UNIVERSIDAD DE EXTREMADURA

Escuela Politécnica

Grado en Ingeniería Informática en Ingeniería de Software

Trabajo Fin de Grado

Modelos predictivos aplicados a IoT

Autor: José Luis Pérez García

Tutor: Andrés Caro Lindo

Co-Tutor/es: José Carlos Sancho Núñez

**Tribunal Calificador**

Presidente: <Nombre y Apellidos>

Secretario: <Nombre y Apellidos>

Vocal: <Nombre y Apellidos>

**RESUMEN**

**ÍNDICE GENERAL DE CONTENIDOS**

[1 INTRODUCCIÓN 6](#_Toc107138966)

[2 OBJETIVOS 7](#_Toc107138967)

[3 ESTADO DEL ARTE 8](#_Toc107138968)

[3.1 MACHINE LEARNING 8](#_Toc107138969)

[3.2 DISPOSITIVOS IoT 11](#_Toc107138970)

[3.3 ANTECEDENTES DE MACHINE LEARNING APLICADO A IOT 13](#_Toc107138971)

[Caso 1 13](#_Toc107138972)

[Caso 2 15](#_Toc107138973)

[Caso 3 17](#_Toc107138974)

[Caso 4 18](#_Toc107138975)

[4 METODOLOGÍA 20](#_Toc107138976)

[5 IMPLEMENTACIÓN Y DESARROLLO 23](#_Toc107138977)

[5.1 Recolección de datos 23](#_Toc107138978)

[5.2 Selección de Features o atributos para el modelo de entrenamiento 26](#_Toc107138979)

[5.3 Conjunto de entrenamiento y prueba 27](#_Toc107138980)

[5.4 Algoritmos implementados 28](#_Toc107138981)

[5.5 Inicialización del algoritmo 38](#_Toc107138982)

[5.6 Entrenamiento del modelo 39](#_Toc107138983)

[5.7 Predicción de datos 40](#_Toc107138984)

[5.8 Creación de la base de datos 41](#_Toc107138985)

[5.9 Creación del Front y el Back 42](#_Toc107138986)

[5.10 Representación de los datos 43](#_Toc107138987)

[6 RESULTADOS Y DISCUSIÓN 46](#_Toc107138988)

[7 CONCLUSIONES 54](#_Toc107138989)

[8 REFERENCIAS BIBLIOGRÁFICAS 55](#_Toc107138990)

[Anexo 1: Otra funcionalidad del entorno web 57](#_Toc107138991)

[Anexo 2: Instalación del entorno 58](#_Toc107138992)

[Anexo 3: Librerías usadas 61](#_Toc107138993)

**ÍNDICE DE FIGURAS**

[Figura 1: Metodología usada en el caso 1 14](#_Toc107138963)

[Figura 2: Metodología del proyecto 20](#_Toc107138964)

[Figura 3: Ejemplo funcionamiento Random Forest 30](#_Toc107138965)

**ÍNDICE DE ILUSTRACIONES**

[Ilustración 1: Resultados RMSE y MAE del Caso 2 (5) 15](#_Toc107138950)

[Ilustración 2: Resultados R2 del Caso 2 (5) 16](#_Toc107138951)

[Ilustración 3: Resultados MSE, RMSE, MAE y RMAE del Caso 3 17](#_Toc107138952)

[Ilustración 4: Metodología del caso 4 18](#_Toc107138953)

[Ilustración 5: Representación de predicción usando XGBoost 22](#_Toc107138954)

[Ilustración 6: Fragmento de código del proceso de Web Scrapping 23](#_Toc107138955)

[Ilustración 7: Tabla de datos de la web Weather Underground (9) 24](#_Toc107138956)

[Ilustración 8; Fragmento de datos del csv 25](#_Toc107138957)

[Ilustración 9: Fragmento de código de la lectura del CSV 25](#_Toc107138958)

