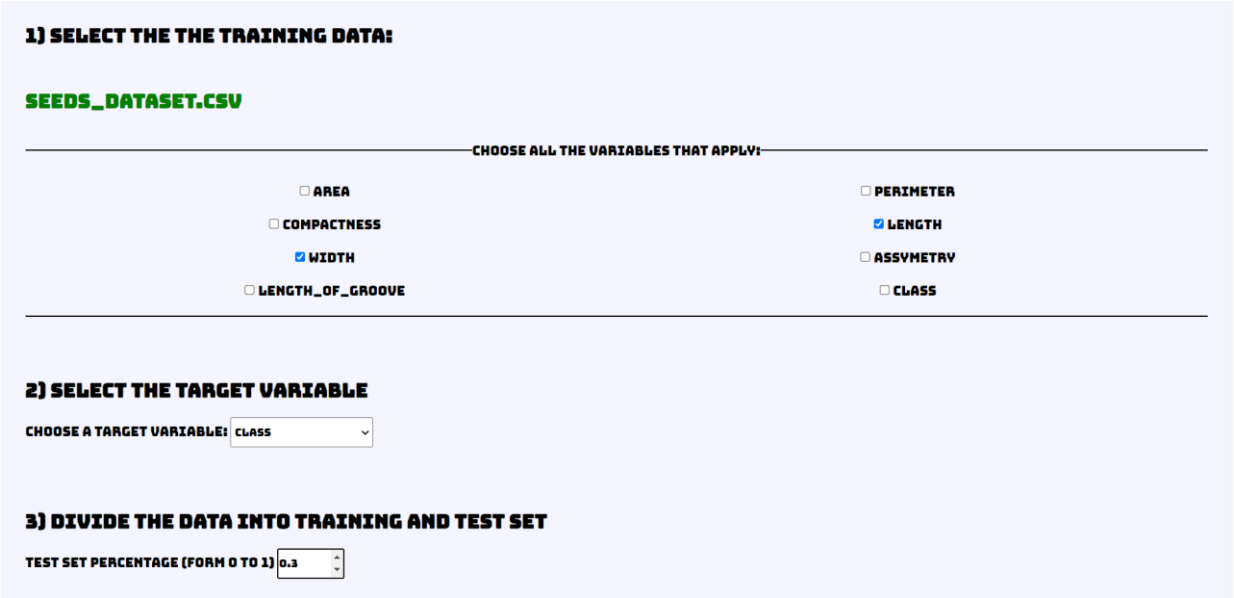


Ilustración 3 - Datos para la generación de un modelo SVM

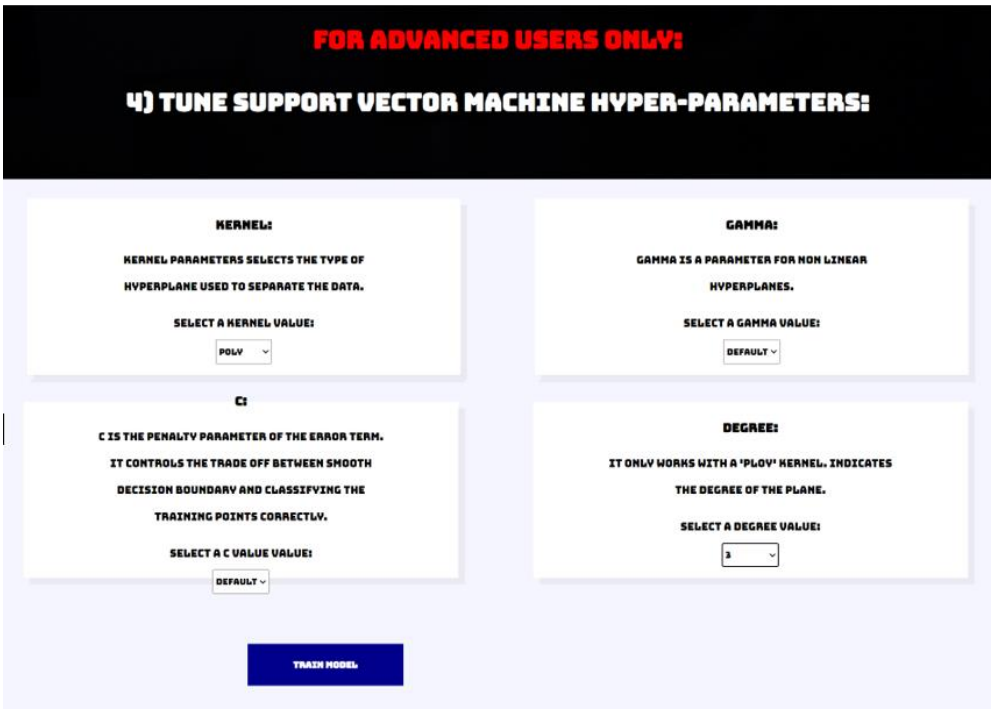


Ilustración 4 - Tuneado de hyper-parametros

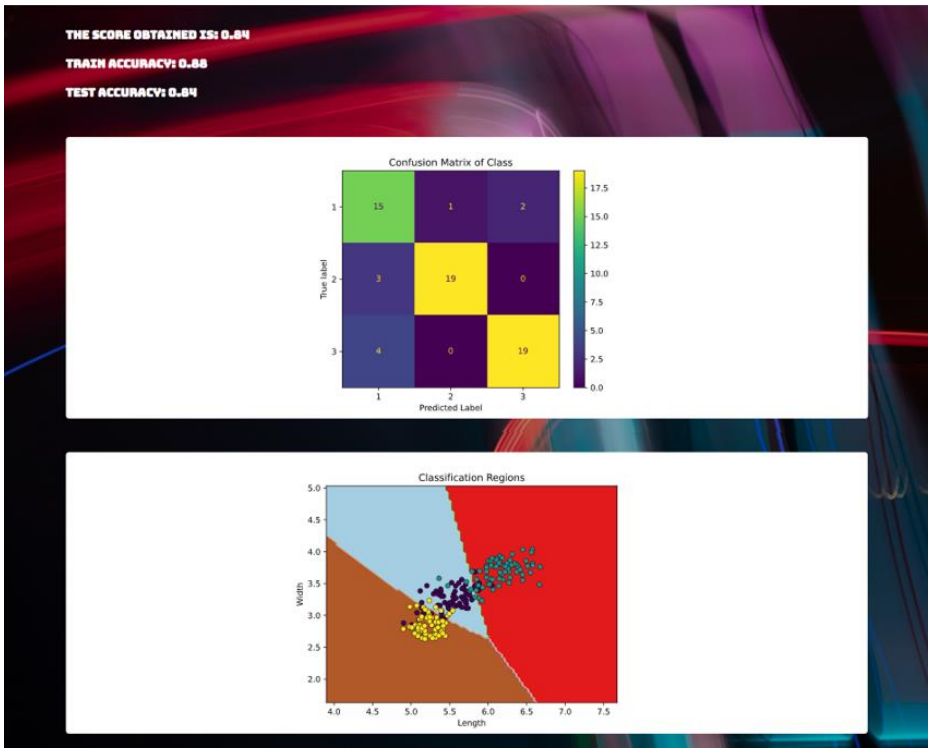


Ilustración 5 - Output del modelo

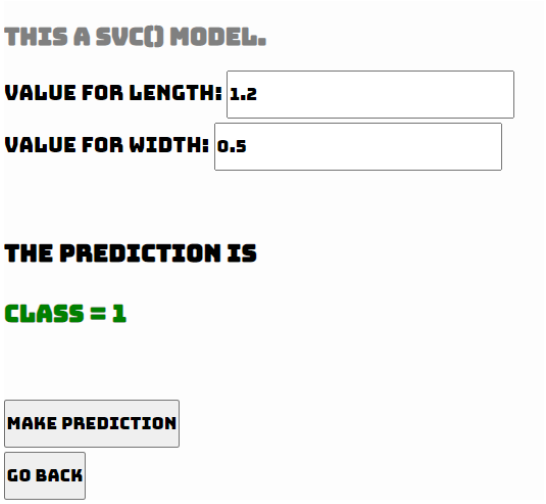


Ilustración 6 - Predicciones del modelo

5. Conclusiones

Este proyecto cumple con los objetivos propuestos y consigue acercarse a lo que sería una plataforma completamente funcional de generación de modelos sin necesidad de escribir líneas de código. Aun así, existen dos problemas clave que se deben solucionar para que el alcance de este proyecto sea mucho mayor.

En primer lugar, los datos deben estar limpios y para tener el mejor rendimiento deben estar preprocesados. En esta versión se ha hecho solo la limpieza básica de los NA para

garantizar que el algoritmo funcione. Como futuro trabajo se debería añadir una capa de preprocesado de datos automática que deje los datos preparados para los modelos.

En segundo lugar, para generar modelos con altas tasas de acierto, se necesita tener ciertos conocimientos del funcionamiento de los algoritmos. Es decir, que, aunque el usuario no necesite tener conocimientos técnicos, para obtener un mayor rendimiento del aplicativo es conveniente conocer los algoritmos. Se podría desarrollar una capa de selección de hyper-parametros para generar modelos más eficientes de forma automática.

6. Referencias

[1] BBVA. (11 de Noviembre de 2019). *Machine learning: What is it and how does it work?* Obtenido de https://www.bbva.com/en/machine-learning-what-is-it-and-how-does-it-work/

[2] Django. (s.f.). *Django documentation*. Obtenido de https://docs.djangoproject.com/en/4.1/

[3] Koehersen, W. (3 de Julio de 2018). *Automated Machine Learning Hyperparameter Tuning in Python*. Obtenido de https://towardsdatascience.com/automated-machine-learning-hyperparameter-tuning-in-python-dfda59b72f8a#8811

[4] Urban, T. (22 de Enero de 2015). *Wait but why*. Obtenido de The AI Revolution: The Road to Superintelligence: https://waitbutwhy.com/2015/01/artificial-intelligence-revolution-1.html

WEB APPLICATION FOR THE GENERATION OF MACHINE LEARNING MODELS AND MAKING PREDICTIONS ON ARBITRARY DATA

Author: Bicand Fernández, Daniel.
Supervisor: Morrás Ruiz-Falcó, Carlos.
Collaborating Entity: ICAI – Universidad Pontificia Comillas

ABSTRACT

During the completion of this final thesis project, a web application with the capacity of making predictions based on machine learning algorithms has been developed.

The application contains 9 different algorithms, 4 of them are regression algorithms and the rest are classification algorithms. Also, the application contains a module that takes care of saving the trained models, so any user can access them whenever they need without needing to re-train their model.

Moreover, the application has a basic data cleaning algorithm that deletes or overwrites unassigned values from the databases uploaded by users.

Keywords: Machine Learning, Software, WebApp, DataScience, Data Analytics

1. Introduction

Throughout history, there has been several industrial revolutions that have increased the productivity in humanity and improved significantly the living conditions in the earth. It is said that we are currently living the fourth industrial revolution, the Artificial Intelligence (AI) revolution.

One of the AI branches is Machine Learning. Machine Learning (ML) is a type of AI that gives computers the capacity to learn without being specifically programed to do so. This means that, given a set of arbitrary data, the computer can learn from them and make accurate predictions based on new unseen data.

The main goal of this project is to bring this technology closer to non-technical profiles. Nowadays, is hard to access and create ML algorithms without having the ability to code. This project will serve as a layer between the code and the users, making the access to ML models easier.

2. Project definition

The project design consists of building a web app that serves as an abstraction layer to generate predictive models. That is, embed the logic behind building and training a predictive model inside a web framework, so any potential user can build a model with just a couple of clicks.

For that, we need to build an environment where a user can download any database they want to use. Then, add to the environment a basic layer of data cleaning so there are no errors when creating the models. Finally, give the users access to the different ML algorithms and the ability to generate any model by setting the training and target variables and the model hyper-parameters.

Also, the users should be able to make predictions with their trained models and save their models to access them whenever they need.

3. Description of the webapp

This project has been developed using the Django framework. Django is a high-level Python web framework that enables rapid development of secure and maintainable websites. Django follows the model–template–views architectural pattern (MVT).

The MVT architecture is like the MVC one, used by other frameworks such as Spring Boot.

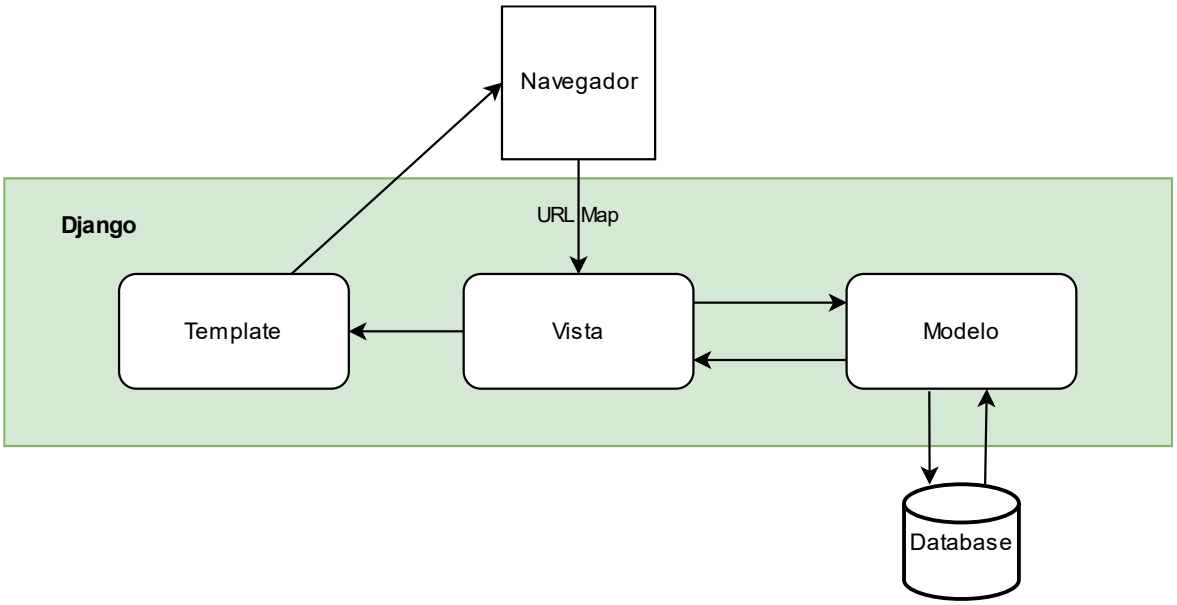


Figure 7 – MVT Architecture

Any Django project is divided in several applications, each one with its own specific functionality. This project contains 4 different applications:

1. SandboxApp
2. Databases
3. ML
4. Models

The first one is the main application. It´s functionality is to load the home template and redirect users to the rest of the applications.

The Databases application contains all the load, delete and cleaning logic for the loaded databases. From this app the users can select any of the loaded databases to train the predictive models.

The ML application is the project core. It contains all the logic in charge of creating and training the predictive models. From this application, the users can select any of the 9 available algorithms and generate a simple model. Also, there is a hyper-parameter tunning option for advance users.

Finally, the Models application allows users to save the trained models and make predictions with them.

Figure 2 shows the flowchart of a model generation.

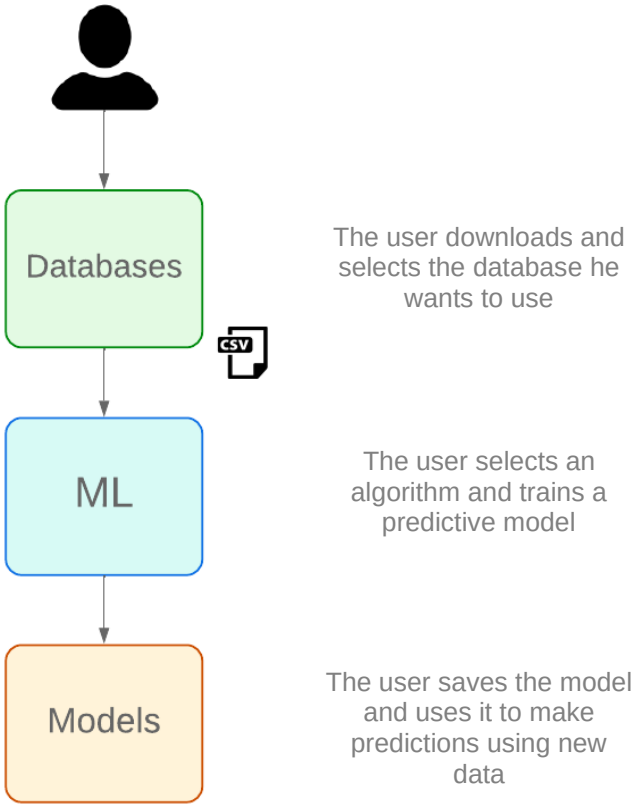


Figure 8 – Flowchart of the generation and use of a predictive model

4. Results

The developed project meets the objectives defined in section 2. The application can generate models and make predictions on any ordered .csv file without the need to write a single line of code.

In the following screenshots, you can see the process from when a user selects the data to create a model, until he agrees with its accuracy and saves the model to make predictions.

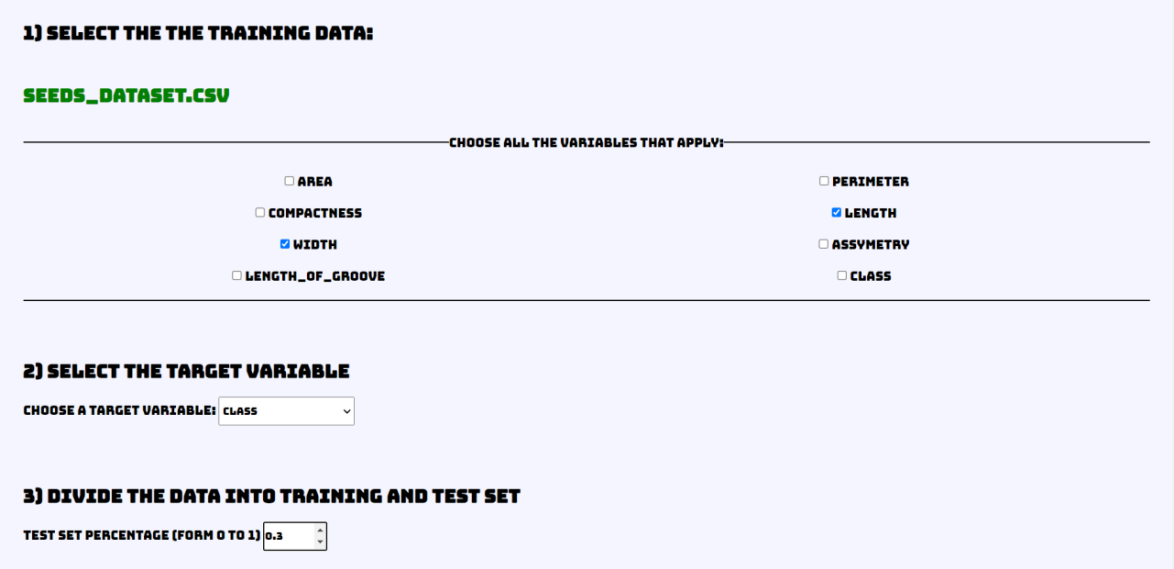


Figure 9 – Generating a SVM model

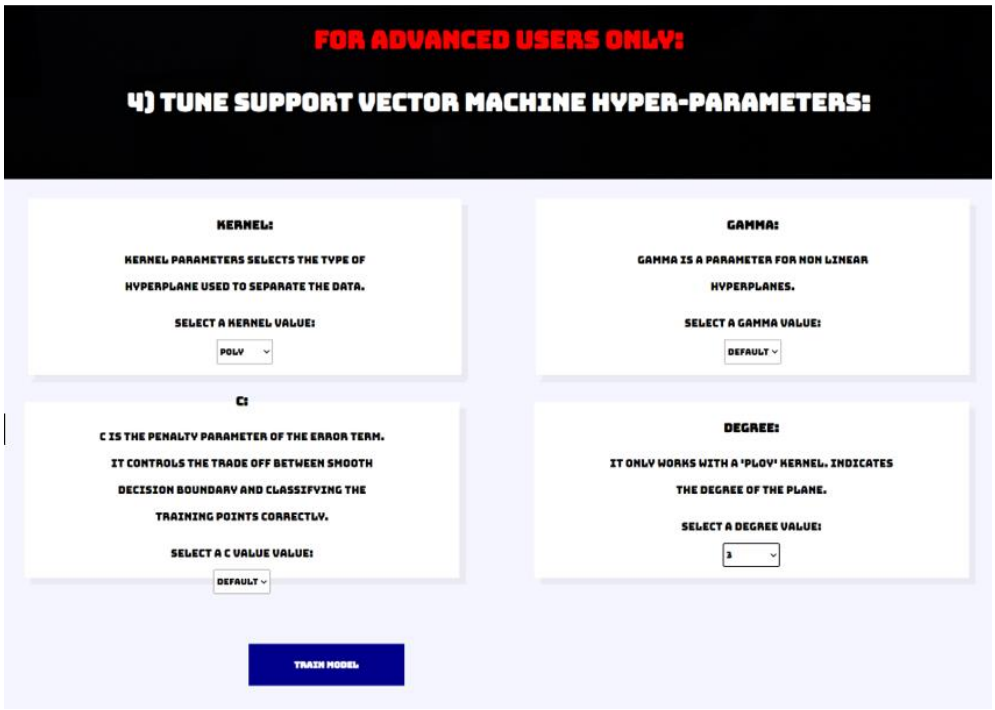


Figure 10 – Tunning SVM hyper-parameters

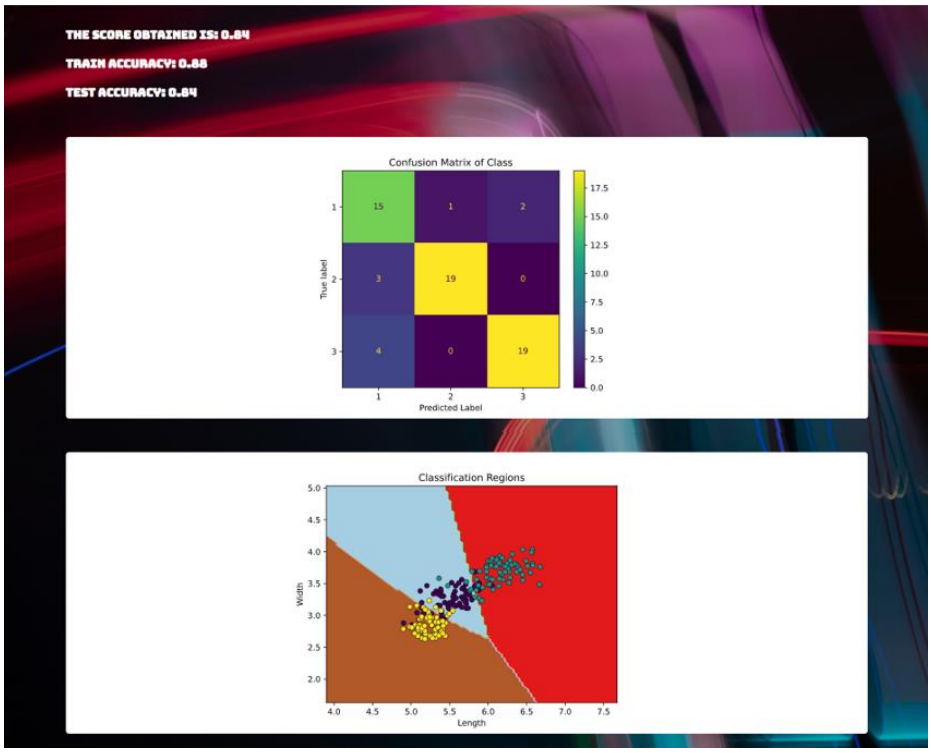


Figure 11 – Model output

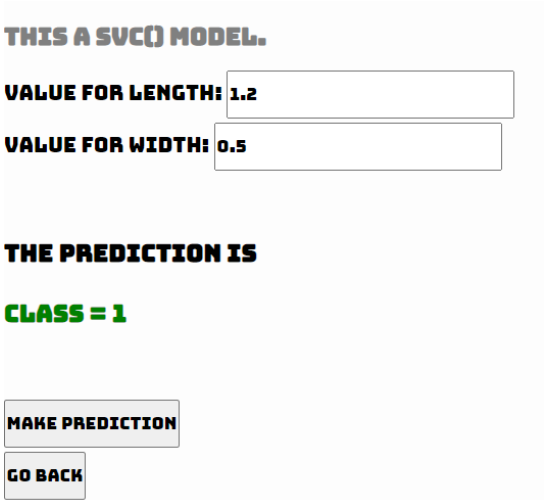


Figure 12 – Model predictions

5. Conclusions

This project manages to get close to what would be a fully functional model generation platform without the need to write lines of code. Still, there are two key issues that need to be addressed if the scope of this project is to be much larger.

Firstly, the data must be clean and for best performance it must be pre-processed. Only basic NA cleanup has been done in this release to ensure the algorithm works. As future work, an automatic data pre-processing layer could be implemented in the project to leave the data ready for the models.

Secondly, in order to generate models with high accuracy rates, some technical knowledge is needed. This is because the correct tunning of hyper-parameters is necessary to find optimal models. So, even if users don't need to have technical knowledge to use the application, if they don't know about ML, they won't extract as much value from the app. On the next iterations of this application, it would be interesting to add a new module capable of selecting the most interesting algorithms and automatically tune its hyper-parameters.

6. References

[1] BBVA. (11 de Noviembre de 2019). *Machine learning: What is it and how does it work?* Obtenido de https://www.bbva.com/en/machine-learning-what-is-it-and-how-does-it-work/

[2] Django. (s.f.). *Django documentation*. Obtenido de https://docs.djangoproject.com/en/4.1/

[3] Koehersen, W. (3 de Julio de 2018). *Automated Machine Learning Hyperparameter Tuning in Python*. Obtenido de https://towardsdatascience.com/automated-machine-learning-hyperparameter-tuning-in-python-dfda59b72f8a#8811

[4] Urban, T. (22 de Enero de 2015). *Wait but why*. Obtenido de The AI Revolution: The Road to Superintelligence: https://waitbutwhy.com/2015/01/artificial-intelligence-revolution-1.html

Índice de la memoria

Índice de figuras

Capítulo 1. INTRODUCCIÓN

Desde el inicio de los tiempos, la humanidad avanza y progresa gracias al descubrimiento de nuevas tecnologías y de su implementación en la vida real. Esta implementación de las nuevas tecnologías aumenta la productividad en la sociedad y reduce la cantidad de recursos que se tienen que dedicar a las tareas que han sido optimizadas. El exceso de recursos se destina a diseñar nuevas tecnologías que aumenten la productividad en otros sectores o incluso en el mismo, produciéndose así un ciclo de mejoras e innovaciones. Desde la primera revolución industrial, se ha repetido este comportamiento cíclico en varias ocasiones, pasando por las distintas revoluciones y llegando hasta la actualidad. No cabe duda de que desde hace unos años estamos viviendo una clara revolución centrada alrededor de la inteligencia artificial y de los algoritmos inteligentes. Las tareas que creíamos que los ordenadores nunca podrían realizar, ahora las realizan mejor que los humanos. Todas estas nuevas tecnologías de predicción cada vez son más usadas por las grandes corporaciones y es casi un requisito indispensable implementarlas para mantenerse competitivo en los mercados. Con el auge de esta revolución se han abierto una gran cantidad de oportunidades. Por estas razones, este proyecto estará centrado en esta nueva revolución, la revolución de la inteligencia artificial o cuarta revolución industrial.

Antes de introducir el enfoque del proyecto, vamos a aclarar en que tecnologías se centrará el mismo ya que el mundo de la inteligencia artificial (IA) es inmenso y abarca mucho. Para empezar, dentro de todas las categorías que conforman el sector de la Inteligencia Artificial, nuestro proyecto se centrará en el Machine Learning (Aprendizaje Automático en español). El Machine Learning (ML) es un tipo de IA que proporciona a las computadoras capacidad de aprender sin ser explícitamente programadas para ello. Es decir, son algoritmos que aprenden en base a los datos que les ofrezcas. Una vez entrenas un modelo de ML, este puede realizar predicciones sobre nuevos datos que nunca haya visto. De esta manera, cualquiera que haya entrenado un modelo robusto puede anticiparse o sacar conclusiones de nuevos datos. Como se puede ver, esta tecnología resulta útil para

cualquier usuario que disponga de datos y quiera aprender de ellos. Ahora bien, dentro de la categoría de Machine Learning existen subcategorías. Los algoritmos que se van a implementar en el desarrollo de este proyecto serán algoritmos de clasificación y regresión. Los algoritmos de regresión estudian la relación entre un cierto número de características y una variable objetivo continua. Algunos ejemplos son; la predicción del precio del café, una estimación de ventas de un producto o la predicción de la presión sanguínea de un paciente. Los algoritmos de clasificación, en cambio, tienen la misión de categorizar variables dadas ciertas características. Por ejemplo, dadas la longitud y anchura de un petalo, los algoritmos de clasificación tratarán de predecir qué tipo de flor es.

Una vez entendido esto, nos podemos centrar en el enfoque y la motivación del proyecto. Pues bien, hoy en día, para implementar las tecnologías que he mencionado antes se necesitan ciertos conocimientos de programación. Existen muy pocas herramientas que te ofrezcan un acceso User Friendly (fácil de usar) a estos modelos predictivos. Es así como nace la idea de este proyecto. Sabiendo que el mundo va dirigido hacia la era de los datos y de las IAs, tarde o temprano tendrá que producirse un acercamiento de estas tecnologías a los perfiles menos técnicos. Este proyecto consiste en eso mismo, proponer una herramienta con un grado alto de abstracción, que acerque estos modelos predictivos a usuarios que no tengan conocimientos técnicos de programación. La motivación del proyecto es encapsular dentro de una aplicación web, una variedad de algoritmos de predicción donde cualquier usuario pueda entrenar estos algoritmos con sus datos y luego sacar conclusiones de ellos.

El logotipo de la aplicación web es el siguiente:

Ilustración 13: Logotipo de la Aplicación

A continuación, en el desarrollo de esta memoria, primero se presentarán las tecnologías usadas (Capítulo 2. Descripción de las Tecnologías). Luego, se extenderá lo introducido en este capítulo comentando el estado y las características del mercado en este sector (Capítulo 3. Estado de la Cuestión). Se definirá como se ha trabajado, cuáles han sido los objetivos y la metodología seguida (Capítulo 4. Definición del Trabajo) y finalmente, se comentará todo lo relativo al sistema desarrollado desde un punto de vista más técnico (Capítulo 5. Desarrollo del Proyecto). Además, se hará un análisis de los resultados obtenidos y se comentaran las conclusiones obtenidas y los posibles trabajos futuros (Capítulo 6. Trabajos Futuros).

Capítulo 2. DESCRIPCIÓN DE LAS TECNOLOGÍAS

Este proyecto se puede dividir en dos grandes bloques. El primer bloque consiste en todo lo relativo a la parte de los modelos de predicción y algoritmos de aprendizaje automático empleados. Este bloque es la parte de negocio dentro del software desarrollado. El segundo, es todo lo relativo al desarrollo software de la aplicación web y su arquitectura. Ambos bloques están interconectados y para tomar la decisión de que tecnologías usar, lo principal ha sido entender cuáles son los puntos de unión de ambos bloques y de qué manera sería más sencillo relacionarlos. Por eso, primero hablaré de las tecnologías utilizadas para desarrollar los modelos predictivos y luego, del software que acoge estas funcionalidades y la muestra al cliente a través de un servidor web.

2.1 BLOQUE I: MODELOS PREDICTIVOS

En este bloque se explica toda la parte de negocio dentro de la aplicación web. Esta parte contiene la creación de los modelos predictivos y todo lo relacionado.

2.1.1 LENGUAJE DE PROGRAMACIÓN UTILIZADO

Hoy en día existen una gran variedad de lenguajes de programación que son especialmente buenos para desarrollar modelos predictivos e implementar algoritmos de aprendizaje automático. De todos los existentes, con la idea de trasladar la implementación de estos algoritmos a una aplicación web, decidí utilizar Python. Esta decisión está fundamentada en varios puntos:

- Python contiene una gran cantidad de librerías como sklearn (hablaremos después de ella) que facilita enormemente la implementación de estos modelos.
- Existen otros lenguajes de programación para la generación de modelos predictivos, pero su uso cada vez es más reducido. Además, es mucho más complicado integrarlos en aplicaciones web.

- El aprendizaje de un lenguaje como Python es mucho más rápido que el de otros lenguajes debido a su simpleza y a su parecido con otros lenguajes que estudiamos en ICAI.
- En relación con el segundo bloque de desarrollo, la integración de los modelos predictivos a una aplicación web es mucho más sencilla si se utiliza Python.

Todos estos puntos fueron lo que influyeron en la decisión de elaborar el primer bloque de desarrollo utilizando Python.

2.1.2 LIBRERÍAS UTILIZADAS

Uno de los puntos positivos que hemos mencionado de utilizar Python son sus librerías. En el desarrollo de este proyecto se ha hecho uso de distintas librerías. Entre ellas algunas muy comunes que no explicaremos en detalle:

- **Pandas**: Una librería especializada en el manejo y análisis de estructuras de datos. Además, permite leer con facilidad datos de ficheros de Excel, en formato .csv y de bases de datos SQL.

- **Matplotlib**: Librería especializada en la creación de gráficos en dos dimensiones. Esta librería ha sido realmente útil a la hora de plotear los resultados de los distintos tipos de modelos utilizados en la aplicación web.

- **Numpy**: Otra librería muy utilizada en Python especializada en el cálculo numérico y en el análisis de datos.

- **Scikit-learn**: También conocida como sklearn, es la librería más útil para implementar modelos de aprendizaje automático en Python. Esta librería utiliza una combinación de otras librerías (Numpy, Pandas, SciPy, Matplotlib, Ipython y SymPy). Dentro de esta librería existen una gran variedad de modelos de aprendizaje automático ya creados. Estos modelos pueden ser ajustados a través de

distintos parámetros, y estos parámetros son los que se utilizan en este proyecto para mejorar las predicciones.

- **Joblib:** Es una librería que se utiliza para el almacenaje de modelos predictivos. Es capaz de almacenar un objeto modelo, que ya ha sido entrenado, en un fichero .sav. Esta librería es de gran utilidad en el proyecto ya que permite a los usuarios almacenar los modelos que entrenen y acceder a ellos cuando lo necesiten.

2.1.3 MODELOS ELEGIDOS

Dentro del mundo del aprendizaje automático existen una gran variedad de algoritmos. Estos algoritmos se pueden dividir en distintas categorías. En cuanto al tipo de aprendizaje, los algoritmos utilizados en la aplicación web son todos del tipo de aprendizaje supervisado. Los algoritmos de aprendizaje supervisado se caracterizan por que usan datos etiquetados. Estos algoritmos son los más comunes y los más sencillos de implementar, sobre todo para el caso de uso que se propone en este proyecto. Otra categoría es el tipo de problema que se resuelve. Existen problemas de regresión, donde se predicen variables numéricas dadas otras variables numéricas. También, problemas de clasificación, donde se busca dados unos inputs iniciales, clasificar en distintas categorías.

Además de esto, cada modelo tiene una selección de hyper-parametros característicos del propio modelo. Los hyper-parametros no son más que pequeñas alteraciones y restricciones que puedes poner al modelo, normalmente con el objetivo de que sea más preciso, aunque también se pueden seleccionar con el fin de que el tiempo de entrenamiento sea menor.

A continuación, expondremos los distintos modelos seleccionados para el proyecto y comentaremos los hyper-parametros que se pueden tunear y su funcionalidad.

2.1.3.1 Modelos de Regresión

- **Regresión lineal**: Es un modelo estadístico, no de aprendizaje automático. Es el modelo más simple que existe y se utiliza para aproximar la relación de dependencia entre una variable dependiente y otras independientes. La fórmula matemática que define este modelo es la siguiente:

$$Y_i = f(X_i, \beta) + e_i$$

Ecuación 1: Regresión Lineal

  - Y_i = Variable Dependiente
  - f = función
  - X_i = variables independientes
  - β = parámetros a estimar
  - e_i = error

  El objetivo de este modelo es estimar los parámetros β para que la combinación de las variables independientes con esos parámetros dentro de la función f predigan la variable dependiente Y.

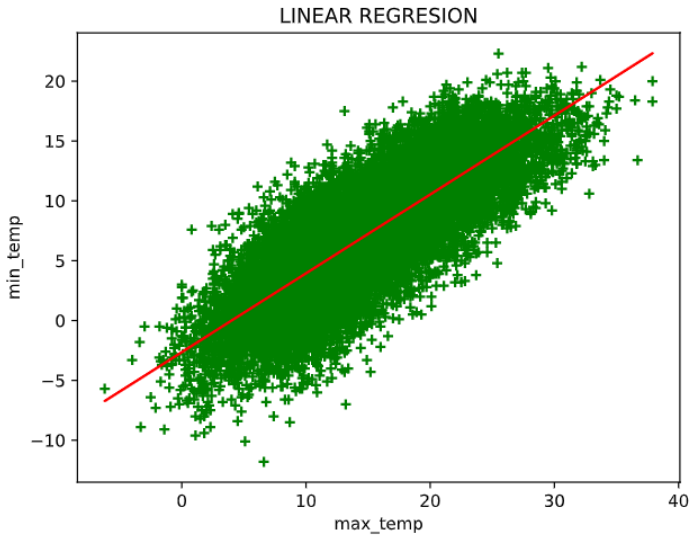


Ilustración 14 - Regresión lineal

- **Árbol de regresión**: Este modelo se basa en árboles de decisión. La diferencia con los árboles de clasificación es que en este caso la variable a predecir es un número real (por ejemplo, el precio del café o el número de visualizaciones de un video de YouTube).

  Este algoritmo funciona especialmente bien en funciones escalonadas. Cada nodo del árbol de decisión se traduce en un cambio de dirección en la recta de regresión.

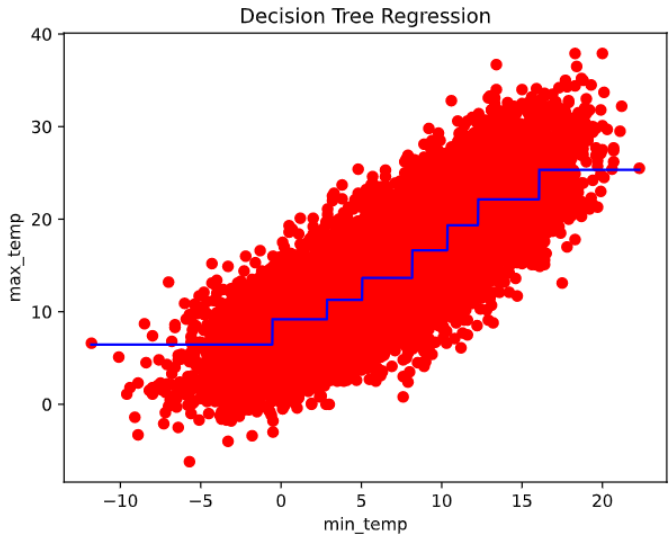


Ilustración 15 - Árbol de regresión

Los hyper-parámetros disponibles para la optimización de este modelo en la aplicación web son los siguientes:

- **Criterio de División:** Es la función que mide la calidad de la división en cada rama del árbol. Las funciones disponibles son; mediante el error cuadrático medio “squared_error” (por defecto), el error absoluto “absolute_error”, el error cuadrático medio con la mejora de Friedman “friedman_mse” y “poisson” que decide las divisiones de las ramas basándose en la reducción en la desviación de Poisson.

- **Profundidad Máxima del Árbol:** Indica el nivel de profundidad máximo para el árbol. La profundidad máxima de un árbol es la cantidad máxima de divisiones que se hacen hasta llegar a un nodo final. El parámetro por defecto no viene inicializado, por lo que, si no se reconfigura este parámetro, se realizaran divisiones en las ramas hasta que se llegue a menos del mínimo de muestras por división.

- **Mínimo de Muestras por División:** Mínimo de muestras requeridas para dividir los nodos internos del árbol. Este parámetro viene por defecto inicializado en 2, es decir, que, si la profundidad está marcada por defecto, el modelo hará subdivisiones hasta que en cada división como mucho queden 2 muestras de lado.

- **Mínimo de Muestras en nodo Final:** Igual que el parámetro anterior solo que para los nodos hoja (nodos finales). Esta inicializada por defecto a 1. Este parámetro consigue un efecto de suavizar el modelo, sobre todo si se usa en regresión.

- **Gradient Boosting Regression**: En español se utiliza el termino de potenciación del gradiente. Este es un modelo de aprendizaje automático que consiste en mejorar la predicción global de un modelo utilizando una gran cantidad de modelos con predicciones débiles. La combinación de muchas predicciones débiles (con una tasa de aciertos baja, pero superior al 50%) genera una predicción más robusta.

  Normalmente, el modelo débil que se utiliza es el de árboles de decisión. Al igual que los árboles de regresión, este modelo tiene varios parámetros que se pueden usar para tratar de mejorar su puntuación. Como el modelo consiste en la combinación de muchos arboles de regresión, algunos de los hiperparámetros a ajustar son los mismos que los explicados previamente así que no entraremos en detalle.

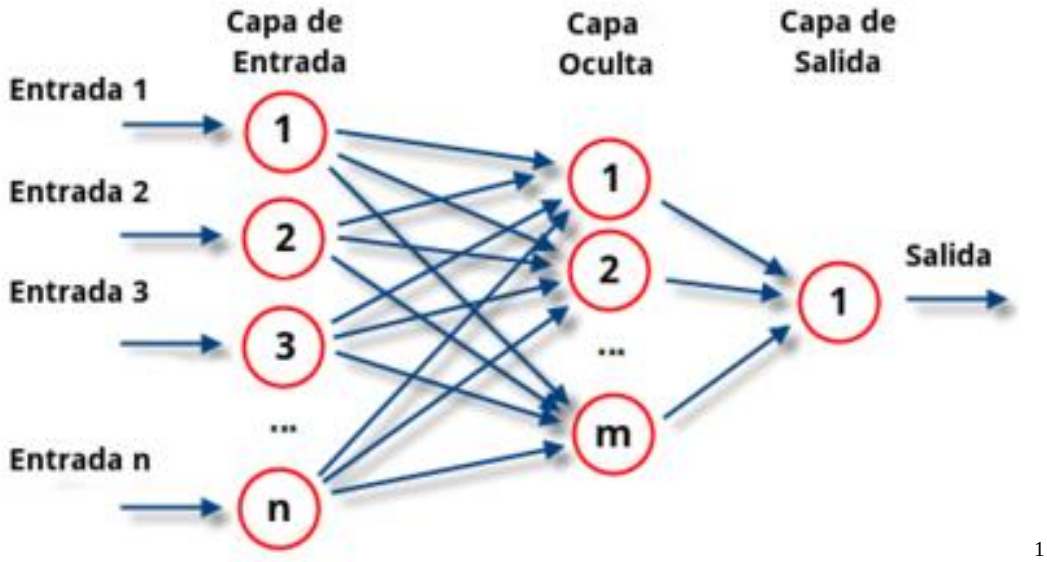
Aun así, hay dos hyper-parámetros que son exclusivos:

- **Ratio de Aprendizaje:** La contribución que hace cada árbol individual al modelo general. Existe un tradeoff entre este ratio de aprendizaje y la cantidad de estimadores a usar. Este parámetro viene ajustado a 0.1 por defecto.

- **Numero de Estimadores:** Indica el número de veces que se iterará cada árbol individual, es decir, la cantidad de árboles que se usaran en el modelo de Boosting. Este algoritmo de aprendizaje automático es bastante resiliente al overfiting (sobre ajuste del modelo) así que cuanto mayor sea el número de estimadores mejor será el modelo, pero también más tardará en converger. El parámetro está ajustado a 100 por defecto.

- **Red Neuronal de Regresión:** Las redes neuronales son algoritmos de aprendizaje profundo que simulan el comportamiento de las neuronas dentro de un cerebro humano.

  Las redes de regresión se caracterizan por tener tan solo un nodo de salida. Esto ocurre porque al tratarse de un problema de regresión y no de clasificación, la variable a predecir es un numero entero y no una categoría. Si fuera una categoría, cada una de ellas sería un nodo de salida, y una vez dado un input, cada salida marcaría una probabilidad de que ese input pertenezca a su categoría.



[1]

Ilustración 16 - Arquitectura Red Neuronal de Regresión

Existen distintos hyper-parametros que se pueden tunear utilizando este algoritmo.

- **Numero de capas ocultas:** Dentro de las redes neuronales hay varias capas de neuronas. Siempre se tiene la capa de input, que es donde están las neuronas por donde entran los datos, y la capa de output, que son las neuronas (en este caso solo una al tratarse de una regresión) por donde sale la predicción. Entre medias existe una capa que se llama “hidden layer” en español capa oculta. Este parámetro indica la cantidad de capas ocultas que queremos que tenga nuestra red neuronal. Por defecto el modelo tiene una hidden layer con 100 neuronas

- **Numero de neuronas en las capas ocultas:** Una vez hemos seleccionado cuantas hidden layers va a tener nuestra red neuronal, debemos de

[1] Red Neuronal Artificial de tipo perceptrón simple con *n* neuronas de entrada, *m* neuronas en su capa oculta y una neurona de salida. https://es.wikipedia.org/wiki/Perceptr%C3%B3n_multicapa

seleccionar cuantos nodos tendrán esas capas. Cuantas más capas y más nodos tengan nuestra red neuronal, más tiempo tardará en entrenarse.

- **Solver:** Cuando se entrena una red neuronal, cada una de las neuronas recibe un peso. Ese peso puede ser calculado de distintas maneras y el parámetro solver indica la manera en la que queremos que se calcule ese peso. El cálculo por defecto es un con optimizador estocástico basado en el gradiente "Adam". También, existe un método que consiste en el descenso del gradiente estocástico "sgd". Por último, el optimizador de la familia de los métodos quasi-Newtonianos, "lbfgs". Este método es mayormente utilizado para cuando se tienen pocos datos.

- **Número máximo de Iteraciones:** Cuantas iteraciones se producen hasta que el modelo deja de entrenarse. Por defecto viene ajustado a 200.

- **Alpha:** Es un parámetro que sirve para la regulación de la red neuronal. El fin último es combatir el sobre ajuste de la red a los datos de entrenamiento. Esto lo consigue limitando el peso máximo de las distintas neuronas dentro de la red

2.1.3.2 Modelos de Clasificación

Los modelos de clasificación buscan dados ciertos inputs, identificar a que categoría pertenecen esos inputs. Una característica de estos modelos, que también tienen algunos de los modelos utilizados en el apartado anterior, es la división de los datos. Por un lado, se selecciona aleatoriamente un set de datos para entrenar el modelo y, por otro lado, un set para testear su rendimiento.