[Ilustración 10: Fragmento de código de la selección de los Features 26](#_Toc107138959)

[Ilustración 11: Añadiendo librería para crear conjunto de entrenamiento y prueba 27](#_Toc107138960)

[Ilustración 12: Ejemplo línea y puntos Regresión Lineal 29](#_Toc107138961)

[Ilustración 13: Ejemplo funcionamiento Decision Tree Regression 36](#_Toc107138962)

**ÍNDICE DE TABLAS**

[Tabla 1: Modelos y algoritmos usados en el caso 1 13](#_Toc107138926)

[Tabla 2: Tabla de predicción para conjunto de test 40](#_Toc107138927)

# INTRODUCCIÓN

# OBJETIVOS

# ESTADO DEL ARTE

Para proceder a realizar un análisis del proyecto realizado, previamente se va a introducir en que consiste el Machine Learning y en que consiste IoT. Posteriormente, se hará un resumen de otros estudios que se han realizado anteriormente en cuanto al trabajo conjunto de ambos.

## MACHINE LEARNING

Actualmente, se pueden encontrar muchos ejemplos en Machine Learning en el día a día, pero se desconoce que lo es, por ejemplo, coches que conducen solos (son ya muy famosos los coches de la marca Tesla), asistentes que traducen instantáneamente de un idioma a otro (la cámara en el traductor de Google) o sugerencias de compra personalizadas (por ejemplo el mail que nos llega de Amazon con el mensaje “Te podría interesar”).

**¿Qué es el Machine Learning?**

El Machine Learning o Aprendizaje Automático se conoce por ser una disciplina del campo de la Inteligencia Artificial que, mediante el uso de algoritmos, proporciona a los ordenadores la capacidad de poder identificar patrones en grandes conjuntos de datos y con ello poder elaborar predicciones. A esto también se le conoce como análisis predictivo. Este aprendizaje dota a los ordenadores la posibilidad de poder realizar tareas específicas de forma autónoma. Así, de esta manera, no sería necesario que un usuario los programe. (1)

El término Machine Learning fue usado por primera vez en el 1959. No obstante, hasta estos últimos años no se ha empezado a escuchar de verdad. Esto se ha debido al gran aumento en la capacidad de computación y al uso de los conjuntos masivos de datos. De hecho, en el mundo del Big Data el Machine Learning es una parte fundamental.

**¿Cómo funciona el Machine Learning?**

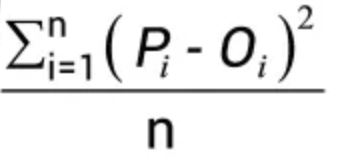
El funcionamiento del Machine Learning se puede resumir de la siguiente manera: el usuario recolecta una serie de datos conocidos como input y partir de estos el ordenador debe ser capaz de encontrar patrones o relaciones entre ellos. Lo que hace el ordenador, una vez tiene el conjunto de datos, es escoger una fórmula matemática que se aplica al conjunto de datos o input de entrenamiento, el cual también pertenece a otro conjunto de datos y devuelve los outputs deseados. Lo realmente interesante es que la misma fórmula puede ser aplicable a otros datos distintos del conjunto de entrenamiento teniendo en cuenta que los nuevos tienen que ser parecidos a los datos de entrenamiento.(2)

El principal objetivo que debe buscar el Machine Learning es la optimización. Nuestra máquina debe ser capaz de escoger el algoritmo que mejor se ajuste y para ello debe elegir la fórmula que relacione los datos y minimice el error, el cual debe ser medido según diferentes métricas dependiendo de las características del modelo.