Una vez se entrena el modelo con los datos de entrenamiento, que suelen ser entre un 80 y 70 por ciento de los datos totales, con el porcentaje restante se comprueba la validez del modelo. Comparar la tasa de aciertos de los datos de testeo con la de los datos de

entrenamiento da mucha información. Si con los datos de testeo se acierta muy poco, pero con los de entrenamiento se acierta habitualmente, entonces podemos decir que el modelo se ha sobre-ajustado (overfitting) a los datos de entrenamiento, y no ha aprendido a predecir en general, si no que tan solo ha aprendido los datos de entrenamiento. Hablaremos más en detalle de estos análisis en el apartado 2.1.4 Gráficas y outputs.

Los cinco algoritmos de clasificación seleccionados son los siguientes:

- **Árbol de clasificación:** Los árboles de clasificación son parecidos a los de regresión. El objetivo, en vez de predecir un número real es predecir una categoría. Por lo tanto, las divisiones y las decisiones que se toman en cada nodo del árbol van enfocadas a dividir el espacio de muestreo, escogiendo los cortes basándose en la cantidad de datos pertenecientes a una categoría que hay en cada región de partición.

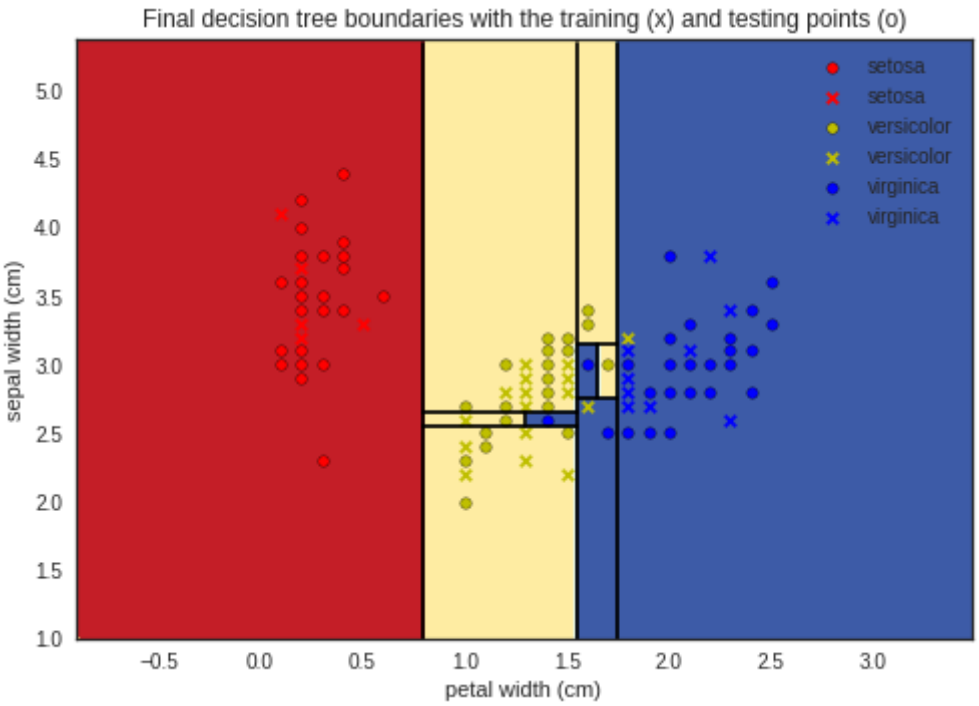


Ilustración 17 - Cortes en un Árbol de Clasificación

Los parámetros que se pueden modificar dentro de este modelo son los mismos que para el árbol de regresión salvo que el criterio para definir los cortes es distinto ya que el objetivo de este algoritmo también es distinto.

- **Criterio de División:** Es la función que mide la calidad de la división en cada rama del árbol. Este cálculo se puede realizar mediante índice de Gini; "gini" que mide el grado de pureza de los nodos, la entropía "entropy" que es una medida que se utiliza para cuantificar el desorden en cada división, y por último con la perdida logarítmica "log_loss".

$$\boldsymbol{E(S) = \sum_{i=1}^{c} -p_i \log_2 p_i}$$

Ecuación 2 – Entropía

$$\boldsymbol{Gini = 1 - \sum_{i=1}^{C} (p_i)^2}$$

Ecuación 3 - Coeficiente de Gini

$$H_p(q) = -\frac{1}{N} \sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i))$$

Ecuación 4 - Log Loss

- **Gradient Boosting Classification:** Lo mismo ocurre en este caso, los parámetros que se pueden modificar son los mismos que en el caso del Gradient Boosting Regressor. La única diferencia es que ahora, en vez de utilizar arboles de regresión como modelos de predicción débil, utiliza arboles de clasificación.

- **SVM (Support Vector Machines):** Este algoritmo consiste en un modelo matemático de generación de hiperplanos dados unos vectores soporte. Básicamente, dadas unas variables de entrada, el modelo busca un hiperplano que quede a la misma distancia de algunos vectores soporte (datos cerca de la frontera de decisión).

  Existen varios parámetros que son muy relevantes a la hora de crear estos modelos ya que muchas veces, las fronteras de decisión no pueden ser tan solo lineales y se necesitan otras divisiones más complejas.

  - **Kernel**: El kernel hace referencia a las funciones de núcleo. Su función es mapear los datos recibidos a dimensiones superiores para facilitar el cálculo de las superficies de decisión. A simple vista puede parecer que esto es algo costoso computacionalmente hablando, pero no lo es.

    Existen distintos tipos de kernels, dentro de este modelo usamos el kernel lineal "lineal", el kernel polinómico "poly" y el kernel rbf "rbf", que mapea un espacio de entrada en un espacio dimensional infinito (esta es la opción por defecto).

    La diferencia en el kernel utilizado es enorme y se puede apreciar muy claramente cuando se dibujan las superficies de decisión.

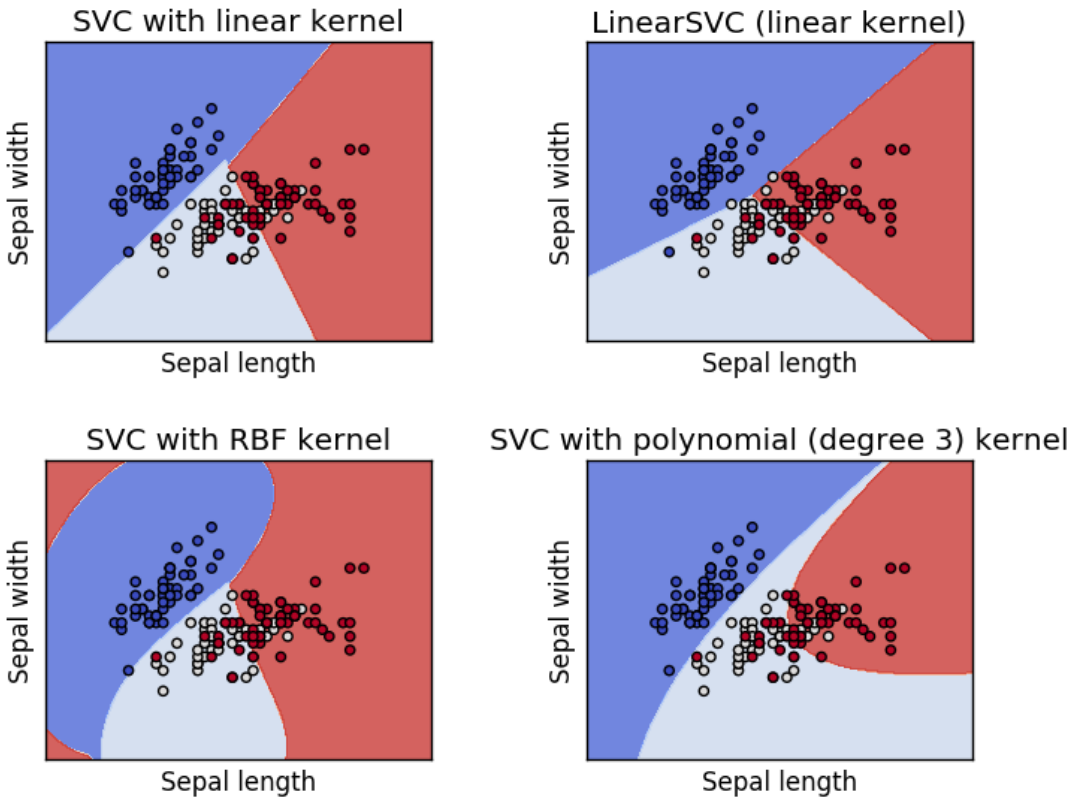


Ilustración 18 - SVC según kernel utilizado

- **C**: Es el parámetro que penaliza el error cometido. Es decir, cuanto más alto es el valor de este parámetro, más se ajusta el modelo a los datos de entrenamiento por que más se penaliza el error de ajuste. Esto puede causar overfitting. En cambio, si el valor del parámetro C es muy bajo, el modelo será muy generalista y tendrá un ratio de acierto más bajo. El valor por defecto de este parámetro es 1.

- **Grado:** Este parámetro indica el grado de del hiperplano para crear la superficie de división. Como se puede intuir, solamente se pone en práctica cuando el kernel que

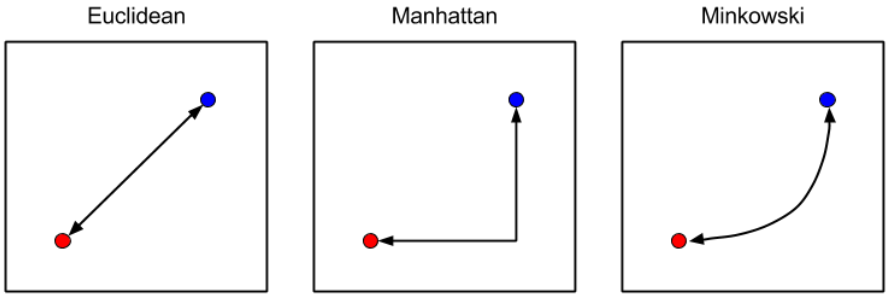
se usa es polinómico ya que si es lineal el grado es 1 y si se usa el kernel "rbf" el grado es infinito.

- **Gamma:** Este parámetro funciona para hiperplanos no lineales, es decir, utilizando kernel polinómicos o rbf. Es un coeficiente que define la función utilizada por el kernel. Cuanto más alto sea el valor, más se ajustará la superficie de decisión a los datos de entrenamiento y más riesgo de overfitting habrá.

- **KNN (K Nearest Neighbors)**: El funcionamiento de este modelo de predicción es mucho más intuitivo que el anterior. Básicamente, dados un set de datos de entrenamiento, este algoritmo recorre cada uno de los datos dentro del set en busca de las categorías de los datos más cercanos. Es decir, que busca cual es la categoría más común de entre los datos vecinos. Haciendo eso con cada uno de los datos acaba construyendo una región de decisión que le sirve para clasificar los datos que se le envíen para predecir.

  Una vez entendido el funcionamiento, surgen varias dudas como cuantos vecinos se comprueban y como se mide la distancia entre los vecinos. Existen valores por defecto, pero ambas variables se pueden tunear gracias a los hyper-parametros.

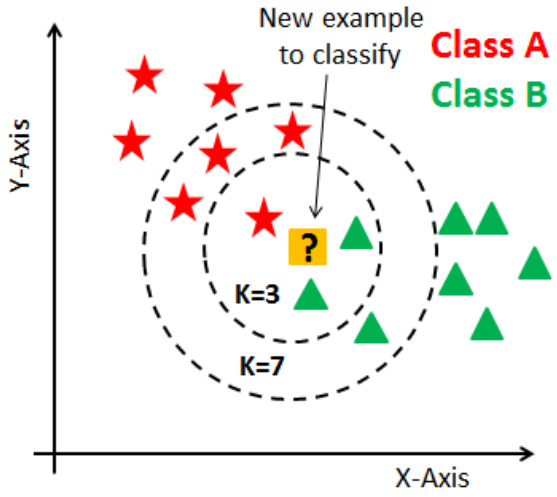
  - **Distancia:** Este parámetro indica como se calcula la distancia entre el dato seleccionado y sus vecinos. Existen varias formas de hacerlo, las más comunes son a través de la distancia Euclidiana y la distancia Manhattan. También, se incluye la distancia minkowski.



[2]

Ilustración 19 - Tipos de distancia KNN

- **K:** El valor de K es el que indica la cantidad de vecinos que se comprobará al correr este algoritmo. Cuando más alto sea K más vecinos se comprobarán y mejor será el modelo, pero también más tardará en converger la función. Lo que se busca es un trade-off entre calidad de modelo y tiempo de conversión. Por defecto, el valor de K viene definido a 5, y el límite es 20.

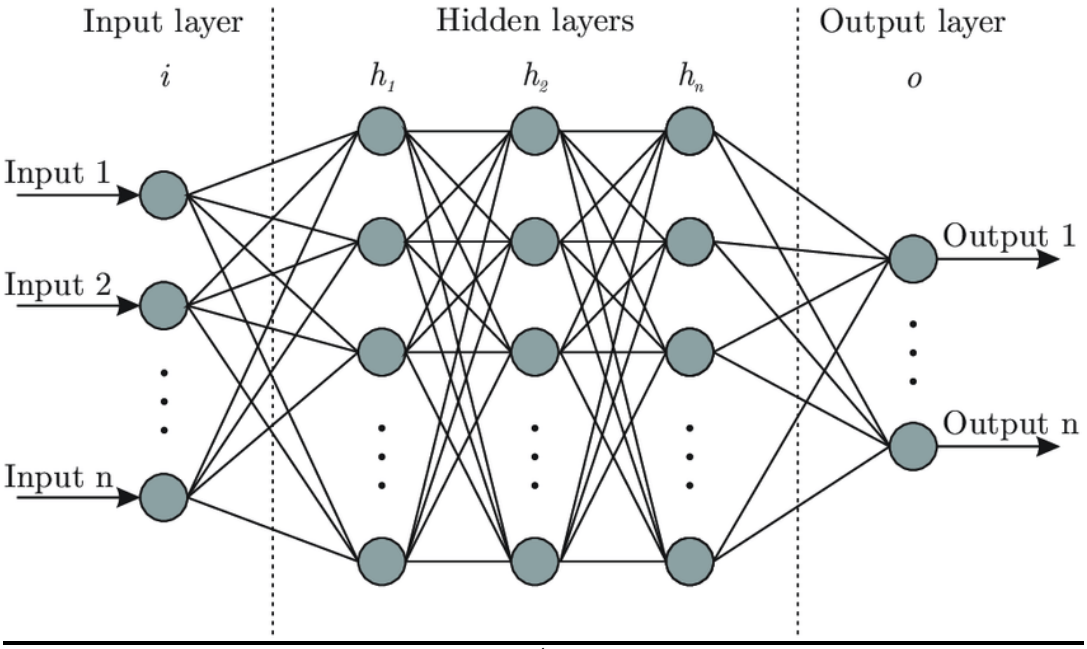


[3]

Ilustración 20 - Descripción grafica del K-valor

[2] Referencia de la imagen: https://es.wikipedia.org/wiki/K_vecinos_m%C3%A1s_pr%C3%B3ximos
[3] Referencia de la imagen: https://towardsdatascience.com/how-to-find-the-optimal-value-of-k-in-knn-35d936e554eb

- **Red Neuronal de Clasificación:** La regresión neuronal de clasificación es muy parecida a la que hemos explicado en el apartado de algoritmos de regresión. Como hemos comentado, en vez de tener un único nodo de salida, ahora tiene n nodos, siendo n la cantidad de categorías dentro de la variable a predecir.



[4]

Ilustración 21 - Arquitectura de una Red Neuronal de Clasificación

Estas redes aprenden gracias a un algoritmo de backpropagation (propagación hacia atrás). Al comienzo de la iteración, se meten los inputs por la primera capa y estos se propagan hasta el final multiplicándose por los distintos pesos de las neuronas. Cuando llega a la capa de salida, se compara el valor predicho con el real y se calcula el error. Luego, ese error se pasa por la red para que los pesos se reajusten y corrijan el error. Finalmente, pasados muchos datos, las redes consiguen aprender.

[4]Referencia de Ilustración 21 : ResearchGate – Artificial neural network architecture (ANN i-h 1-h 2-h n-o)

2.1.4 GRÁFICAS Y OUTPUTS

Finalmente, una vez el modelo es creado y entrenado, se tiene que evaluar su rendimiento. Para esto, existen algunos indicadores que muestran la puntuación del modelo y el ratio de acierto. También existen unas graficas que ayudan a entender el comportamiento del modelo y a elegir que parámetros utilizar.

- **Score:** El score es la puntuación del modelo, esta se da sobre 1 siendo 1 la tasa de aciertos exacta. Para calcular el score lo que se hace es, una vez entrenado el modelo, se predicen los datos de testeo. Según como difieran las predicciones de la realidad el score será más alto o bajo. No existe un score a partir del cual podemos decir que el modelo es bueno ya que depende mucho de los datos, pero cuanto más alto sea, más acertadas serán nuestras predicciones.

- **Importancia de las variables:** Otra de las cosas que se miden en la aplicación es la importancia de las variables. Esto es, como de relevante es una variable a la hora de predecir la variable objetivo. El cálculo que se realiza es haciendo una distribución de pesos, que en total suman 1, a todas las variables dentro del modelo. De esta forma, el usuario puede entender que variables tienen más peso e influencia en el resultado. También puede considerar eliminar del modelo las menos influyentes.

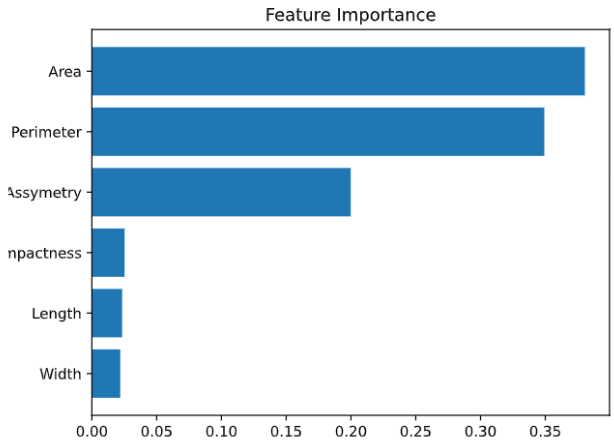


Ilustración 22 - Importancia de las variables de entrenamiento

- **Matriz de confusión:** La matriz de confusión es una gráfica muy útil que se usa en problemas de clasificación. Esta matriz está organizada de la siguiente forma. Por un lado, tenemos la realidad de los tests y por el otro lado tenemos los valores predichos. Si tuviéramos un modelo con un score de 1, todos los valores de la realidad coincidirían con la casilla de los valores predichos. En cambio, si nuestro modelo falla, los resultados de una categoría serán clasificados en otra por los tests.

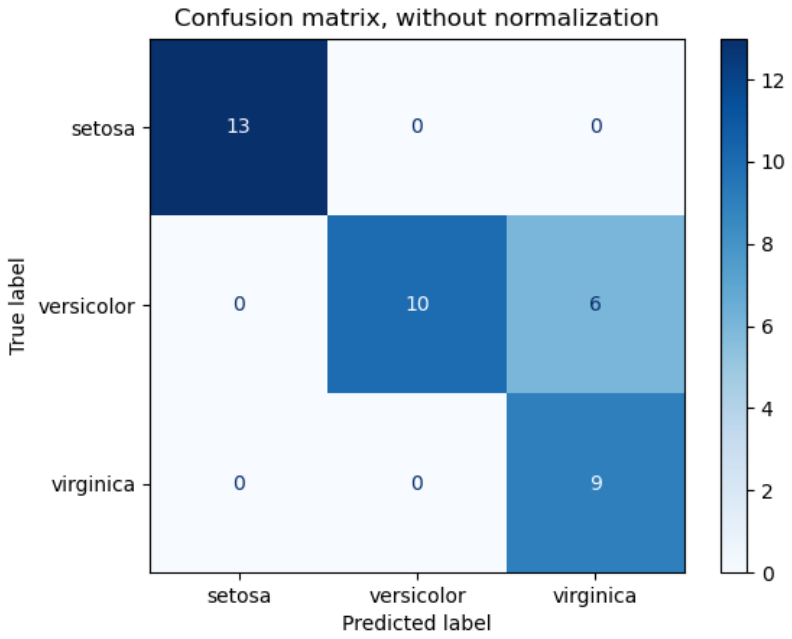


Ilustración 23 - Matriz de Confusión

En la ilustración 15, podemos ver como 6 plantas que son de la categoría versicolor han sido clasificadas como virginica. El resto de las plantas han sido clasificadas correctamente.

Es importante calcular los aciertos con el set de testeo y no con el de entrenamiento. De esta manera comprobamos el funcionamiento real del modelo ya que si el modelo está sobre-entrenado, las predicciones no serían realistas.

- **Regiones de clasificación:** Esta grafica es bastante simple de entender, simplemente divide el espacio en distintas regiones donde cada una representa una de las categorías dentro de la variable a decidir. Estas regiones solo se pueden dibujar cuando se seleccionan 2 variables en el modelo ya que es una gráfica en 2 dimensiones. También existen librerías que pueden graficar estás regiones en espacios tridimensionales.

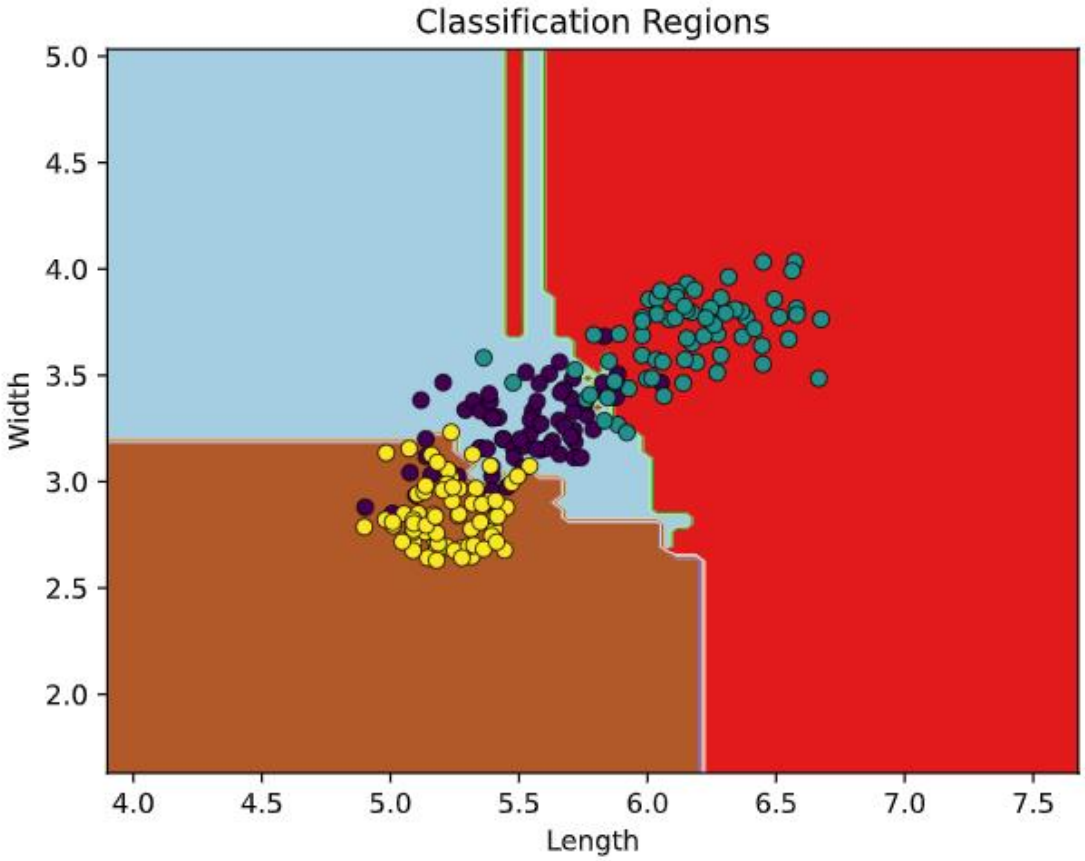


Ilustración 24 - Regiones de clasificación usando KNN

- **Curvas ROC y AUC:** La curva ROC es una curva que se utiliza para medir la eficacia de las clasificaciones realizadas. Esta grafica va muy de la mano de la matriz de confusión ya que indica las diferentes ratios de aciertos. El AUC no es más que el área debajo de la curva ROC. Cuanta más área haya, más aciertos hace el modelo y por tanto más precisas son las predicciones.