Las métricas más usadas para medir la efectividad de la predicción de un modelo son: (3)

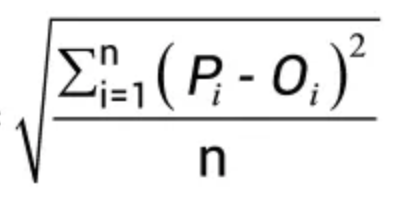
-MSE/RMSE: mide el error cuadrado promedio de las predicciones del modelo, es decir, para cada valor predicho y real, la diferencia al cuadrado de esos puntos entre el número total de valores. Cuanto mayor sea este valor, peor es el modelo. La diferencia entre RMSE y MSE es que la primera es la misma fórmula pero con raíz cuadrada.

La fórmula del MSE es:



Donde Pi son los valores predichos, Oi los valores reales y n el número de muestras totales.

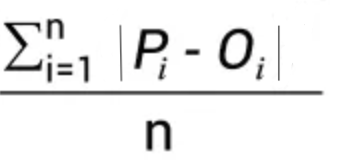
La fórmula del RMSE es:



Al igual que en el MSE, Pi son los valores predichos, Oi los valores reales y n el número de muestras totales.

-MAE: mide la diferencia absoluta que hay entre dos variables continuas. Sirve para medir la precisión de un cálculo de predicción comparando los valores predichos frente a los reales, es decir, por ejemplo la temperatura real frente a la temperatura prevista.

La fórmula del MAE es:



Al igual que en el MSE y RMSE, Pi son los valores predichos, Oi los valores reales y n el número de muestras totales.

-R^2: está relacionada con el MSE. No importa si los valores de salida son muy grandes o muy pequeños, el R² siempre estará entre -∞ y 1. Cuando R² es negativo, significa que el modelo es peor que predecir la media.

**Algoritmos de Machine Learning**

Los algoritmos de Machine Learning se dividen en tres categorías, siendo las dos primeras las más usadas:

-Aprendizaje supervisado: este tipo de algoritmos necesitan un aprendizaje previo, que está basado en un sistema de etiquetas asociadas a datos, que permiten poder tomar decisiones o hacer predicciones. Por ejemplo, un detector de spam que se usa para etiquetar un mail como no deseado, hace uso de patrones que ha aprendido del histórico de correos (remitente, relación entre texto e imágenes, palabras clave en el asunto, etc.).

-Aprendizaje no supervisado: estos algoritmos no requieren de un conocimiento previo. Analizan los datos con el objetivo de encontrar patrones que permitan organizarlos de alguna forma. Por ejemplo, en marketing es muy usado para extraer patrones de grandes conjuntos de datos que provienen de las redes sociales y así poder crear campañas de publicidad altamente segmentadas.

-Aprendizaje por refuerzo: el objetivo de este tercer tipo de algoritmos es que este aprenda de la experiencia, de forma que sea capaz de tomar la mejor decisión ante diferentes situaciones mediante un proceso de prueba y error en el que se recompensan las decisiones correctas. Por ejemplo, en la actualidad lo se pueden encontrar en reconocedores faciales, clasificación de secuencias de ADN o diagnósticos médicos.

## DISPOSITIVOS IoT

IoT, Internet de las cosas o Internet of Things, está muy presente en la actualidad y cada vez hay más fabricantes que crean dispositivos de este tipo.

**¿Qué es IoT o Internet de las cosas?**

IOT es una red de **interconexión digital entre dispositivos**, personas e Internet que hace posible compartir datos, lo que permite que se pueda capturar información sobre el uso y el rendimiento de los dispositivos para así poder detectar patrones y hacer recomendaciones con las que mejorar la experiencia del usuario.

Por ejemplo, IOT es la conexión entre tu smartphone y los dispositivos Smart que tienes en casa, que pueden ser: un aire acondicionado o una Raspberry Pi (que controla la programación de tu televisor) o también puede ser un sensor de temperatura (que puede almacenar datos históricos de la temperatura de un lugar).

El termino IOT se empieza a usar cuando los dispositivos (no solo ordenadores) empiezan acceder a la red, para obtener información que necesitan y así poder dar sus servicios.

**Cómo funciona el IoT**

Los dispositivos IoT se conectan entre sí con un proceso llamado Machine to Machine (máquina a máquina). En este proceso, los dos dispositivos se comunican utilizando cualquier tipo de conectividad como, por ejemplo, Wifi o Bluetooth. De esta manera se consigue que puedan realizar un trabajo sin necesidad de que haya un usuario de por medio.

Una vez se han conectado los dispositivos, estos generan gran cantidad de datos que se almacenan en una plataforma IoT, la cual recolecta, procesa y, posteriormente ,analiza los datos. Gracias a estos datos el usuario puede sacar conclusiones de hábitos y preferencias de él mismo.

**Influencia de los dispositivos IoT en nuestra vida actual**

Durante el día a día se pueden encontrar una enorme cantidad de dispositivos que forman parte del Internet de las cosas. Se van a describir a continuación algunos ejemplos:

-Vehículos autónomos: como ya se habló anteriormente en la sección de Machine Learning, cada vez se tiene más presente el futuro de los coches con conducción autónoma (lo cual, no es otra cosa que coches que conducen solos). Conforme pasa el tiempo, estos tienen más tecnología y se basan en el uso de sensores para poder realizar dicha conducción autónoma.

-Robots aspiradoras: estos tienen gran cantidad de sensores que les permiten realizar la limpieza sin chocarse con ningún obstáculo.

-Smart home: los dispositivos smart pueden ser desde sensores de presencia que encienden las luces automáticamente, hasta sensores de temperatura o humedad (son en estos en los que se ha basado el estudio) que recogen constantemente datos que luego pueden ser analizados por el usuario.

## ANTECEDENTES DE MACHINE LEARNING APLICADO A IOT

En este punto se van a describir algunos artículos académicos publicados en la web de trabajos previos en los que se ha usado un sistema de Machine Learning aplicado a dispositivos IoT. Todos los artículos se basan en datos relacionados con el clima (contaminación del aire, radiación del sol y temperaturas). Se ha comprobado que son teóricos y no tienen una aplicación o implementación donde ver resultados en tiempo real. Existen diferentes tecnologías con las que alcanzar buenos resultados, pero se puede afirmar que todos los artículos dan como útil el uso del Machine Learning aplicado a IoT.

Caso 1**: Predicción de la energía solar diaria para sistemas fotovoltaicos con una distribución de tiempo variable** (4)

En este artículo se realizó un estudio para conseguir un pronóstico, lo más preciso, de la producción de energía en sistemas fotovoltaicos. Para ello se usaron 12 algoritmos diferentes. En la siguiente tabla se muestra cada modelo con los algoritmos que incluye:

|  |  |
| --- | --- |
| **Modelo** | **Algoritmos** |
| Linear Regression | Multivariate Linear Regression (MLR), Least Absolute Shrinkage and Selection Operator (LASSO), Seasonal Auto-Regressive Integrated Moving Average with exogenous input variables (SARIMAX) |
| Support Vector Regression | Support Vector Machine (SVM) |
| Ensemble Learning | Random Forest(RF), Gradient Boosting regression (GB) |
| Deep Learning | Artificial Neural Network (ANN), Long Short-Term Memory (LSTM) |
| Physical Models | Physical Model |
| Benchmark Models | Diurnal Persistence (DP), Clear Sky Persistence (CSP) |

Tabla 1: Modelos y algoritmos usados en el caso 1

Como punto de partida, la metodología que se siguió se puede ver en el siguiente gráfico:

Figura 1: Metodología usada en el caso 1

Se usó un conjunto de datos o Dataset que contenía las horas de claridad diarias desde Febrero de 2014 a Febrero de 2017.

El Dataset, además de lo mencionado, también tenía los siguientes datos, considerados como atributos: presión del aire, presión media al nivel del mar, temperatura ambiente, temperatura en el punto de rocío, precipitación, velocidad del viento por zona, velocidad del viento meridional, variación de la nubosidad, cielo despejado (GHI), radiación solar, ángulo cenital, ángulo acimutal y seno y coseno de la hora del día.