Ilustración 25 - Curvas ROC [5]

- **Curva de perdidas:** En una red neuronal, la función de perdidas indica la cantidad de error que se ha cometido en la predicción. El objetivo es reducir al máximo esta función de perdidas. Es por ello, que, en cada iteración, al hacer la propagación hacia atrás del error, los pesos de las neuronas se reajustan y el error va gradualmente disminuyendo.

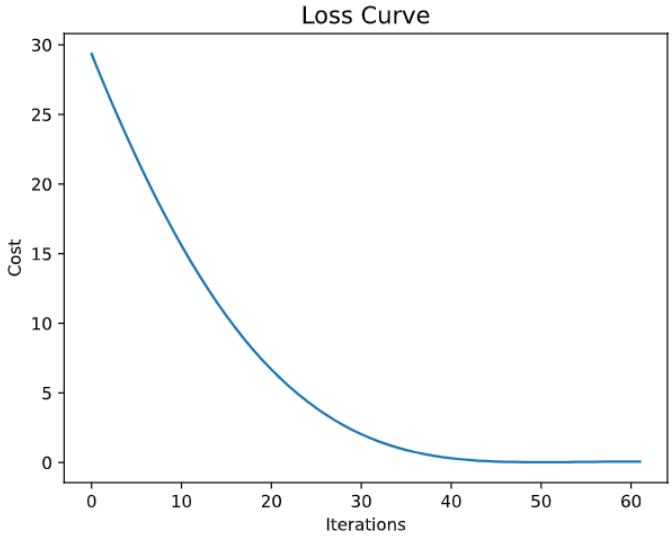


Ilustración 26 - Curva de perdidas

[5] Referencia Ilustración 9 - Wikimedia Commons File: Roc curve.svg

2.2 BLOQUE II: APLICACIÓN WEB

2.2.1 DJANGO:

A la hora de desarrollar una aplicación web, se necesita seleccionar un framework de trabajo. Los frameworks no son más que herramientas de desarrollo que conjuntan distintos módulos de desarrollo en uno solo. Es decir, en un mismo entorno tienes configurado un servidor para correr la página web, una base de datos donde se pueden almacenar variables y toda la parte de programación del frontend y backend del aplicativo. En este caso, al haber elegido el lenguaje Python para el desarrollo del primer bloque, el framework a utilizar debería integrar este lenguaje. De todos los frameworks que funcionan con Python, Django es el mejor valorado.

Dentro de las características que ofrece Django, las más atractivas son la rapidez con la que se construyen aplicaciones dentro de este framework y la escalabilidad que tiene. También, en el caso del proyecto, lo bien que se integra con la lógica de modelos predictivos que se ha comentado en el bloque I. Otro de los puntos positivos de Django es que permite dividir el proyecto en distintas aplicaciones. Es decir, en un proyecto común, puede haber distintas aplicaciones cada una con su propia arquitectura MVT (hablaremos de ella en el Parte I2.2.3).

2.2.2 APLICACIÓN DJANGO:

Una vez se crea un proyecto Django, se genera un directorio con varios ficheros, entre ellos el fichero settings.py, que es el que contiene toda la configuración del proyecto.

Lo siguiente que se debe hacer es crear una aplicación, ya que Django funciona a través de aplicaciones. Cada vez que una aplicación es creada hay que añadir la url de esa aplicación

al fichero settings.py. Una aplicación Django contiene una funcionalidad especifica y su esqueleto sigue la arquitectura MVT.

Dentro de Django, se pueden tener varias aplicaciones interconectadas, pero con distintas funcionalidades. El desarrollo modular favorece la organización del proyecto y potencia su escalabilidad.

Dentro de cada aplicación se gestiona la estructura de ficheros interna y los distintos mapeados de url necesarios. Todos esos mapeados, son redireccionados al paquete del proyecto original, donde se creó el fichero settings.py y desde ahí, se conectan las urls con el servidor y luego con el navegador del cliente.

2.2.3 (MVT) MODEL VIEW TEMPLATE:

Django es un framework web que utiliza una arquitectura singular llamada Model View Template (MVT). Normalmente, los frameworks de desarrollo web utilizan una arquitectura parecida llamada Modelo Vista Controlador (MVC). Existen diferencias claves entre ambas arquitecturas y con ello Pros y Contras:

Pros:

1) Con la arquitectura MVT las partes del controlador las realiza el propio framework.
2) Tanto las peticiones recibidas como devueltas de HTTP se gestionan en la Vista.
3) Las modificaciones se realizan de manera más sencilla.
4) Es bueno tanto para aplicaciones pequeñas como grandes.

Contras:

1) Es más complicado entender el flujo programar la navegabilidad de la aplicación.
2) Hay que hacer mapeos de las URL’s.
3) Está débilmente acoplado.

[6]El típico flujo dentro de una arquitectura como esta es el siguiente. Para empezar, desde el navegador se hace una petición HTTP y el servidor de Django la procesa. Para ello accede al mapeado de url y una vez ubica el destino carga una vista. Dentro de esa vista es donde está la lógica del programa. En nuestro caso, es en las vistas donde están los algoritmos de aprendizaje automático. Desde esas vistas se pueden cargar y guardar datos en cualquier base de datos a través de las clases modelo. Estas clases son las que dan forma a la base de datos utilizada en las aplicaciones. Por último, desde la vista se cargan los templates, que son las plantillas que contienen el código HTML. Se puede observar el diagrama de la arquitectura y el flujo en la Ilustración 23. Además, se pueden enviar datos desde la vista a ese template a través de la función `render()` que ofrece Django. Hablaremos de cómo funciona está conexión Vista-Template y como se envían los datos procesados para mostrarlos a través del template en el navegador del cliente.

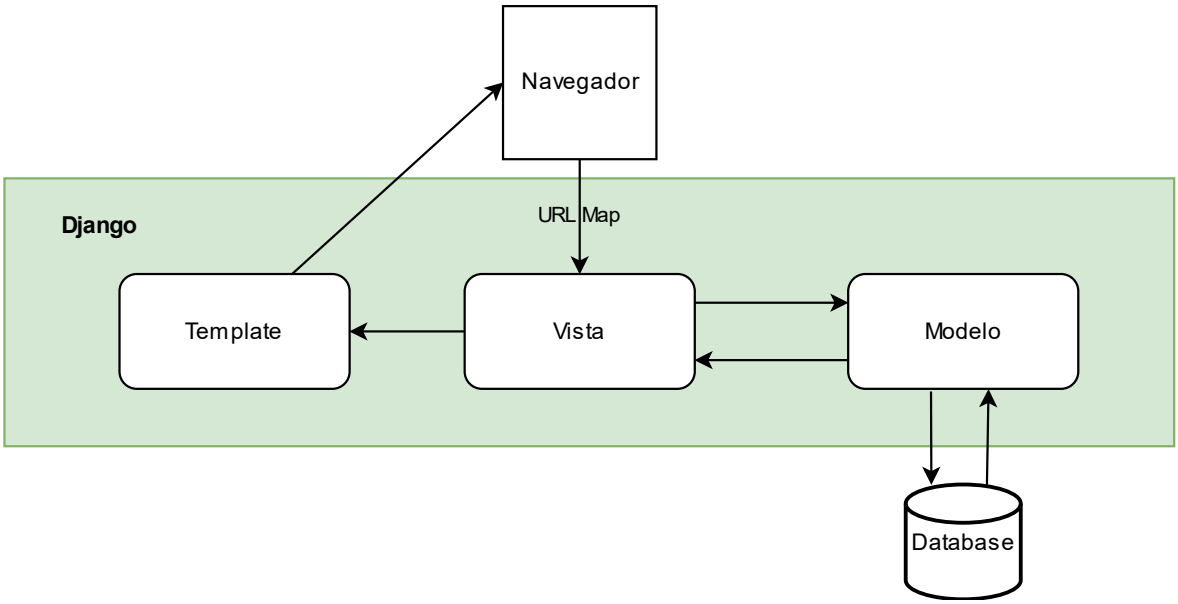


Ilustración 27 - Arquitectura MVT

[6] Elaboración propia

2.2.4 TEMPLATE / FRONTEND:

Una de las facilidades que ofrece Django es la simpleza con la que se cargan los htmls en el navegador del usuario (Frontend). Django tiene una librería que ofrece shortcuts. En esta librería hay un método que se llama `render()` y es el que se encarga de renderizar la página html. Esto funciona de la siguiente manera:

1) Se recibe una petición HTTP y a través del mapeado url se llega a una vista. Esta vista recibe la request y realiza la lógica que sea (ejemplo: Cargar un modelo de regresión lineal con datos obtenidos de una base de datos)
2) Los datos que queramos mostrar a continuación por pantalla se cargan en un diccionario Python (por convenio se llama `context)`. En este diccionario se pueden cargar objetos Python de todo tipo, tan solo hay que darles un nombre a esos elementos para luego poder ubicarlos desde el template. Una carga típica de un valor en el contexto tiene la siguiente pinta:

```
context[“valor”] = variable
```

3) Una vez se tienen los datos cargados en el diccionario, simplemente se debe llamar al método render y devolverle la petición recibida, la url con el html que se va a cargar en el servidor, y los datos cargados en el contexto para que se puedan acceder a ellos desde el html y mostrarlos por pantalla.

```
return render(request, “pagina.html”, context)
```

Una vez entendido este proceso, lo siguiente que debemos tener en cuenta es como se cargan en el html los elementos enviados a la plantilla a través del contexto. Django ofrece una variedad de etiquetas y anotaciones para acceder de forma simple a estos datos. Si queremos directamente mostrar la variable cargada en el contexto, se debe utilizar la siguiente anotación:

```
<h3>El valor en el contexto es: {{valor}}</h3>
```

Las dobles llaves "{{ - }}" hacen referencia al contexto, y con el nombre que hayamos indicado en la vista se accede a esa variable.

Otra funcionalidad que ofrece Django y que es realmente útil es la de poder aplicar código Python dentro del html que define el template. Un caso de uso muy descriptivo para ejemplificar esta funcionalidad es cuando en el contexto pasamos una lista de elementos y necesitamos mostrarlos como opciones en un formulario.

Vista:

```
context["lista"] = lista_de_valores
```

Template:

```
{% for element in lista %}
     <option value="{{element}}">{{element}}</option>
{% endfor %}
```

Las etiquetas "{% - %}" indican a Django que lo que está escrito dentro es parte de código. De esta forma se pueden añadir condiciones if, y cualquier tipo de bucle que facilita mucho la lógica para mostrar los elementos del contexto.

Estas etiquetas han sido de gran utilidad durante el desarrollo del proyecto ya que, a la hora de cargar las variables para la generación de los modelos de aprendizaje automático, se necesitaba un formulario dinámico que se ajustase a distintas bases de datos.

2.2.4.1 POST Request:

Por último, si queremos enviar datos desde un template a la vista tenemos que hacer una petición HTTP Post. Esta petición se realiza en los formularios html utilizando la siguiente etiqueta:

```
<form action="{% url 'mlalgorithms'%}"  method="POST">
```

Por motivos de seguridad, los datos que se envíen necesitan de cierta certificación, y Django ofrece este servicio. Esta medida de seguridad sirve para protegerse ante ataques CSRF.

Los ataques CSRF son ataques que consisten en hacer creer a los usuarios de una aplicación web, que están rellenando un formulario cargado por la web aunque sea realmente un usuario malicioso quien haya cargado ese formulario. Sin está protección, un hacker podría crear un formulario falso donde se pidan contraseñas o datos de tarjetas de crédito, y robar esos datos de los usuarios.

La solución a este problema se consigue con el uso de tokens. Un token csrf es un valor único, secreto e impredecible que se genera en el servidor y se transmite hasta el cliente para validar el origen de la petición. Para implementar el uso de este token simplemente hay que añadir la siguiente línea en los templates cuando se realicen peticiones http al servidor:

```
{% csrf_token %}
```

Los datos enviados a través de la petición POST son recogidos en la vista de la siguiente forma:

```
    if request.method == "POST":
        dato_1 = float(request.POST["dato_1"])
        dato_2 = str(request.POST["dato_2"])
```

Los datos “dato_1” y “dato_2” se referencian desde el html añadiendo un parámetro name a las etiquetas de los formularios:

```
<input type="checkbox" value="selected" name="dato_1">
```

Capítulo 3. ESTADO DE LA CUESTIÓN

La oferta de servicios relacionados con la informática y la computación está viviendo un auge desde el inicio de la cuarta revolución industrial. Existen una gran variedad de productos que se ofrecen -aaS (Las siglas vienen de as a Service, que traducido al español significa "como servicio"). Todos estos servicios se suelen ofertar con subscripciones y en la nube, aunque cada vez se están viendo modelos de oferta más innovadores y distintos. ¿Pero porque se ha derivado a este modelo de negocio y porque está teniendo tanto éxito?

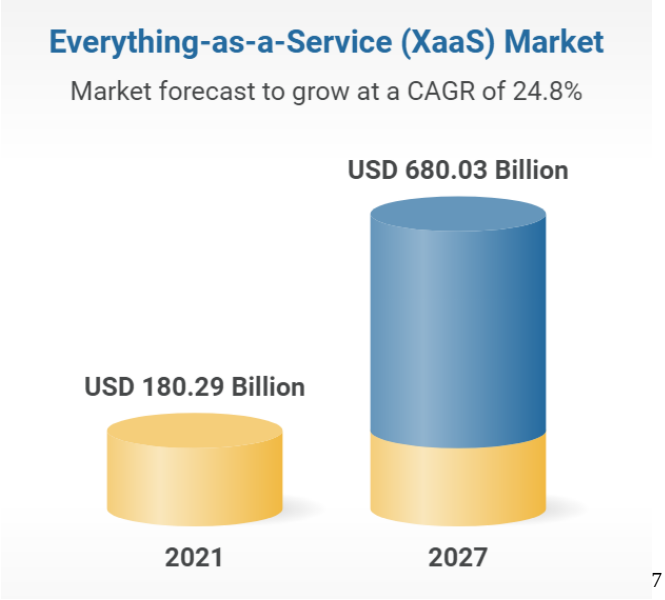
3.1 EL BOOM -AAS

Con el mundo de la informática y computación ocurre un gran problema. Los costes fijos que supone tener una infraestructura informática robusta y segura son muy elevados. La computación es de los sectores que más se beneficia de las economías de escala. No tiene sentido comprar y desarrollar una infraestructura de servidores y ordenadores para entrenar un par de modelos predictivos y luego no volver a usarla. El modelo comercial "-aaS" propone una gran solución a todas las empresas que no se dedican al mundo de la computación o informática, pero que necesitan hacer uso de ella de una manera esporádica.

Otro de los grandes beneficios de los modelos "-aaS" es el ahorro en el coste del personal técnico y la ventaja de no necesitar a ingenieros cualificados para realizar las tareas de forma interna. Es decir, si vas a llevar a cabo un proyecto que necesita una potente infraestructura de base de datos y de servidores, en vez de contratar a personal para desarrollarla, puedes solicitar esa infraestructura como servicio a una de las grandes tecnológicas. Te ahorras todos los costes fijos, pagas de manera mensual por el servicio y no necesitas a personal adicional para crear desde 0 la infraestructura.

Por estas razones, el mercado de los servicios de computación en la nube está creciendo con un ritmo increíble. Según Research and Markets, la tienda más grande del mundo dedicada a la investigación de mercados, en 2027, el mercado de los productos como

servicio (XaaS) llegará a tener un valor de 680 billones de dólares, aproximadamente la mitad del producto interior de España.



[7]

Ilustración 28 – Crecimiento esperado del mercado de los productos como servicio.

Dentro del mercado de los productos como servicio existen varias categorías como por ejemplo IaaS (infraestructura como servicio), SaaS (software como servicio) o el DaaS (Datos como servicio). Este proyecto entra dentro de la categoría del MLaaS, y a continuación analizaremos las características de este mercado, las soluciones que ofrecen los competidores y como se ajusta nuestro proyecto a la situación.

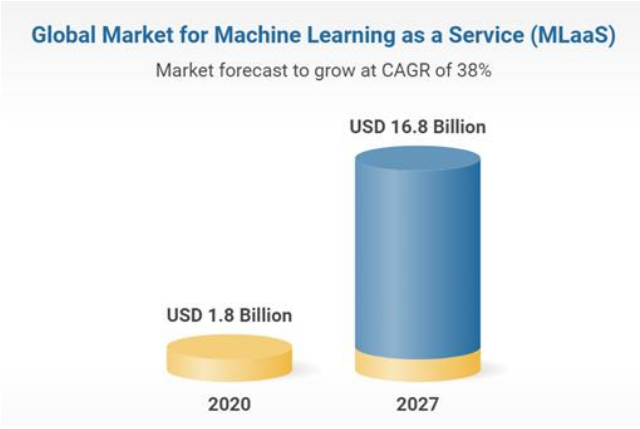
[7] Referencia de la Ilustración 28 – Crecimiento esperado del mercado de los productos como servicio.: Research and Markets - https://www.researchandmarkets.com/

3.2 MLAAS

Echando la vista atrás, en el capítulo 2 comentamos que este proyecto está dividido en dos grandes bloques. Por un lado, el bloque de Machine Learning, que es el núcleo del proyecto y por otro lado, el bloque de desarrollo web, que es el que convierte al producto en un servicio. Es decir, que con la combinación de los dos bloques de desarrollo obtenemos un proyecto centrado en la oferta del Machine Learning como servicio (MLaaS).

3.2.1 ANÁLISIS DEL MERCADO

El mercado del Machine Learning como servicio, dentro de las categorías que engloban el mercado de productos como servicio, es la que se predice que más crecimiento tendrá en los próximos años. En el mercado de los productos como servicio, el ratio del crecimiento anual compuesto (CAGR) esperado es de 24.8% y solo la categoría de MLaaS tiene un CAGR esperado de 38%, es decir, 13,2 puntos porcentuales más. En la Ilustración 29 - Crecimiento esperado del MLaaS se puede ver la gráfica del crecimiento esperado del mercado.



[8]

Ilustración 29 - Crecimiento esperado del MLaaS

[8] Referencia de la Ilustración 28 – Crecimiento esperado del mercado de los productos como servicio.: Research and Markets - https://www.researchandmarkets.com/

Esto significa, que en los próximos años van a surgir muchas oportunidades en el mercado y que habrá una pelea continua entre los grandes jugadores por hacerse con la mayor cuota de mercado posible.

Por estas razones, este es un momento ideal para la realización de este proyecto, ya que estamos al principio del boom, y un producto como este puede aportar una gran utilidad a las pequeñas y medianas empresas que lo necesiten.

3.2.2 PRINCIPALES COMPETIDORES

Los principales competidores en este mercado son empresas tecnológicas muy avanzadas y con una gran infraestructura. Para tener una cuota alta en un mercado como este se necesitan grandes servidores y un alto poder de computación ya que todo lo que gira en torno al aprendizaje automático son los datos, y su procesamiento y almacenaje a gran escala es bastante costoso. Estas empresas se centran en dar soluciones a grandes empresas con una gran cantidad de datos. Además de ofrecer modelos de ML que están completamente fuera del procesamiento de datos estructurados como puede ser procesamiento de imágenes y videos o del lenguaje natural.

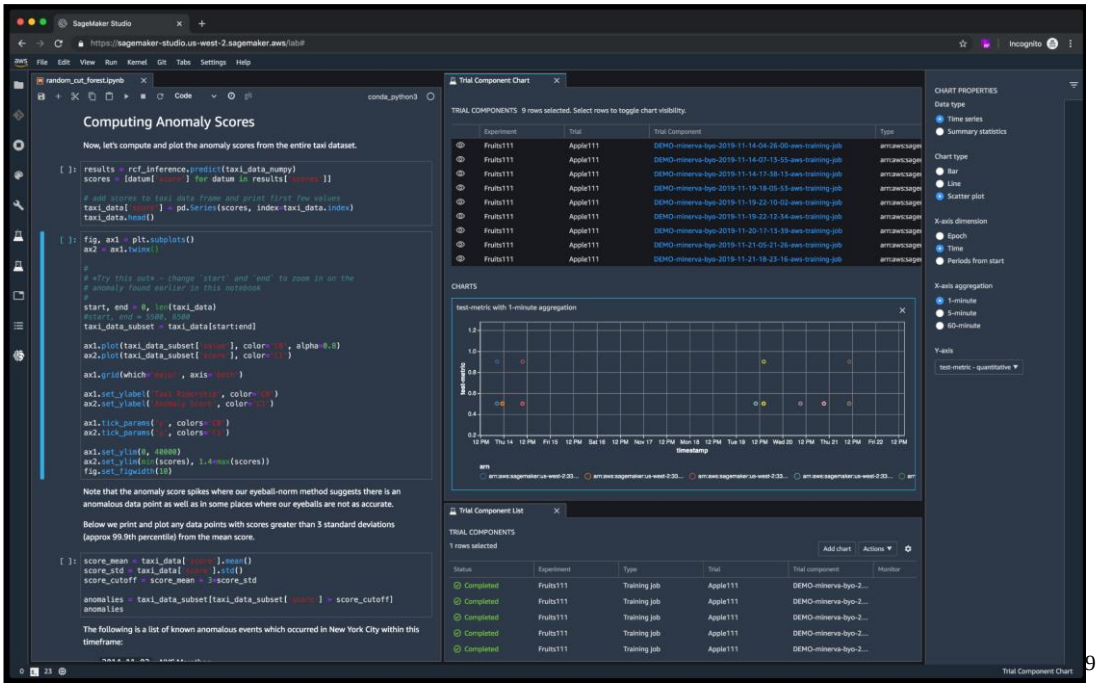
Los principales competidores son:

1) **Amazon Web Services:** Dentro de los servicios que ofrecen en el mercado de MLaaS, Amazon SageMaker es el producto que más se parece a lo que ofrece este proyecto. El resto de los servicios están enfocados a la transcripción de lenguaje natural, detención de objetos en imágenes y videos y reconocimiento de voz.