Una vez se siguieron los pasos que se indican en metodología de la **Figura 2**, se obtuvo el siguiente resultado:

-Perspectiva técnica: los resultados fueron favorables. Se pudo comprobar que los modelos basados en Ensemble Learning (ANN y LSTM) fueron los que más bajo Mean Absolute Error (MAE) consiguieron, seguidos de los modelos de Deep Learning (RF y GB) que también obtuvieron un buen pronóstico.

Perspectiva económica: los resultados fueron negativos. El modelo que mejor funcionó fue el Modelo Físico, seguido de los modelos de Deep Learning (RF y GB).

Caso 2**: Predicción de la polución del aire en ciudades inteligentes mediante algoritmos de Machine Learning** (5)

Este artículo se realizó en Murcia, y tuvo como finalidad realizar una predicción del nivel de ozono en la región. Para ello se usaron 6 algoritmos diferentes que fueron: Bagging, Random Committe, Random Forest, Decision Tree, K Nearest Neightbors, y Hieralchical cluster.

Como punto de partida, la metodología que se siguió es la misma que la de **Figura 2.**

Se usaron cinco conjuntos de datos obtenidos en diferentes puntos o ciudades. Los datos contenían los siguiente atributos: el promedio por hora de elementos químicos (NO, NO2, SO2, NOx, PM10, C6H6, tolueno (C7H8), xileno (XIL)) y los parámetros climáticos: temperatura, humedad relativa, dirección, velocidad del viento, presión atmosférica y radiación solar. Todos estos datos se recogieron cada día durante los años 2013–2014. En la siguiente tabla se recoge los resultados obtenidos para cada Dataset y algoritmo:

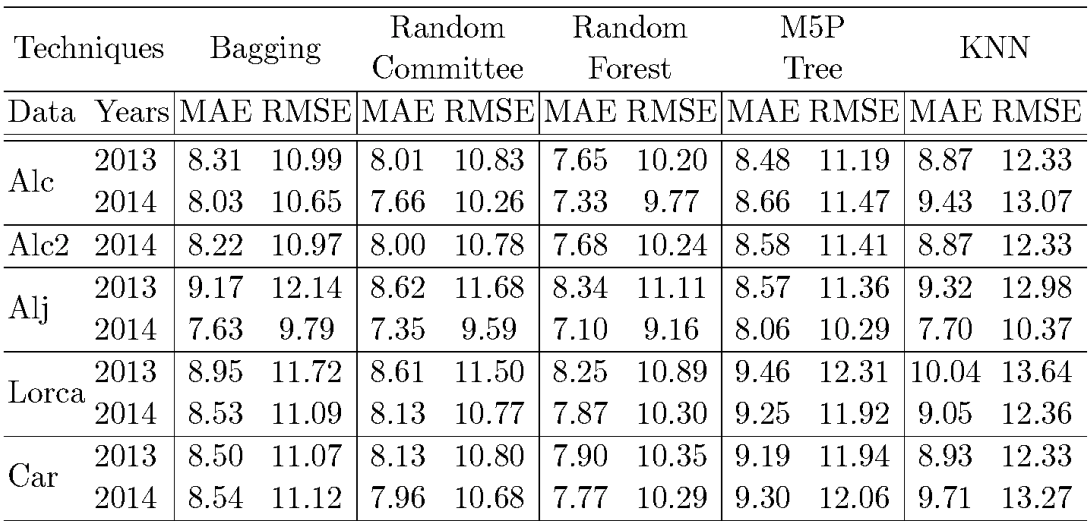


Ilustración 1: Resultados RMSE y MAE del Caso 2 (5)

Como se puede observar en la tabla de arriba, el algoritmo que dio mejor resultado fue el Random Forest, ya que tenía el RMSE y MAE más bajos.