   SageMaker en cambio, proporciona servicios a los científicos de datos para entrenar e implementar modelos de aprendizaje automático sin preocuparse del código.

Amazon SageMaker

De todas formas, el enfoque sigue siendo para perfiles que tienen un alto entendimiento de los distintos algoritmos y modelos de ML y, además, para empresas con un volumen de datos importante. No hay más que echar un vistazo a la interfaz de la plataforma para darse cuenta de que este servicio no es el adecuado para perfiles menos técnicos.



[9]

Ilustración 30 - Interfaz de usuario SageMaker

[9] Captura de pantalla del servidor de Amazon SageMaker

2) **Google AI Platform**: Al igual que Amazon, Google se centra en una gran variedad de productos del mundo de la IA como son el NLP (Procesamiento del Lenguaje Natural) y la visión por ordenador. Dentro de todos los servicios que ofrece, AutoML es el que más se parece a lo que ofrece la aplicación desarrollada en este proyecto. Aun así, dentro de su servicio de Machine Learning para perfiles poco técnicos, la versión de algoritmos para datos estructurados en tablas está todavía en versión Beta.

3) **Microsoft Azure:** Azure tiene menos variedad que Amazon y Google. Además, su foco no está en ofrecer soluciones a personas con un perfil técnico nulo. Azure propone soluciones de ML para empresas grandes y con una gran cantidad de datos a manejar. Más que una herramienta que sustituya la necesidad de los perfiles técnicos para realizar predicciones es una herramienta que complementa a esos perfiles además de proporcionar una infraestructura donde poder entrenar los modelos con el menor coste de tiempo posible.

4) **IBM Watson:** IBM ofrece también servicios de reconocimiento visual y de clasificación del lenguaje natural. Además de ello, ofrece un servicio de AutoAI,

que consiste en un modelo que limpia y prepara los datos de forma automática, y después, compara entre distintos modelos seleccionando el mejor para realizar predicciones. Todo esto lo hace a través de su entorno web con la herramienta de Watson. Aun así, esta herramienta está enfocada para grandes empresas y perfiles que siguen requiriendo ciertos conocimientos técnicos.

5) **BigML:** Esta empresa propone varias formas para dar sus servicios. El principal y más utilizado es a través de llamadas a APIs REST. Esto es un microservicio que se ejecuta desde los servidores de BigML y al que se accede haciéndole peticiones desde otras páginas webs. También propone una interfaz web donde se pueden subir datos y desarrollar modelos predictivos.

Capítulo 4. DEFINICIÓN DEL TRABAJO

Ahora ya sabemos que soluciones se han propuesto en el mercado y una pequeña descripción de cada una de ellas. Dentro de este marco debemos entender porque nace este proyecto y cuáles son sus objetivos. Además, en este capítulo también describiremos la metodología de trabajo que se ha seguido y la planificación que se ha llevado a cabo.

4.1 JUSTIFICACIÓN

En la introducción, hemos comentado cual era el estado en el que se encuentra ahora la humanidad con respecto a la cuarta revolución industrial, la revolución de la IA. Algo que está claro es que el mundo del Machine Learning y el desarrollo de modelos predictivos para cualquier tipo de tareas, está actualmente en pleno auge y cada día se ven más casos donde se implementan estos modelos. Pasó lo mismo con internet y la digitalización. Cuando se creó internet, podríamos decir que, durante la tercera revolución industrial, poco a poco las empresas y las personas se fueron digitalizando hasta que el paradigma de la sociedad cambió. Ahora, todo está en internet, y lo que no está seguramente acabe estando o desapareciendo. Pocas veces pensamos en la magnitud que ha tenido este proceso de digitalización. Si lo piensas, hoy en día si quieres ir a comer, te puedes meter en internet y acceder a miles de restaurantes cada uno con su página web. O si quieres tener una consulta con una clínica privada, también puedes acceder a miles de páginas webs de centros privados. Esas páginas han sido creadas por alguien que ha ayudado a esos sitios a digitalizarse. Los trabajadores de esos restaurantes, de esas clínicas no tienen conocimientos técnicos ni saben programar, pero tienen una página web y están digitalizados.

Una vez entendemos esto, y observamos lo que está pasando con la revolución de la IA, tarde o temprano las nuevas tecnologías que se están desarrollando acabaran llegando a estos restaurantes o clínicas, porque necesitaran utilizar modelos predictivos o gestores

inteligentes para ser competitivos. Es entendiendo estos ciclos de innovación y observando lo sucedido en el pasado donde se comprende perfectamente con que justificación nace este proyecto. Si el proceso de digitalización ha sido global, el proceso de AIlización también lo será.

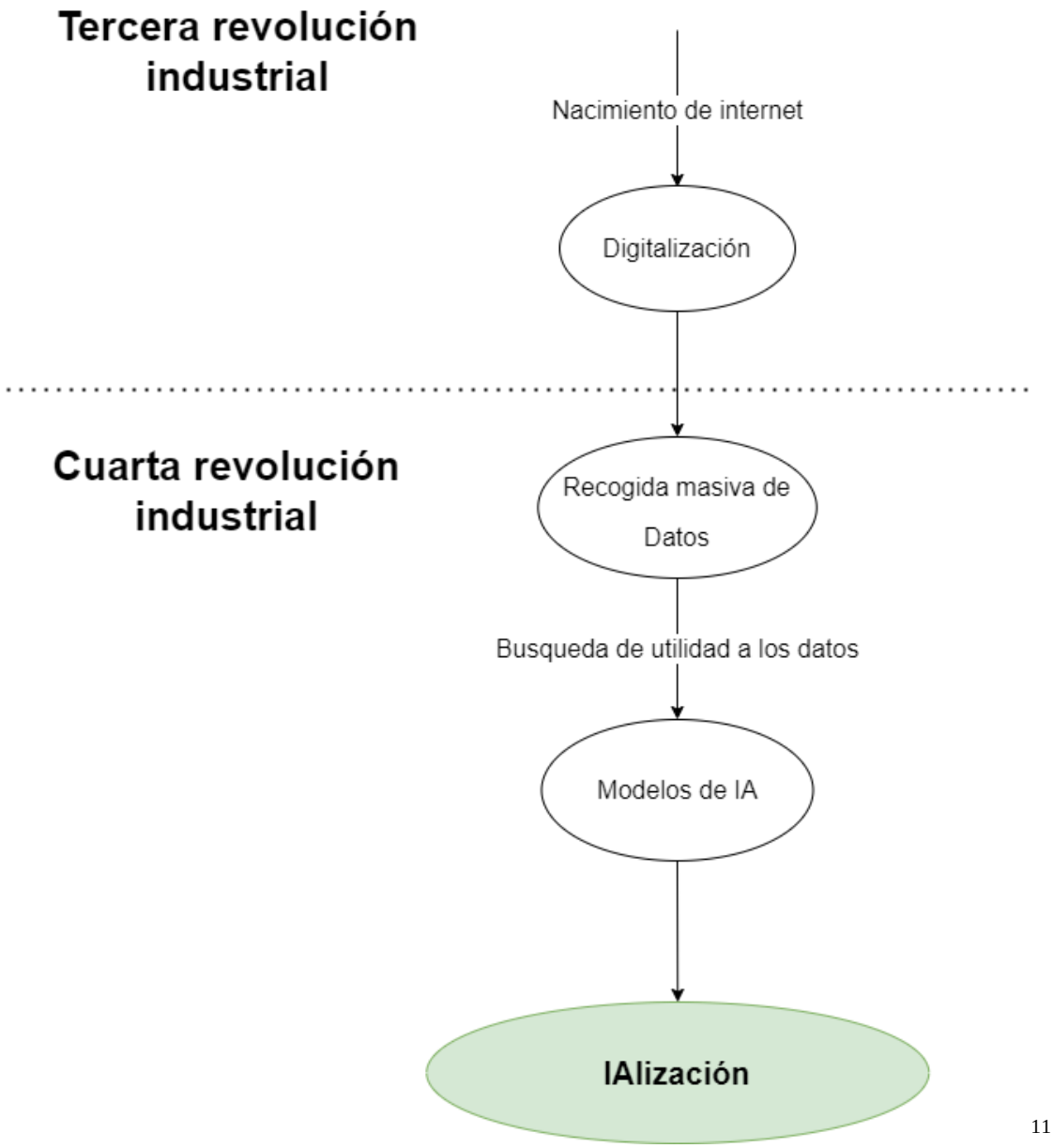
4.1.1 IALIZACIÓN

El proceso de IAlización es un término inventado que hace de homologo al proceso de digitalización, pero con la inteligencia artificial.

Si tratamos de observar con perspectiva lo sucedido en los últimos años, primero, con el nacimiento de internet, el mundo real se digitalizó y convirtió en un mundo digital. Una vez digitalizado, recoger datos se convirtió en una tarea muy simple. Las empresas más punteras y con más conocimientos técnicos se dieron cuenta del valor que tenían esos datos y empezaron a desarrollar modelos matemáticos complejos que fueran capaces de aprender sobre ellos. Solo tenemos que irnos al paper publicado más innovador y disruptivo en el campo de la IA titulado "Attention is all you need", donde se describe la arquitectura de los transformadores [10]. Ese paper fue escrito en su mayoría por ingenieros de Google, los pioneros en digitalización y dueños de la mayor cantidad de datos que existen en el mundo. Con esto, lo que quiero transmitir es que después de los datos, viene la IA. En la Ilustración 31 - Diagrama del proceso de IAlizaciónse puede observar un diagrama que explica este proceso.

Ahora, estos modelos son cada vez más accesibles a empresas menos técnicas. Pero, todavía no existe una herramienta que vaya destinada a esos negocios que no tienen idea de cómo programar o ni si quiera de lo que es el Machine Learning. Este proyecto se centra en el acercamiento de esta tecnología a perfiles no técnicos.

[10] Los transformadores son la arquitectura de IA más puntera que existe a día de hoy. Su poder reside en que en vez de procesar los datos secuencialmente, se llega a un entendimiento del contexto de los datos y de esta manera se obtiene información adicional que ayuda a que el aprendizaje sea más robusto.



[11]

Ilustración 31 - Diagrama del proceso de IAlización

[11] Elaboración propia

4.1.2 ¿QUÉ NO SE ESTÁ HACIENDO?

En el apartado *3.2.2 Principales competidores*, hemos mencionado lo que estaban haciendo las empresas con más representación en el mercado del MLaaS. Todas ellas proponen soluciones válidas para grandes empresas con necesidad de grandes infraestructuras y con capacidad y personal técnico. Pero estas soluciones se quedan demasiado grandes para los pequeños negocios o autónomos que tan solo necesitan sacar predicciones de un par de tablas de Excel. Al igual que la digitalización, la ialización se producirá tanto a gran como a pequeña escala. Este proyecto acerca la increíble capacidad que tienen los modelos de aprendizaje automático a perfiles sin conocimientos técnicos y a empresas que no precisan de una gran infraestructura para analizar sus datos. Por hacer una comparación con lo ocurrido en la digitalización, ese proyecto es el homologo a los negocios que han transformado digitalmente a todas esas pequeñas empresas y negocios que se estaban dejando de ser competitivos.

Estos grupos no están siendo atendidos por ninguna empresa, pero conociendo las grandes ventajas que pueden ofrecen estos servicios sabemos que estas tecnologías están por llegar, y este proyecto pretende ser el catalizador de la iniciación de este proceso en el mundo de los pequeños negocios y empresas.

4.2 OBJETIVOS

Entendiendo el contexto y conociendo la justificación del proyecto, los objetivos de este se hacen más claros e intuitivos. A la hora de definir objetivos se pueden utilizar varios niveles y capas. Hay objetivos generales sobre los que nacen otros objetivos más concretos. El objetivo a más alto nivel es simple y concreto: Producir un acercamiento de las tecnologías de ML a las pequeñas empresas y negocios.

Ahora bien, para concretizar en pequeños objetivos y enfocar el trabajo de manera más precisa debemos preguntarnos como se lograría cumplir este objetivo principal.

Los distintos objetivos que derivan del objetivo principal son los siguientes:

- **Para el usuario:**
  - La aplicación debe tener una interfaz simple para garantizar su entendimiento.
  - La experiencia de usuario debe ser prioridad y se debe hacer foco en garantizar la simpleza de uso ya que la aplicación va destinada a perfiles no técnicos.
  - La manera de subir datos e interactuar con ellos debe ser intuitiva y sencilla.
  - Un usuario debe ser capaz de crear un modelo de machine learning y poder realizar predicciones sobre el mismo.

- **Funcionalidades:**
  - La aplicación debe incorporar una variedad de modelos de machine learning ofreciendo versatilidad.
  - La aplicación debe funcionar con cualquier base de datos que sea subida por los usuarios, para ello se debe ofrecer un sistema de limpiado de datos básico.

4.3 METODOLOGÍA

Decidir una buena metodología de trabajo es fundamental para llevar un desarrollo correcto y organizado del proyecto, especialmente cuando se trata de un proyecto de desarrollo de software. Existen distintas estrategias y metodologías que indican de qué manera es más productivo organizarse o planificar el trabajo dependiendo de los requisitos de este. Las dos metodologías más comunes son las metodologías agiles (Agile) o las metodologías en cascada (Waterfall). Cada una de estas dos está enfocada y pensada para usar en distintas situaciones.

Las prioridades de las metodologías agiles son:

- Respuesta al cambio
- Colaboración con el cliente
- Producto funcionando
- Personas y sus interacciones

Las metodologías agiles se utilizan cuando se necesita un alto grado de adaptación a las peticiones cambiantes de los clientes. También es útil cuando se trabaja con un cliente en un proyecto software innovador, donde hay variaciones en el estado del arte de las tecnologías aplicadas y se necesita poder actualizarse con "agilidad".

Las metodologías en cascada son distintas. Una planificación basada en la metodología waterfall es una planificación cerrada de principio a fin con distintas etapas de desarrollo. Una planificación ágil en cambio se centra en ordenar los requisitos en distintos bloques de entrega denominados backlogs, y cada periodo de tiempo fijado se desarrolla completamente en un ciclo ese backlog, es decir, que en cada periodo de entrega se pasa por las distintas etapas repetidamente.

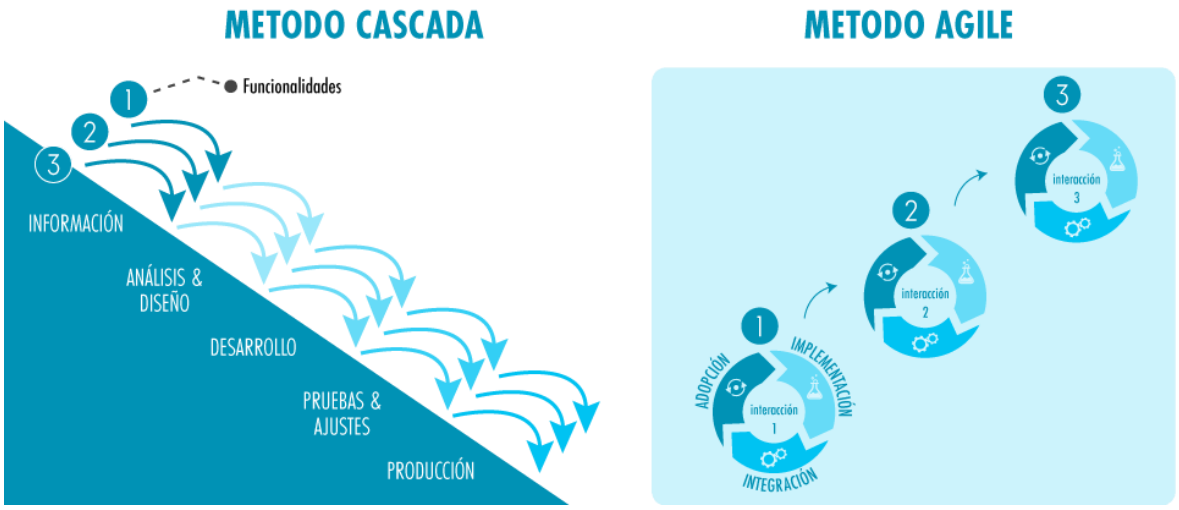


Ilustración 32 - Cascada vs Agile [12]

En este proyecto, las tareas y requisitos están predefinidos y son fijos. Es por ello por lo que la metodología elegida para trabajar en el proyecto ha sido la metodología en cascada o Waterfall.

[12] Referencia de la imagen: https://consulting-systems.tech/cascada-vs-scrum/

4.4 PLANIFICACIÓN

Una vez seleccionada la metodología de trabajo en cascada, ya tenemos un framework mental para poder organizarnos de manera más productiva y rendir más en el desarrollo del proyecto.

En la planificación para el desarrollo de este proyecto se ha utilizado distintos niveles y subniveles de etapas. En el plano y nivel más alto y general, la división de tareas fue la siguiente:

[13]

Ilustración 33 - Diagrama Gantt del Proyecto

[13] Los tiempos indicados en el diagrama Gantt son estimaciones flexibles.

Cada una de estas etapas son cruciales para el desarrollo del proyecto, pero dentro de cada una se subdividen distintas etapas y una planificación exhaustiva.

4.4.1 DEFINIR VISIÓN

Durante el desarrollo de esta etapa, la misión principal era definir cuál es el proyecto que se va a realizar, que alcance se quiere tener y que problema quiere atacar. Esta etapa podría considerarse como previa a lo que es el desarrollo del trabajo de fin de grado.

Durante esta etapa y simultaneando con la etapa de recopilación de información nació la idea del proyecto que se ha llevado a cabo. También se desarrollaron el Anexo A y B con las características del trabajo.

4.4.2 RECOPILACIÓN DE INFORMACIÓN

Esta etapa va de mano con la etapa de definición de visión. Aquí es cuando se estudió la viabilidad del proyecto, los recursos que había para utilizarlos, si había algún trabajo parecido. Es una etapa que se anexa e integra completamente a la etapa de formación.

4.4.3 FORMACIÓN

Cuando la información recopilada empieza a ser más precisa, poco a poco se convierte en formación. Las ideas que están en el aire se empiezan a asentar en páginas con tutoriales y cursos donde se aprende a utilizar los recursos que se han encontrado para el desarrollo del proyecto.

Durante esta etapa, primero realicé un cursillo de formación en Python, el lenguaje que había decido utilizar para el desarrollo del proyecto. Gracias a que conozco otros lenguajes como Java y C, aprender Python fue relativamente sencillo, aunque llevó algo de tiempo programar de forma fluida. Otra de las formaciones que seguí fue un curso de Django. En youtube hay varios tutoriales donde te enseñan a construir una aplicación con Django desde 0, y así hice.

Por último, la formación en Machine Learning y en el manejo de las librerías utilizadas como Sklearn, Numpy y Pandas.

4.4.4 DISEÑO

Al igual que ocurre con la etapa de información y formación, una vez que empiezas a manejarte con las herramientas que vas a utilizar y conoces su funcionamiento, en tu cabeza se empiezan a diseñar distintas arquitecturas para el desarrollo del proyecto que has definido. Esto ocurre naturalmente cuando tienes ya unos conocimientos avanzados.

Durante esta etapa, se elaboró el plan y la estrategia de desarrollo del proyecto. El desarrollo se dividió en varios módulos y distintas etapas que mostraremos en el siguiente apartado.

4.4.5 DESARROLLO

Los módulos principales de la aplicación, y de los que hablaremos a lo largo del capítulo 5 son los siguientes:

1) Algoritmos de ML
2) Gestión de bases de datos y .csv
3) Guardado y predicción de modelos entrenados

Estos tres grandes módulos fueron desarrollados secuencialmente, y finalmente, durante las últimas semanas de desarrollo, se programó el estilo de la aplicación, la navegabilidad y la interconexión de los módulos.

También, se añadieron mejoras como el muestreo de gráficas y limpieza de los .csv subidos.

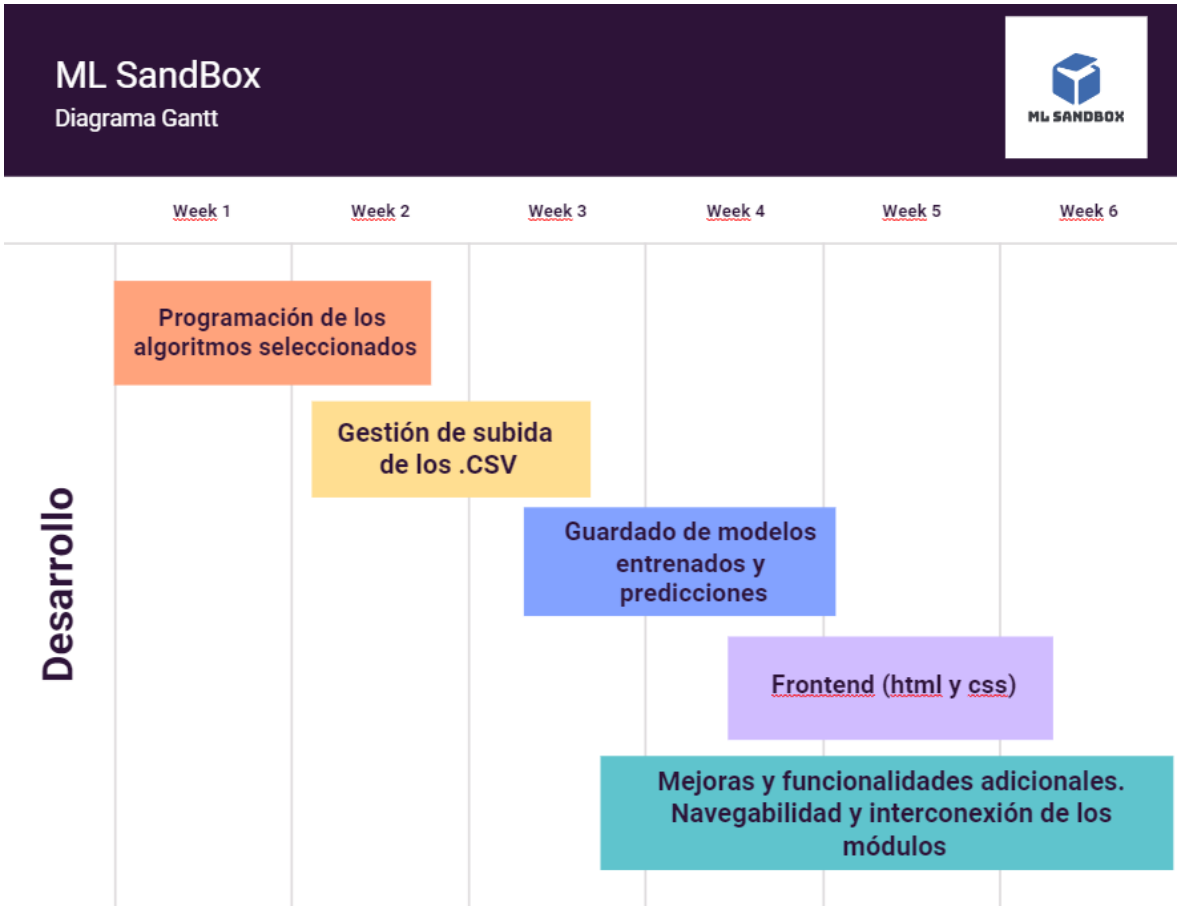


Ilustración 34 – Diagrama Gantt del desarrollo

Capítulo 5. DESARROLLO DEL PROYECTO

Como hemos comentado en el capítulo 2, dentro de un solo proyecto en Django pueden existir distintas aplicaciones con distintas funcionalidades y cada una con su arquitectura MVT. La mejor forma de explicar el desarrollo de este proyecto es haciéndolo de manera modular y siguiendo la arquitectura que se ha seguido en el desarrollo. Es decir, que iremos aplicación por aplicación explicando sus funcionalidades, sus ventanas y el diseño que se ha seguido. Pero antes de eso, es necesario entender la arquitectura general del proyecto.

5.1 ARQUITECTURA GENERAL

Lo primero que se debe hacer cuando se va a desarrollar un proyecto con Django es crearlo. Los proyectos de Django se crean desde la terminal con el siguiente comando:

```
django-admin startproject <nombre_del_proyecto>
```

Con la generación del proyecto, Django crea automáticamente distintos ficheros de Python que contienen información del proyecto, la configuración del servidor y un fichero de gestión del proyecto con distintas funciones. Este último fichero es el más importante y se llama manage.py. El comando que se debe ejecutar es el siguiente:

```
python manage.py startapp <nombre_de_la_app>
```

Este proyecto se divide en 4 distintas aplicaciones.

1) SandBoxApp
2) Databases
3) ML
4) Models

Cada una de estas aplicaciones contiene una funcionalidad distinta dentro del proyecto.

Cada vez que creas una aplicación, dentro de la carpeta del proyecto se genera una subcarpeta con el nombre de la aplicación creada que contiene los distintos ficheros:

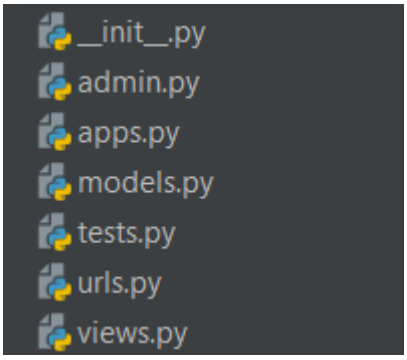


Ilustración 35 - Ficheros de las apps de Django

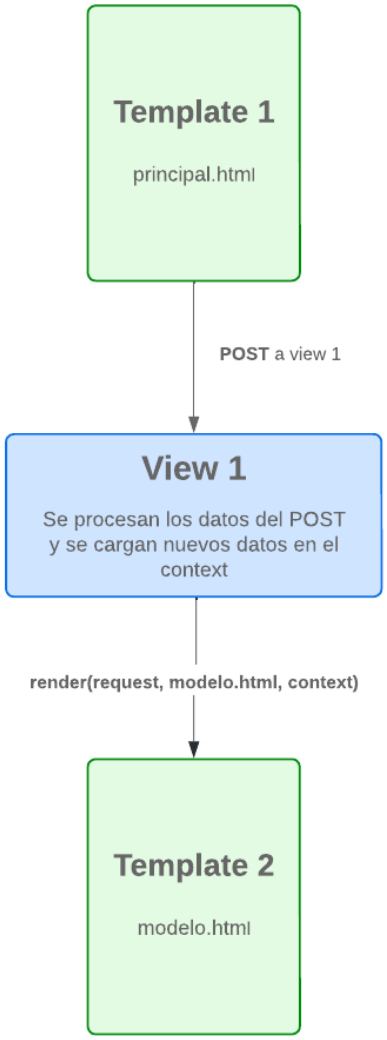
Para correr el servidor con el proyecto debemos llamar a otra función del fichero manage.py llamada runserver. El comando es el siguiente:

```
python manage.py runserver
```

Antes de entrar más en detalle y explicar el desarrollo de cada aplicación es crucial entender cómo se pasa la información de vista a vista y como se muestra por el template.

Como habíamos explicado en el capítulo 2, usando la función `render()` que ofrece Django, la muestra de los templates por el servidor se simplifica enormemente. Además, se puede enviar información adicional a los html a través de un diccionario que se carga desde la vista. Este diccionario por convenio se llama context, y ahí es donde se cargan todas las variables y datos que se necesiten mostrar en los templates.

También, se necesitan recoger datos de las peticiones POST de las vistas anteriores, procesarlos y luego cargarlos en el siguiente template. El diagrama de flujo de una operación como esta es el siguiente:



[14]

Ilustración 36 - Flujo de datos Vista-Template

[14] Elaboración propia

Los datos de la petición POST realizada desde la template 1 a la vista 1 se recogen de la siguiente manera:

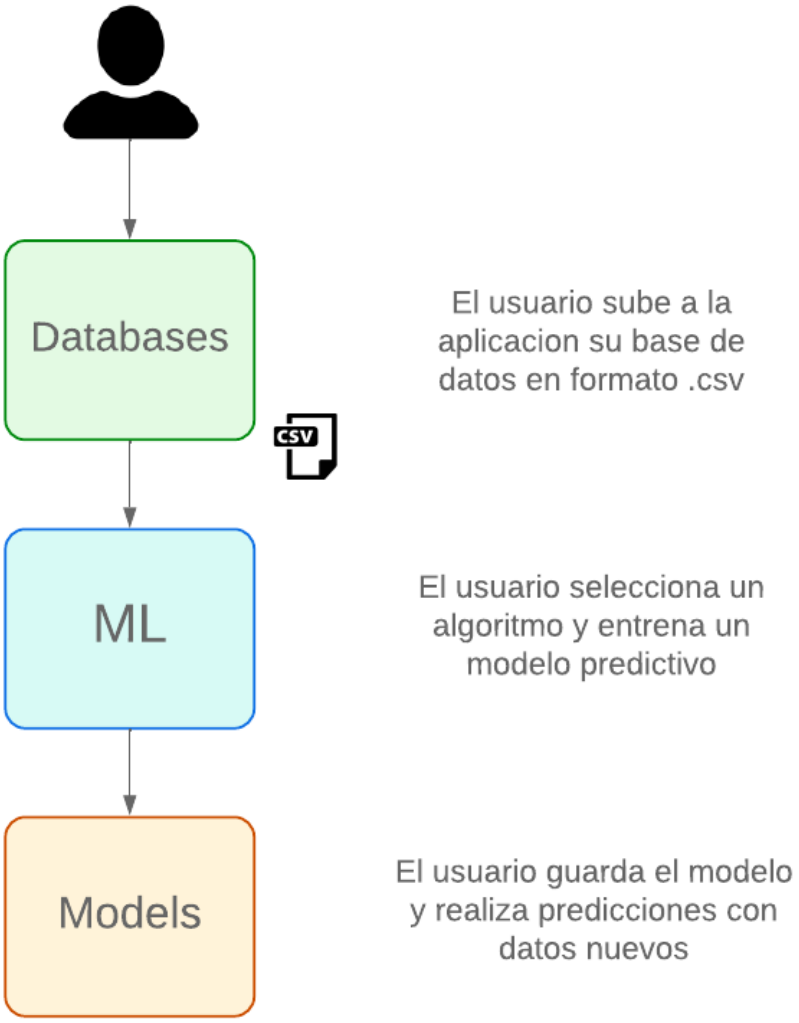
```
if request.method == "POST":
    filename = str(request.POST["filename"])
```

Luego, se rellena el siguiente contexto para cargar el template de esa vista, en este caso modelo.html:

```
context['database'] = filename
df = pd.read_csv('media/'+str(filename))
context['columns'] = list(df.columns)
```

Finalmente se renderiza y se carga el template:

```
return render(request, "ML/linear_regression.html", context)
```



[15]

Ilustración 37 - Diagrama de flujo del proyecto

[15] Elaboración propia

5.2 SANDBOXAPP

La primera aplicación es la aplicación base. Esta contiene solamente la ventana de inicio y es la que redirige al resto aplicaciones del proyecto.

Este es el ejemplo de aplicación más simple e ilustra muy bien en que consiste la arquitectura Modelo Vista Template de Django.

Dentro de los distintos ficheros que genera django cuando se crea una app, en el fichero views.py es donde se programan las distintas vistas de la app. Cada vista tiene asociado un template, que en nuestro caso es una página de html.

Pues bien, en la SandboxApp, tan solo hay una vista llamada home, y es la encargada de mostrar el template de la ventana de home.

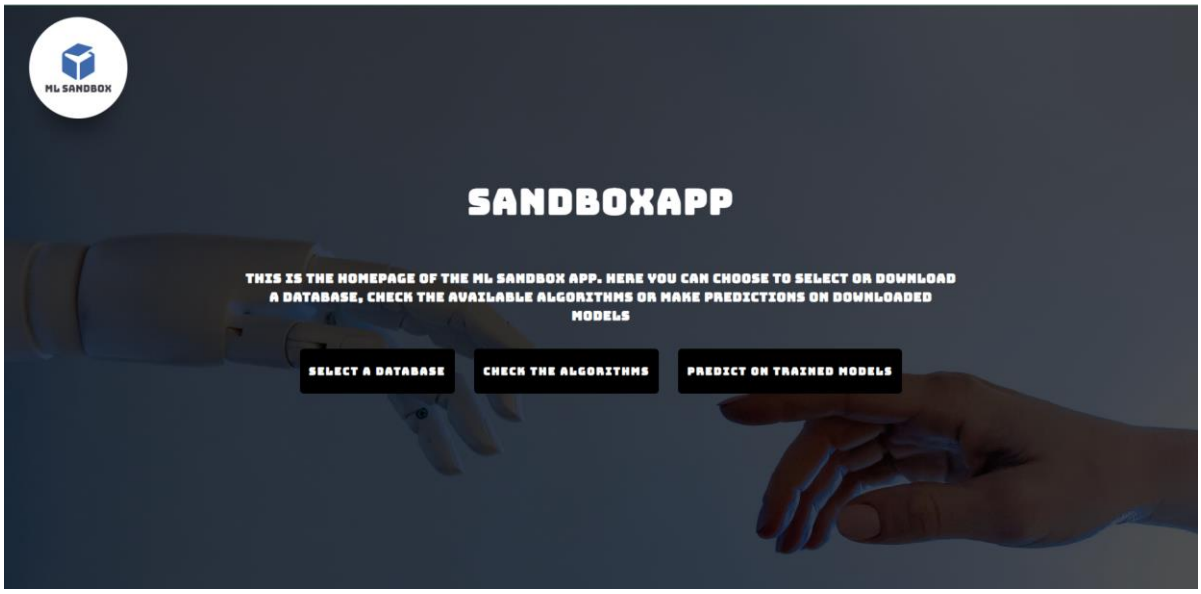


Ilustración 38 - Ventana principal

El código para renderizar la página desde el fichero views.py es el siguiente:

```
def home(request):

    return render(request, "SandBoxApp/home.html")
```

Desde esta ventana se pueden acceder a las 3 distintas funcionalidades principales del proyecto. La primera es la de selección de base de datos, que corresponde a la aplicación Databases. La segunda opción lleva al usuario a la ventana donde se encuentran los distintos algoritmos de machine leaning, que corresponde a la aplicación de ML. La última opción lleva al usuario a los modelos que ha grabado previamente para que pueda realizar predicciones sobre ellos. Si no tiene un modelo creado deberá crearlo antes para poder usarlo.

5.3 DATABASES

Desde esta aplicación se gestiona la lógica de subida y limpieza de datos. Las funcionalidades son subir, borrar y seleccionar bases de datos para la creación de modelos predictivos.

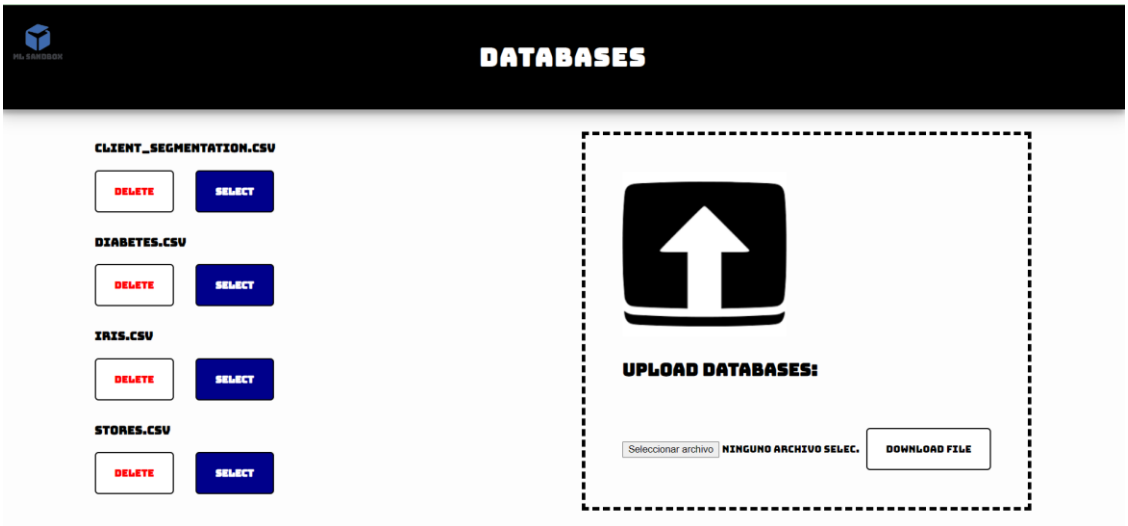


Ilustración 39 - Ventana de Databases

Para subir una base de datos se debe hacer lo siguiente:

1) Seleccionar el archivo que quieras subir desde local.
2) Presionar download file
3) Seleccionar para realizar predicciones.

Las bases de datos deben ser .csv para el correcto funcionamiento de los modelos.

Para conseguir borrar y seleccionar las bases de datos sin cambiar la template se deben programar varias vistas con el mismo template común. En este caso, la aplicación de databases contiene 4 vistas distintas:

1) **Databases**: Esta es la vista principal que carga todas las bases de datos que hayan sido previamente descargadas por el usuario.

```
def databases(request):
    context = dict()
    onlyfiles = [f for f in listdir(os.path.abspath('media/')) if
isfile(join (os.path.abspath ('media/'), f))]

    context['databases'] = onlyfiles

    if request.method == "POST":
        myfile = request.FILES['upload']
        fs = FileSystemStorage()
        filename = fs.save(myfile.name, myfile)
        context['url'] = fs.url(filename)

    return render(request, "Databases/databases.html", context)
```

2) **Delete_database**: Esta vista contiene el mismo template que la anterior, es decir, es el mismo html. La diferencia es que cuando se pulsa el botón delete de una base de datos, se hace una petición POST desde la vista de databases a la vista delete_database, que es la que contiene la lógica de borrado y, además, carga el html como la vista anterior.

```
if request.method == "POST":
    filename = request.POST['filename']
    os.remove(os.path.abspath('media/'+filename))
```

3) **Upload**: Esta vista contiene la lógica de subida. Cuando se sube una base de datos se carga en el fichero media del proyecto y se refresca el template para que aparezca en el html.

```
if request.method == 'POST':
    myfile = request.FILES['upload']
    fs = FileSystemStorage()
    filename = fs.save(myfile.name, myfile)
    context['url'] = fs.url(filename)
```

4) **Edit**: Esta es una vista adicional que se ejecuta cada vez que se selecciona una base de datos para entrenar modelos. La funcionalidad de esta vista es preparar la base de datos para que se pueda utilizar correctamente en la aplicación de ML sin que haya problemas con valores no asignados.

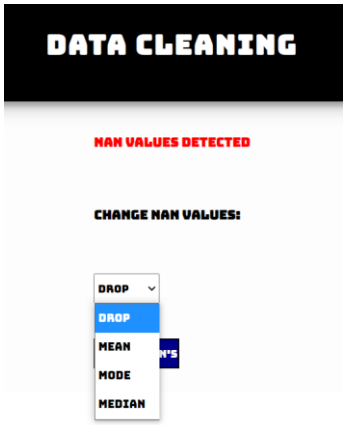


Ilustración 40 - Limpieza de datos

El usuario puede seleccionar entre 4 opciones distintas.

I. **Drop**: Consiste básicamente en eliminar las filas que contengan valores no asignados.
II. **Mean**: Los valores que no han sido asignados se sustituyen por la media de esa columna.
III. **Mode**: Los valores que no han sido asignados se sustituyen por la moda de esa columna.
IV. **Median**: Los valores no asignados se sustituyen por la media de esa columna.

Cuando el usuario postea la opción, en la vista se recoge del html el atributo 'nan' y luego se procede a realizar los cambios en la base de datos.

```
nan = str(request.POST['nan'])
context['message'] = 'changed'
nan_columns = df.columns[df.isna().any()].tolist()
context['count'] = df.isna().sum().sum()
if nan =="drop":
    df2 = df.dropna().reset_index(drop=True)
    df2.to_csv('media/'+filename, encoding='utf-8', index=False)
elif nan =="mean":
    for c in nan_columns:
        df[c].fillna(int(df[c].mean()), inplace=True)
    df.to_csv('media/' + filename, encoding='utf-8', index=False)
elif nan == "median":
    for c in nan_columns:
        df[c].fillna(int(df[c].median()), inplace=True)
    df.to_csv('media/' + filename, encoding='utf-8', index=False)
elif nan == "mode":
    for c in nan_columns:
        df[c].fillna(int(df[c].mode()), inplace=True)
    df.to_csv('media/' + filename, encoding='utf-8', index=False)
context['message'] = 'changed'
```

5.4 ML

En esta aplicación reside el núcleo del proyecto. Aquí es donde están programadas las distintas vistas para cada algoritmo y la generación de modelos predictivos.

En primer lugar, el template principal se carga una vez el usuario ha seleccionado una base de datos. Esto se hace mediante una petición POST y en el contexto se envía el nombre de la base de datos a utilizar. Esta se muestra para que el usuario sepa que base de datos está utilizando en ese momento.

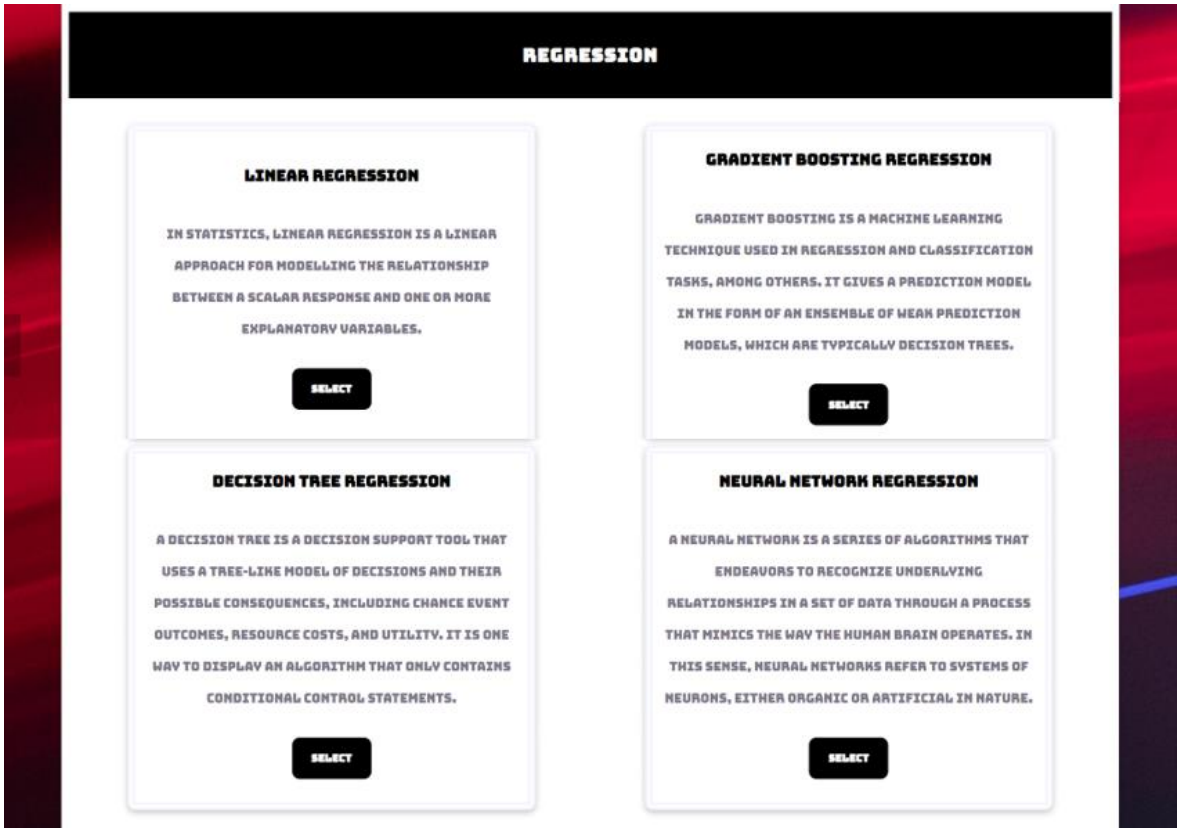


Ilustración 41 - Algoritmos de Regresión

Cada uno de los algoritmos tiene una vista con un template asociado. En total, 9 algoritmos y una vista principal.

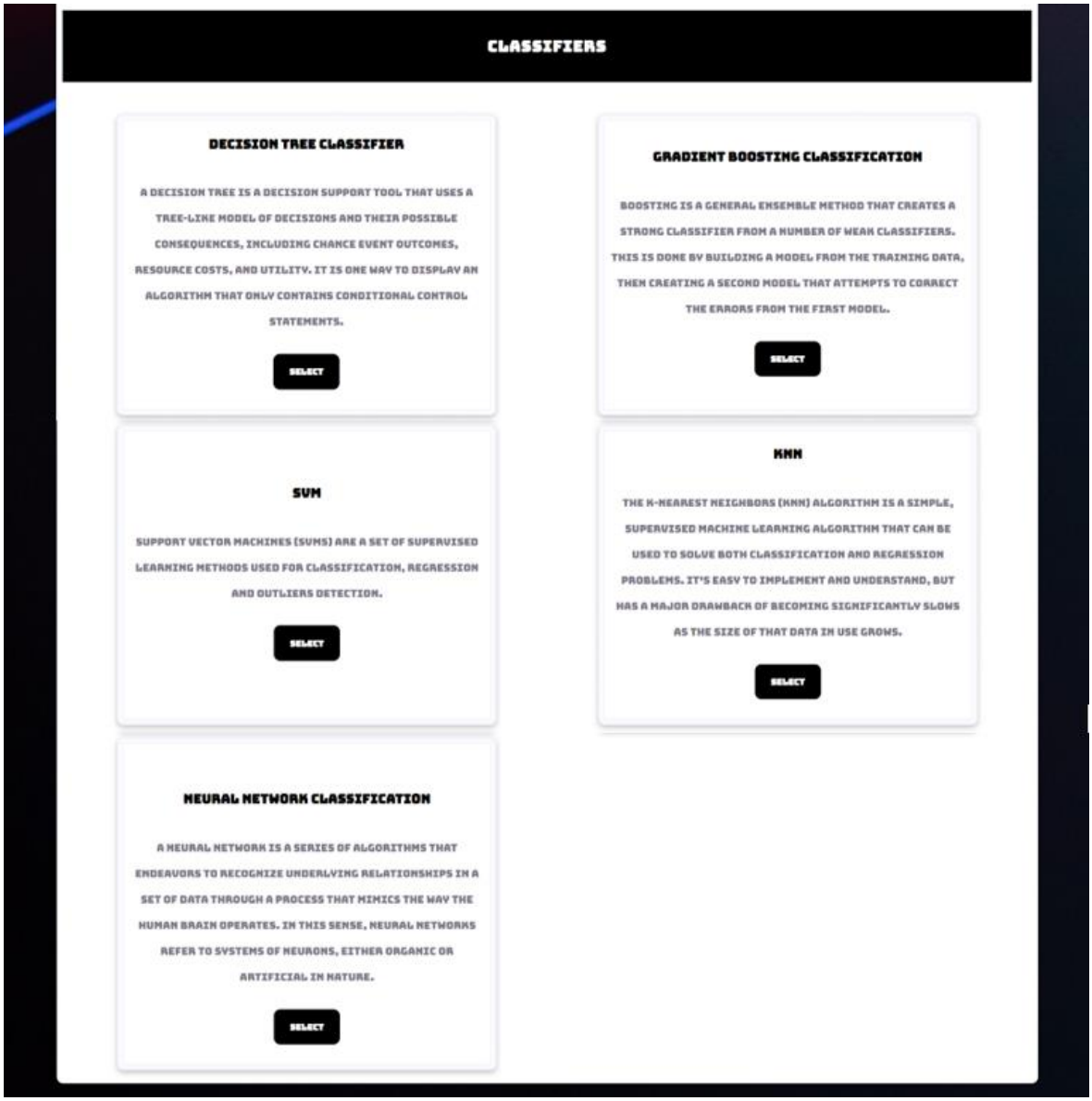


Ilustración 42 - Algoritmos de clasificación

Una vez el usuario selecciona un algoritmo, se carga su template correspondiente.