En cuanto a la relación con R2, se estableció un umbral de 0,75 y se consideró como buen resultado todo el que lo sobrepasara. En la siguiente tabla se pueden ver los resultados:

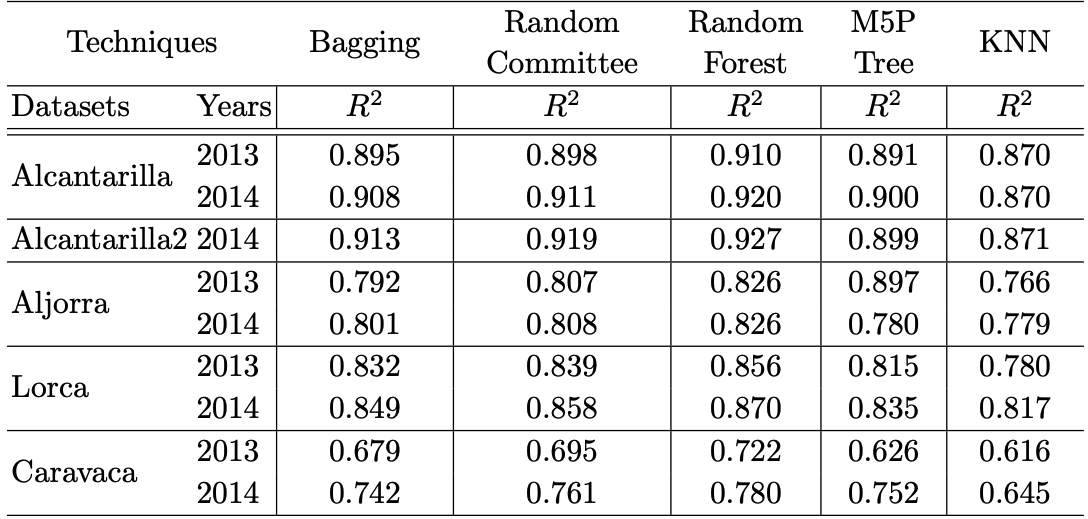


Ilustración 2: Resultados R2 del Caso 2 (5)

En este caso, lo sobrepasaron todos los algoritmos, ya que todos dieron un R2 por encima del 0,80.

Caso 3**: Comparación de modelo XGBoost con otros modelos para predecir energía eólica a corto plazo** (6)**.**

En este tercer artículo se realizó un estudio para conseguir una predicción de la energía eólica que se podría generar teniendo en cuenta los problemas que surgen debido al método de obtención. Uno de estos principales problemas a los que se enfrenta esta energía es a la aleatoriedad del viento, esto producido por variedad de factores como el terreno, la estación del año (no es lo mismo verano que invierno), la presión del aire, la temperatura, etc.

El estudio se basó en la comparación de los resultados de un algoritmo propio (creado por las mismas personas que realizan el estudio) usando como base XGBoost, con otros algoritmos como Random Forest (RF), Classification and Regression Trees (CART), Back Propagation Neural Network (BPNN), XGBoost y Support Vector Regression (SVR).

El modelo de datos contenía los siguientes atributos: día de la semana (0-6), día del año (0-365), día del mes (1-31), mes del año (1-12), hora del día (0-23), minuto del día (0-1339), valor de la energía eólica de las 24h anteriores, valor de la energía eólica de las 48h anteriores, velocidad del viento, dirección del viento, temperatura, humedad y presión.

El valor que se quería predecir es la velocidad del viento. En la siguiente tabla, se pueden ver los resultados de los algoritmos que se compararon. Se puede ver que para realizar esta comparación se han usado como indicadores MSE, RMSE, MAE y RMAE.

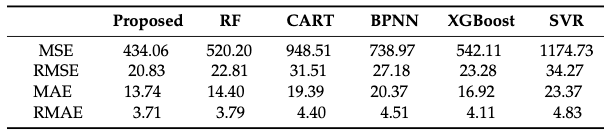


Ilustración 3: Resultados MSE, RMSE, MAE y RMAE del Caso 3

En la tabla se muestra que el algoritmo que mejor resultado consiguió fue el que el articulo propuso, consiguiendo valores más bajos en todos los indicadores.

Caso 4**: Predicción del tiempo utilizando técnicas de aprendizaje automático** (7)**.**

En este último artículo se realizó un estudio para conseguir una predicción de la temperatura ambiental.

La metodología que se usó para llevar a cabo este estudio fue la que se puede ver la siguiente imagen.

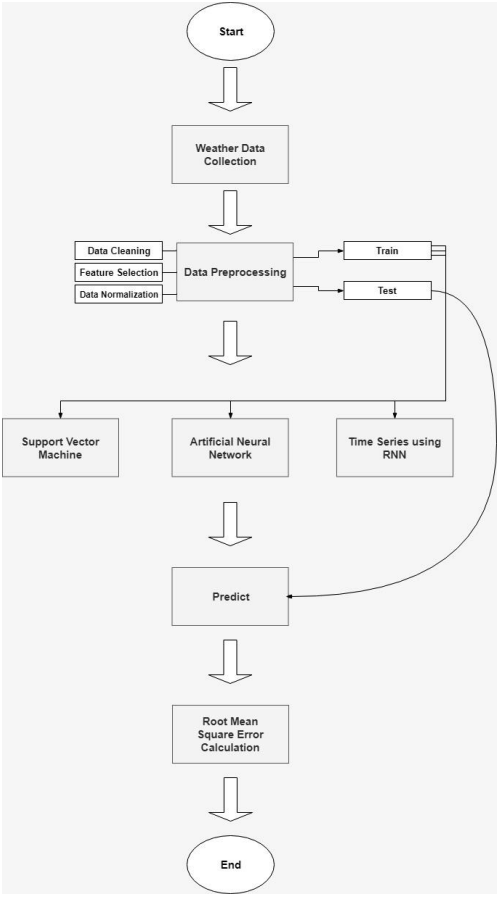


Ilustración 4: Metodología del caso 4

Se puede ver en la imagen, que lo primero que se hizo fue recolectar datos relacionados con el clima con lo necesario para poder tener un modelo de datos sólido y así conseguir una buena predicción. Posteriormente, realizaron el preprocesado de los datos, que consiste en la limpieza de datos no válidos, selección de los mejores features o atributos y normalización de los datos. Después de esto, estos datos se le pasaron a los algoritmos, en este caso, los que usaron son Support Vector Machine (SVM), Artificial Neural Network (ANN) y Time Series using RNN.

Con respecto al conjunto de datos, los atributos que contenía son los siguientes: temperatura del aire a 2 metros de altura sobre la superficie terrestre, presión atmosférica a nivel de estación meteorológica, presión atmosférica reducida al nivel medio del mar, humedad relativa a una altura de 2 metros sobre la superficie terrestre, dirección media del viento a una altura de 10-12 metros sobre la superficie de la tierra, nubosidad total, visibilidad horizontal, temperatura del punto de rocío a una altura de 2 metros sobre la superficie de la tierra. Este conjunto de datos contiene datos desde 2006 a 2018.

Después del análisis de todos los modelos, se puede ver en la siguiente tabla los resultados usando como referencia el indicador de error RMS:

|  |  |  |
| --- | --- | --- |
| **Modelo** | **Tiempo de predicción** | **RMS** |
| Support Vector Machine (SVM) | 8 semanas | 6,67 |
| Artificial Neural Network (ANN) | 8 semanas | 3,1 |
| Time Series using RNN | 8 semanas | 1,41 |

Fijándonos en la tabla se puede ver que el algoritmo que mejor resultado dio fue Time Series using RNN, con error cuadrático medio o RMS de 1,41.