Al template de los algoritmos se les debe pasar el nombre de la base de datos para que desde la vista se puedan recoger las columnas y los datos almacenados para la creación del modelo.

5.4.1 RECOGIDA DE DATOS PARA ENTRENAR EL MODELO

Dentro de esa vista se necesitan recoger los siguientes datos para la creación de los modelos:

1) En primer lugar, necesitamos saber que datos quiere usar para realizar las predicciones y entrenar el modelo. Estas se recogen en el contexto de la view, que se pasará a la template una vez se cargue el html. En el contexto, las columnas de la base de datos vienen dada en forma de lista, para acceder individualmente a cada uno de los elementos desde el html, se puede utilizar una función for en el propio template.

   Vista:

```
context['columns'] = list(df.columns)
```

   Template:

```
{% for c in columns %}
    <div>
      <input type="hidden" name="{{c}}" checked>
      <input type="checkbox" id="{{c}}" value="selected"
name="{{c}}">
      <label for="{{c}}"> {{c}} </label>
    </div>
{% endfor %}
```

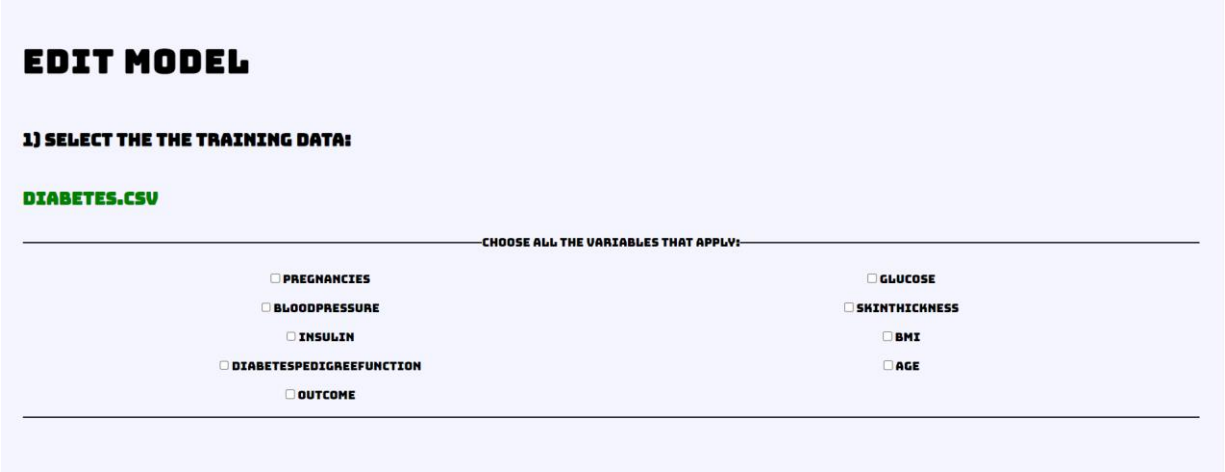


Ilustración 43 - Selección de columnas

2) Luego, con las mismas columnas, el usuario debe seleccionar el target, es decir, la variable que quiere predecir. El método de muestreo es parecido al paso 1) haciendo uso de un bucle for dentro del html.
3) En algunos casos, los algoritmos funcionan dividiendo los datos en datos para entrenar el modelo (train set) y datos para testear el modelo (test set). Estos datos se recogen desde el html también y se reciben en la view cuando se hace un POST.

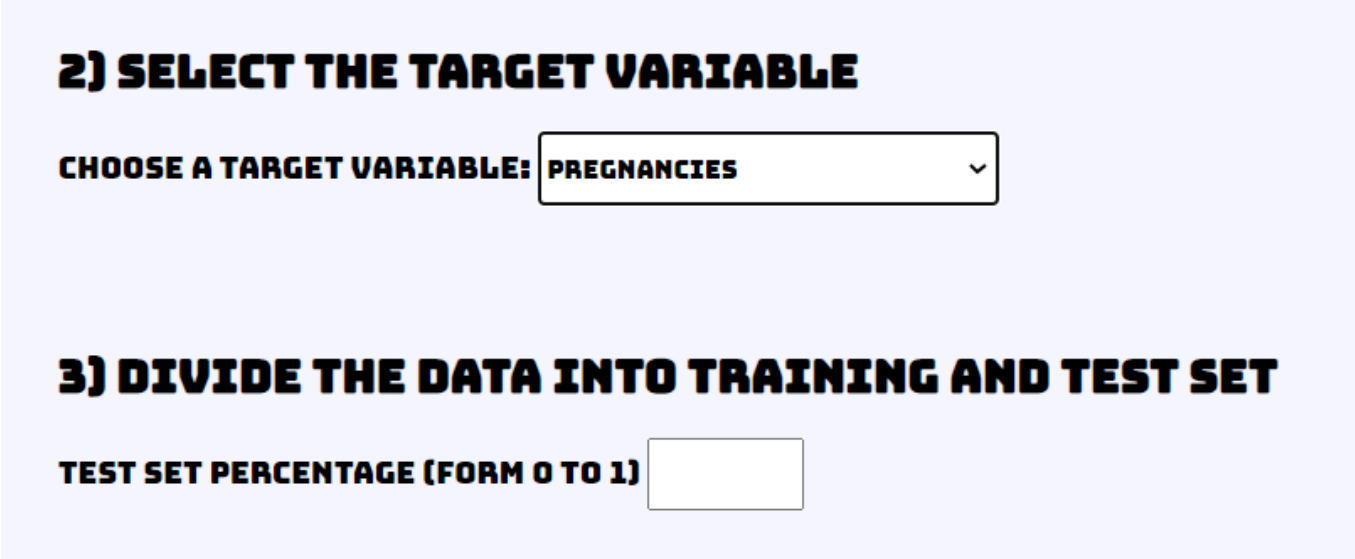


Ilustración 44 - Target y Ratio Test-Train

4) Dentro de cada algoritmo, existen unos hyper-parametros para personalizar aún más las características del modelo. En la aplicación se da opción a la modificación de esos hyper-parametros para conseguir una tasa de acierto mayor a la que conseguirías con los parámetros por defecto.

   De todas formas, esta selección es opcional y es recomendada únicamente a los usuarios más avanzados.

   En la siguiente captura se puede ver un ejemplo de los hyper-parametros disponibles para la generación de un modelo de KNN.

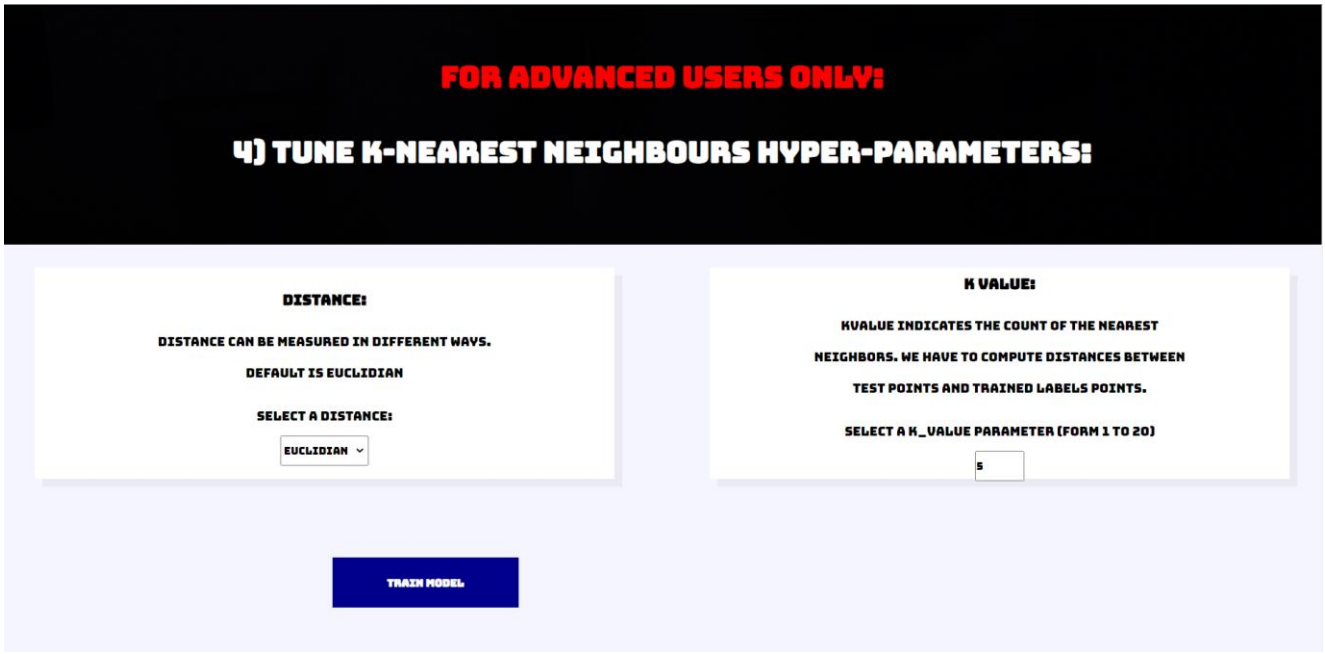


Ilustración 45 - Selección de Hyper-Parametros KNN

5.4.2 ENTRENAMIENTO DEL MODELO

Cuando el usuario ha seleccionado los datos para crear su modelo, este debe presionar el botón “Train Model”. Una vez pulsado, se realizará una petición POST a la misma vista, donde se creará el modelo con los datos indicados y se mostraran los resultados obtenidos.

1) Recogida de datos del POST:

```
test_ratio = float(request.POST["test_ratio"])
target = str(request.POST["target"])
data = list()
for c in context['columns']:
    s = str(request.POST[c])
    if s == "selected":
        data.append(c)
distance = str(request.POST["distance"])
n_neighbors = int(request.POST["n_neighbors"])
```

2) Preparación de los datos para la creación del modelo:

```
X = df[data]
y = df[target]
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=test_ratio)
```

3) Creación y entrenamiento del modelo:

```
model = KNeighborsClassifier(p=distance, n_neighbors=n_neighbors)
model.fit(X_train, y_train)
```

5.4.3 RESULTADOS

Una vez el modelo ha sido entrenado, se enseña por pantalla al usuario las características de su modelo, el ratio de aciertos, la precisión y las diferencias entre el set de datos de entrenamiento y el set de datos de testeo.

View:

```
train_yhat = model.predict(X_train)
context['train_acc'] = "%.2f" % accuracy_score(y_train,
train_yhat)
test_yhat = model.predict(X_test)
context['test_acc'] = "%.2f" % accuracy_score(y_test, test_yhat)
context['score'] = "%.2f" % model.score(X_test, y_test)
```

Template:

```
<h3>THE SCORE OBTAINED IS: {{score}}</h3>
<h3>TRAIN ACCURACY: {{train_acc}}</h3>
<h3>TEST ACCURACY: {{test_acc}}</h3>
```

Además de mostrar los resultados por pantalla, también se muestran diferentes graficas que aportan información adicional.

Matriz de confusión:

```
def plotConfusion(model,X,y, target):
    plot_confusion_matrix(model, X, y)
    plt.title('Confusion Matrix of ' + str(target))
    plt.ylabel('True label')
    plt.xlabel('Predicted Label')
    fig = plt.gcf()
    plt.close()
    imgdata = StringIO()
    fig.savefig(imgdata, format='svg')
    imgdata.seek(0)

    return imgdata.getvalue()
```

Regiones de clasificación:

```
def plotClassifier(model,X,y):
    _, ax = plt.subplots()
    DecisionBoundaryDisplay.from_estimator(model, X,
cmap=plt.cm.Paired,       ax=ax)
    sns.scatterplot(x=X.iloc[:, 0], y=X.iloc[:, 1],
palette=plt.cm.Set3, c=y.to_numpy(), alpha=1.0, edgecolor="black")
    plt.xlabel(str(list(X.columns)[0]))
    plt.ylabel(str(list(X.columns)[1]))
    plt.title("Classification Regions")
    fig = plt.gcf()
    plt.close()
    imgdata = StringIO()
    fig.savefig(imgdata, format='svg')
    imgdata.seek(0)
    return imgdata.getvalue()
```

Árbol de decisión:

```
def plot_dtc(model):
    # tree.plot_tree(model)
    fig = matplotlib.pyplot.gcf()
    fig.set_size_inches(18.5, 10.5, forward=True)
    _ = tree.plot_tree(model, filled=True)
    _ = plt.gcf()
    plt.close()
    imgdata = StringIO()
    fig.savefig(imgdata, format='svg')
    imgdata.seek(0)

    return imgdata.getvalue()
```

Regresión lineal:

```
def lr_plot(X,y,target,model):
    plt.scatter(X, y, color='green', marker='+')
    line = model.predict(X)
    plt.plot(X, line, color="red")
    plt.xlabel(str(list(X.columns)[0]))
    plt.ylabel(str(target))
    plt.title('LINEAR REGRESION')
    fig = plt.gcf()
    plt.close()
    imgdata = StringIO()
    fig.savefig(imgdata, format='svg')
    imgdata.seek(0)

    return imgdata.getvalue()
```

Importancia de las variables:

Esta grafica se muestra tan solo en los algoritmos que usan árboles de decisión, es decir, en los árboles de regresión y clasificación y en el gradient boosting. Básicamente se mide cuanta varianza es capaz de eliminar cada variable, es decir, que variable consigue particionar mejor los datos.

```
def plot_variable_importance(fi,columns):
    sorted_idx = np.argsort(fi)
```

```
    pos = np.arange(sorted_idx.shape[0]) + 0.5
    plt.barh(pos, fi[sorted_idx], align="center")
    plt.yticks(pos, np.array(columns)[sorted_idx])
    plt.title("Feature Importance")
    fig = plt.gcf()
    plt.close()
    imgdata = StringIO()
    fig.savefig(imgdata, format='svg')
    imgdata.seek(0)
    return imgdata.getvalue()
```

Curva de perdidas en redes neuronales:

La función de perdidas muestra cuanto se equivoca la red neuronal en cada iteración. Normalmente, en estas graficas se observa como a medida que se va entrenando el modelo, la red neuronal aprende y la función de perdidas baja. Al principio lo hace rápido, pero llega un momento que el aprendizaje es más lento.

```
def plotLossCurve(model):
    plt.plot(model.loss_curve_)
    plt.title("Loss Curve", fontsize=14)
    plt.xlabel('Iterations')
    plt.ylabel('Cost')
    fig = plt.gcf()
    plt.close()
    imgdata = StringIO()
    fig.savefig(imgdata, format='svg')
    imgdata.seek(0)

    return imgdata.getvalue()
```

Predicciones en regresiones:

Esta grafica muestra un set de 30 datos con las predicciones realizadas por el modelo, y el dato real.

```
def plotPrediction(df_temp):
    df_temp = df_temp.head(30)
    df_temp.plot(kind='bar', figsize=(10, 6))
    plt.grid(which='major', linestyle='-', linewidth='0.5',
color='green')
```

```
    plt.grid(which='minor', linestyle=':', linewidth='0.5',
color='black')
    fig = plt.gcf()
    plt.close()
    imgdata = StringIO()
    fig.savefig(imgdata, format='svg')
    imgdata.seek(0)

    return imgdata.getvalue()
```

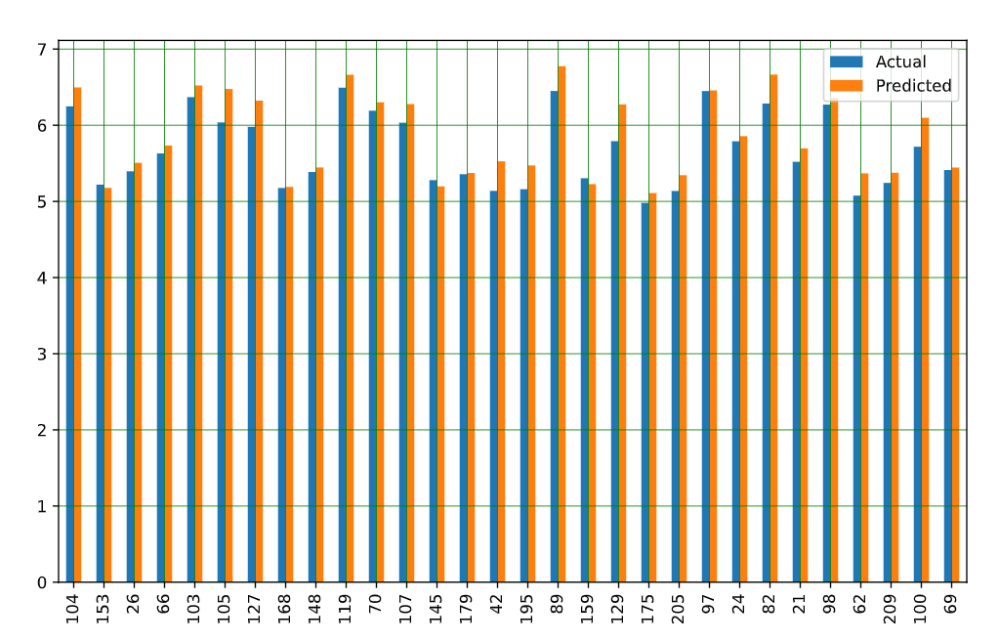


Ilustración 46 - Grafica de comparación Realidad-Predicción

Es importante que las variables se guarden como una imgdata para que se puedan mandar de la vista al template y que se puedan renderizar en la página web. Para referenciar las gráficas en el template hay que hacerlo de la siguiente manera:

```
<div class="images_result">
    {{ plot|safe }}
</div>
```

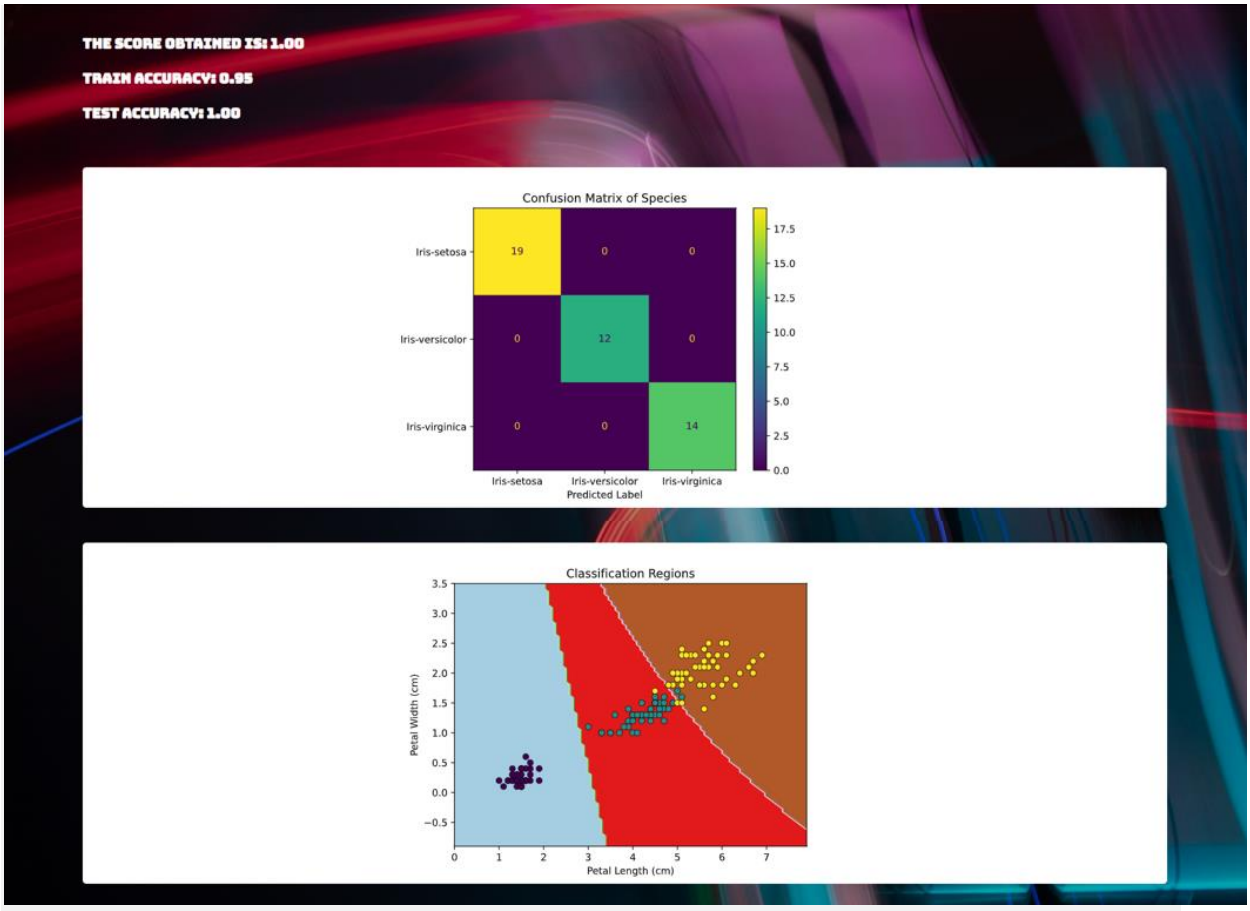


Ilustración 47 – Ejemplo de output de un modelo entrenado

5.5 MODELOS

Finalmente, en la aplicación de modelos se gestiona el grabado de modelos y la pantalla para realizar predicciones.

La generación de modelos de la aplicación ML solamente es útil si podemos utilizar esos modelos. Una vez tenemos un modelo entrenado, podemos guardarlo en el proyecto y acceder al cuándo queramos para realizar predicciones.

Existe una librería de Python llamada joblib, que se encarga de facilitar el guardado de modelos.

```
joblib.dump(model, 'Models/media/Models/'+modelname)
```

El problema de grabado de modelos es que una vez entrenamos un modelo y se reciben los datos del POST con las características, estos datos se pierden ya que se vuelve a cargar de nuevo el template con la misma vista. Una forma de solucionar este problema es con ayuda de un recurso que ofrece Django llamado `request.session[]`

Una sesión de Django no es más que un mecanismo para guardar información en el servidor durante la interacción que tiene un usuario con la página web.

Para almacenar solo los modelos que el usuario considere se hace lo siguiente. Una vez el usuario ha seleccionado las características de su modelo, en el `request.session[]` se guardan estas características. Aquí no se puede guardar directamente el fichero ni el objeto de tipo modelo, solo se pueden guardar objetos de Python.

```
save_dict['data'] = data
save_dict['target'] = target
save_dict['test_ratio'] = test_ratio
save_dict['filename'] = filename
save_dict['algorithm'] = "knn"
save_dict['distance'] = distance
save_dict['n_neighbors'] = n_neighbors
request.session['_my_data'] = save_dict
```

Siempre que se crea un modelo, se actualiza el request session con los nuevos datos del modelo.

Una vez el usuario decide que su modelo es bueno y quiere guardarlo, tiene que indicar un nombre y presionar el botón “Save Model”.

Ilustración 48 - Botón de guardado de modelo

Hasta el momento que se pulsa el botón seguimos en la aplicación de ML, pero una vez pulsado, se carga la vista de predicciones de la aplicación de modelos.

Ilustración 49 - Vista de predicciones

Una vez pulsado el boton, se carga un formulario con las variables utilizadas para entrenar el modelo. Es ahí donde se deben poner los datos para generar la predicción. Con los datos ya añadidos, se presiona el botón de “Make Prediction” y aparecerá la clasificación predicha.

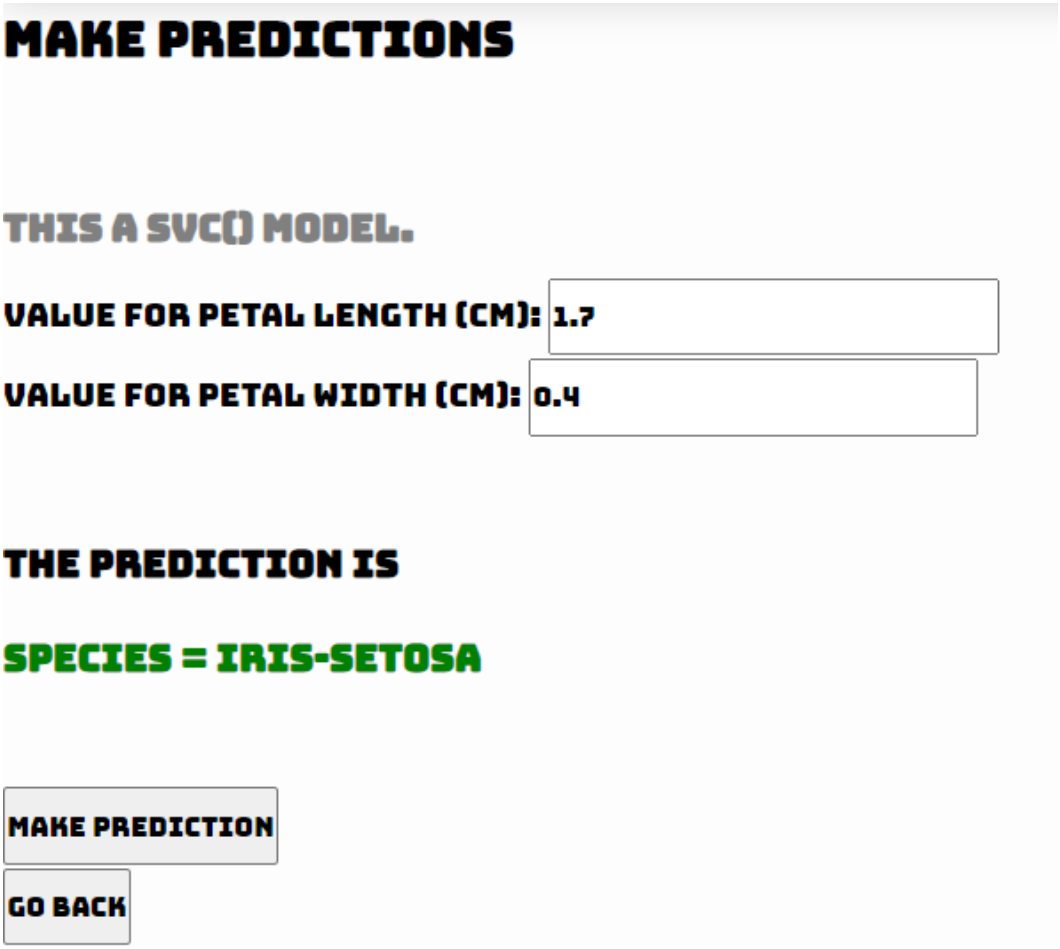


Ilustración 50 - Predicción generada

Por último, desde la aplicación principal se puede acceder a todos los modelos guardados en el servidor para poder hacer predicciones de manera sencilla y sin tener que reentrenar ningún modelo.

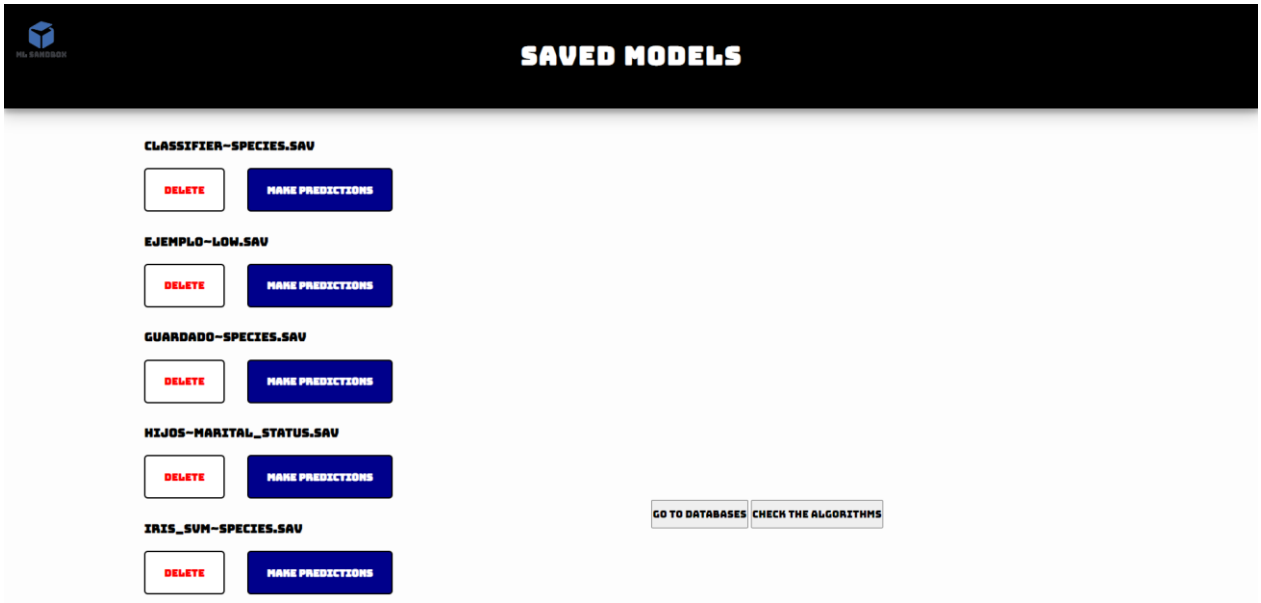


Ilustración 51 - Pantalla de Modelos Guardados

Capítulo 6. TRABAJOS FUTUROS Y CONCLUSIÓN

A pesar de que la aplicación desarrollada es funcional, para crear una herramienta que sea capaz de competir en el mercado y presente una gran utilidad a sus usuarios queda tiempo de trabajo.

En este capítulo se analizará y criticará el trabajo desarrollado con el fin de mostrar las distintas mejoras que se pueden incorporar a la app y que esto se traduzca en futuros trabajos.

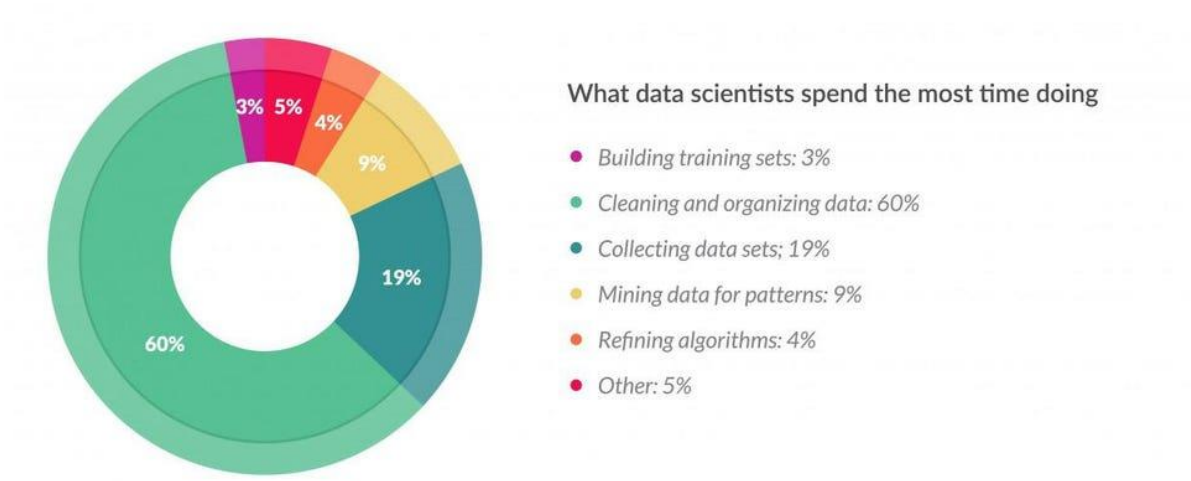
6.1 EL PROBLEMA DE LOS DATOS

Uno de los problemas que tiene esta herramienta es que, para que funcione, los datos tienen que venir en formato .csv y bien ordenados.

En el mundo de los científicos de datos, es bien sabido que la mayor parte del trabajo que se realiza va destinado a pre-procesar los datos. Es decir, que, para obtener modelos predictivos con altas tasas de acierto, es casi indispensable que los datos que se le den al modelo para entrenarlo estén limpios, normalizados y tuneados de la mejor forma posible.

La herramienta desarrollada en este proyecto no hace eso, la única limpieza de datos que hace es remplazar y sustituir los valores no asignados (NaNs) por la media, mediana o moda de esa columna.

En la siguiente grafica se puede observar el tiempo que dedican los científicos de datos a distintas tareas y más del 80% tiene que ver con la limpieza, organización y recogida de datos.



16

Ilustración 52 - Tareas de los científicos de datos

Por lo tanto, es necesario incluir una robusta capa de preprocesado y ordenado de datos.

Existen tres grandes categorías de preprocesado de datos:

- **Limpieza:** Consiste en la gestión de los datos vacíos, como los NaNs y en la limpieza de los datos ruidosos. Se pueden implementar varias técnicas como eliminación de outliers y clustering.
- **Transformación:** La transformación de datos consiste en aplicar operaciones matemáticas para facilitar a los modelos las distintas operaciones que tiene que realizar. Esto se puede conseguir con normalizaciones, discretizaciones y con selección de atributos.
- **Reducción:** Por último, reducir la cantidad de datos descubriendo redundancias entre ellos. Reducción de dimensionalidad, selección de un subset de atributos, análisis de las componentes principales (PCA).

[16] Imagen obtenida de la revista Forbes: https://www.forbes.com/sites/gilpress/2016/03/23/data-preparation-most-time-consuming-least-enjoyable-data-science-task-survey-says/

6.2 EL PROBLEMA DE LA SELECCIÓN CORRECTA DE HYPER-PARAMETROS

Otro de los problemas de esta herramienta es que al final, para crear un modelo predictivo con una alta tasa de aciertos, hay que conocer bien los hyper-parametros y hacer varias pruebas modificando sus valores.

El objetivo de esta herramienta es que no se necesite ningún conocimiento técnico para usarla. Básicamente el futuro de la herramienta sería una plataforma donde subes un batiburrillo de datos, y esta herramienta los ordena y limpia de manera automática y aparte, te selecciona los mejores modelos con los mejores hyper-parametros.

Ahora mismo la selección de hyper-parametros es algo tediosa y como marcamos en la propia aplicación, recomendamos que solo los expertos cambien sus valores.

El trabajo que se debe de realizar en este sentido es, crear un seleccionador de hyper-parametros automático. Esto se puede hacer entrenando distintos modelos con distintos valores en sus parámetros y escogiendo los modelos con mejor tasa de acierto. Comparando siempre las tasas de acierto de train y de test y comprobando que el modelo no esté sobre entrenado.

En el estado del arte actual del Machine Learning existen varios métodos para realizar esta tarea. Un artículo muy interesante de la revista medim explica alguna de estas técnicas.

"Increasingly, hyperparameter tuning is done by automated methods that aim to find optimal hyperparameters in less time using an **informed search** with no manual effort necessary beyond the initial set-up"

(Koehersen, 2018)

6.3 CONCLUSIÓN

Está claro que la IA está revolucionando y cambiando el paradigma del mundo actual. Cada día son más las empresas que quieren entrar en el mercado del MLaaS debido a su pronóstico de crecimiento. No hay un momento mejor para desarrollar herramientas como está.

Durante el desarrollo de esta aplicación se ha conseguido cumplir con los objetivos marcados. La aplicación es capaz de generar y entrenar modelos al gusto del usuario utilizando sus propias bases de datos. Además, es capaz de almacenar ese modelo y facilitar la realización de predicciones sobre el mismo. También, esta aplicación colabora con el desarrollo de la agenda 2030, ayudando a solucionar algunos de los problemas marcados en los ODS.

Aun así, nos encontramos en una industria muy competitiva ya que las grandes empresas son conocedoras de esta enorme oportunidad en el mercado y están actualmente dedicando todos sus recursos para aumentar su cuota. Es decir, que la competición en este sector es muy alta.

También, hemos analizado dos de los grandes problemas que no se han resuelto en el desarrollo de nuestra aplicación y que son cruciales para tener un programa competitivo y útil en la vida real. De todas formas, durante la realización del proyecto se han dado firmes pasos hacia ese destino con una versión beta funcional.

Para terminar, quiero remarcar el estado del arte actual de la IA con proyectos elaborados por DeepMind como Alpha Fold, capaz de predecir el patrón de doblado de las proteínas recién ensambladas. El modelo de programación del lenguaje natural GPT3, una IA capaz de mantener conversaciones coherentes con los humanos que ha conseguido que hasta sus desarrolladores se pregunten si esa IA tiene sentimientos. Y, por último, mi preferido, el modelo DALLE-2, una IA capaz de crear imágenes dado un texto de input.

Hace unos días conseguí acceso a este modelo, y dejo como final del proyecto una imagen generada por el mismo.

"Una IA contemplando la tierra desde la luna"

Ilustración 53 - Imagen auto-generada con DALL-E 2

Capítulo 7. BIBLIOGRAFÍA

[1] BBVA. (11 de Noviembre de 2019). *Machine learning: What is it and how does it work?* Obtenido de https://www.bbva.com/en/machine-learning-what-is-it-and-how-does-it-work/

[2] Bommana, H. (26 de 05 de 2019). *Introduction to Neural Networks*. Obtenido de https://medium.com/deep-learning-demystified/introduction-to-neural-networks-part-1-e13f132c6d7e

[3] Brooks, H. (s.f.). *Architecture for a generalised regression neural network (GRNN)*. Obtenido de https://www.researchgate.net/figure/Architecture-for-a-generalised-regression-neural-network-GRNN-The-input-layer-where_fig2_270274130

[4] Chart, L. (s.f.). Obtenido de https://lucid.app/lucidchart/

[5] Contributors, M. (13 de Agosto de 2022). *Django introduction*. Obtenido de https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction

[6] Django. (s.f.). *Django documentation*. Obtenido de https://docs.djangoproject.com/en/4.1/

[7] Education, I. C. (19 de 08 de 2020). *Supervised Learning*. Obtenido de https://www.ibm.com/cloud/learn/supervised-learning#:~:text=Supervised%20learning%2C%20also%20known%20as,data%20or%20predict%20outcomes%20accurately.

[8] España, B. (29 de 10 de 2020). *Qué son regresión y clasificación en Machine Learning*. Obtenido de https://agenciab12.com/noticia/que-son-regresion-clasificacion-machine-learning#:~:text=Algoritmos%20de%20regresi%C3%B3n,-

El%20an%C3%A1lisis%20de&text=%C2%BFQu%C3%A9%20significa%20este%20concepto%3F,de%20un%20conjunto%20de%20datos.

[9] Fraj, M. B. (24 de 12 de 2017). *In Depth: Parameter tuning for Gradient Boosting*. Obtenido de https://medium.com/all-things-ai/in-depth-parameter-tuning-for-gradient-boosting-3363992e9bae

[10] Fraj, M. B. (20 de 12 de 2017). *InDepth: Parameter tuning for Decision Tree*. Obtenido de https://medium.com/@mohtedibf/indepth-parameter-tuning-for-decision-tree-6753118a03c3

[11] Fuchs, M. (30 de 06 de 2019). *Metrics for Regression Analysis*. Obtenido de https://michael-fuchs-python.netlify.app/2019/06/30/metrics-for-regression-analysis/#summary-of-the-metrics

[12] Infograph. (s.f.). Obtenido de https://infograph.venngage.com/

[13] ITx, C. (7 de 08 de 2019). *What is Machine learning as a service (MLaaS)?* Obtenido de https://coreitx.com/blogs/what-is-machine-learning-as-a-service-mlaas#:~:text=What%20is%20MLaaS%3F,predictive%20analytics%20and%20deep%20learning.

[14] Johari, A. (04 de 12 de 2017). *A Beginner's Guide To Scikit Learn*. Obtenido de https://medium.com/edureka/scikit-learn-machine-learning-7a2d92e4dd07

[15] Koehersen, W. (3 de Julio de 2018). *Automated Machine Learning Hyperparameter Tuning in Python*. Obtenido de https://towardsdatascience.com/automated-machine-learning-hyperparameter-tuning-in-python-dfda59b72f8a#8811

[16] Krishnan, S. (26 de 08 de 2021). *Decision Tree for Classification, Entropy, and Information Gain*. Obtenido de https://medium.com/codex/decision-tree-for-classification-entropy-and-information-gain-cd9f99a26e0d

[17] Nations, U. (s.f.). *Infrastructura*. Obtenido de https://www.un.org/sustainabledevelopment/es/infrastructure/

[18] Nations, U. (s.f.). *Objetivos de desarrollo sostenible*. Obtenido de https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/

[19] Orza, P. (26 de 07 de 2022). *MLaaS Platforms: The Comparative Guide*. Obtenido de https://levity.ai/blog/mlaas-platforms-comparative-guide

[20] Pro, P. (22 de 07 de 2022). *Why data preparation is an important part of data science?* Obtenido de https://www.projectpro.io/article/why-data-preparation-is-an-important-part-of-data-science/242#:~:text=Steve%20Lohr%20of%20The%20New,be%20explored%20for%20useful%20nuggets.%22

[21] Rojo, O. (21 de 07 de 2020). *Machine Learning Algorithms With Scikit-Learn*. Obtenido de https://medium.com/swlh/machine-learning-algorithns-with-scikit-learn-ec6181365ba

[22] Tavasoli, S. (11 de 08 de 2022). *Top 10 Machine Learning Algorithms*. Obtenido de https://www.simplilearn.com/10-algorithms-machine-learning-engineers-need-to-know-article

[23] Urban, T. (22 de Enero de 2015). *Wait but why*. Obtenido de The AI Revolution: The Road to Superintelligence: https://waitbutwhy.com/2015/01/artificial-intelligence-revolution-1.html

[24] Villacís, D. (s.f.). *Cascada vs Scrum*. Obtenido de https://consulting-systems.tech/cascada-vs-scrum/

ANEXO I: ALINEACIÓN DEL PROYECTO CON LOS ODS

El 25 de septiembre de 2015, los líderes mundiales adoptaron un conjunto de objetivos globales para erradicar la pobreza, proteger el planeta y asegurar la prosperidad para todos como parte de una nueva agenda de desarrollo sostenible. En total, se propusieron 17 objetivos con metas específicas que deben alcanzarse para el año 2030.

Los objetivos son los siguientes:

Ilustración 54 - Objetivos de desarrollo sostenible [17]

Gobiernos, entidades privadas, la sociedad civil y autónomos colaboran para cumplir con las estas metas, y, como no podía ser de otra manera, este proyecto está también alineado con varios de los objetivos propuestos para la agenda 2030.

Los ODS se dividen en 5 categorías, denominadas las ‘5 P’ por sus nombres en inglés; Prosperidad (Prosperity), Planeta (Planet), Personas (People), Paz (Peace) y Cooperación (Partnership).

[17] Imagen obtenida de la pagina oficial de las Naciones Unidas. https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/

Este proyecto, se centra en la 'P' de prosperidad. En concreto, en los objetivos 8, 9 y 12.

8) Trabajo decente y crecimiento económico:

De este objetivo, descienden 12 subobjetivos. Este proyecto hace hincapié en dos de esos subobjetivos. El primero es el 8.2 que consiste en la diversificación, tecnología e innovación. El segundo es el 8.3, que revindica un fomento a la pequeña y mediana empresa.

Como se ha hablado en capítulos anteriores, la herramienta desarrollada en este proyecto por un lado es innovadora y utiliza una tecnología avanzada que todavía esta en su fase de instauración en la vida cotidiana. Por otro lado, el objetivo principal del proyecto era acercar estas herramientas a las pequeñas y medianas empresas, incluso a usuarios autónomos que necesiten de un software simple y fácil de usar como este. Es por ello por lo que se pone en foco a estos pequeños empresarios dándoles soporte a que se produzcan innovaciones en sus negocios.

9) Industria innovación e infraestructura:

Al igual que ocurre con el objetivo 8, del objetivo 9 también surgen 8 distintos subobjetivos.

El objetivo 9.5 entre otras cosas, se centra en mejorar la capacidad tecnológica de los sectores industriales de todos los países. Propone conseguir esto a través de la investigación y de la innovación. Esta herramienta, tiene como objetivo mejorar la capacidad tecnológica en todos los distintos sectores, ya que

puede utilizarse en cualquiera.

Por otro lado, el objetivo 9.B, también defiende el apoyo al desarrollo de las tecnologías, la investigación y la innovación, para que se produzca una adición de valor a los productos básicos, entre otras cosas. Una herramienta universal, que se pueda utilizar en cualquier set de datos, aporta un valor añadido a cualquier negocio que recoja datos para analizarlos y sacar conclusiones de ellos.

12) Producción y consumo responsables:

Finalmente, podríamos argumentar que una herramienta como la desarrollada en este proyecto, fomenta una producción de recursos responsable. Usando los algoritmos predictivos incluidos en esta herramienta, se pueden realizar predicciones basándose en datos de demanda y producción pasados. Esto puede permitir un ahorro de recursos a los negocios y, por otro lado, un aumento de la producción responsable ya que se reducen los desperdicios (Objetivo 12.3) y, además, se hace un uso más eficiente de los recursos naturales (Objetivo 12.2).
Por otro lado, esta herramienta colabora en la creación de un marco de producción sostenible (Objetivo 12.1).

Los ODS se escribieron para buscar respuestas a las mayores amenazas y a los problemas más complejos que tiene la humanidad. Trabajando juntos en la resolución de estos problemas no solo contribuimos para mejorar nuestras vidas, si no que contribuimos para mejorar la de nuestros hijos y nietos.