# METODOLOGÍA

Partiendo de la base de que el significado de Metodología es según Wikipedia *“conjunto de procedimientos racionales utilizados para alcanzar el objetivo”* (8)*,* este proyecto se realiza en función de una serie de procedimientos para llegar a un objetivo final. En el siguiente gráfico se muestra los pasos que se han ido llevando a cabo para el desarrollo de todo el análisis. Posteriormente, se dará una explicación de cada uno de ellos.

Inicio

Creación de base de datos

Recolección de datos

Limpieza de datos inválidos

Selección de los Features o atributos

Preprocesado

Creación Front y Back

División de datos en Train y Test

Inicializar algoritmo

mo

Entrenamiento de datos

Realizar predicción

Creación de modelo

Pintar modelo en Front

Figura 2: Metodología del proyecto

**Parte izquierda del flujo**

En la parte izquierda del flujo se realizan los pasos necesarios para crear un modelo entrenado y poder posteriormente usarlo para obtener predicciones. Los paso son:

-Recolección de datos: el primer paso de la metodología que se ha seguido, ha consistido en realizar un análisis de algunas páginas web para poder recolectar datos y conseguir un modelo sólido que contenga la suficiente información para obtener una buena predicción.

-Preprocesado: en este paso se realizan dos acciones, que son las siguientes:

* Limpieza de datos: en ocasiones, cuando revisas el modelo de datos, encuentras que hay algunos atributos que para algún día no tienen elementos o son incorrectos. Cuando esto ocurre se debe realizar una limpieza, o bien eliminarlos o modificarlos manualmente para que sea un conjunto consistente.
* Selección de los Features o atributos: uno de los requisitos de los algoritmos de Machine Learning es que se le debe indicar cuáles van a ser los atributos o características donde debe buscar el patrón que aprender. Por ejemplo, presión atmosférica, día del mes o mes de año.

-División de datos en Train y Test: en este paso se divide el conjunto de datos en un conjunto llamado Train, que se usa para entrenar el modelo. También, en un conjunto Test, que se utiliza para una vez entrenado el modelo, pasárselo y contractar si los datos que predice son correctos. Normalmente, la división suele ser de un 80% para crear el modelo de entrenamiento y de un 20% para poder contractar los resultados con el conjunto de Test.

-Inicializar algoritmo: en este paso se inicializa el algoritmo, pasándole por parámetros la configuración que necesite.

-Entrenamiento de datos: una vez se ha inicializado el algoritmo, se realiza el entrenamiento, que no es otra cosa que crear un modelo a partir de los datos que se han recolectado y aprende un patrón para poder realizar la posterior predicción.

-Realizar predicción: se le pasa algoritmo un conjunto de datos que no ha sido previamente entrenado y debe devolver un resultado, que tienen que ser comprobado con una métrica de error, por ejemplo Mean Absolute Error (MSE).

-Creación de modelo: se crea un modelo para poder realizar predicciones de datos, cargándolo previamente sin necesidad de hacer un entrenamiento de ellos cada vez que se haga una predicción.

**Parte derecha del flujo**

En la parte derecha del flujo se realizan los pasos para crear un Back y un Front donde poder recrear los datos. Los pasos son:

-Creación de base de datos: este paso consiste en crear una base de datos donde almacenar información, que posteriormente será tratada por el Back y representada por el Front.

-Creación Front y Back: se crea un Back con PHP, donde se realizan todas las llamadas a la base de datos y se ejecutan todas las funciones que se pueden manejar desde el Front, este último creado mediante el uso de HTML, CSS y JavaScript

-Pintar modelo en Front: una vez se han realizado todos los pasos de la metodología desde el JavaScript del Front, se hace un solicitud al PHP del Back, que a su vez hace una solicitud a Python y devuelve una predicción con los datos que ha recogido del formulario del Front. Estos datos se ilustran creando una representación de una línea de tiempo actual y una línea de predicción. En la siguiente imagen se puede ver la representación para un algoritmo XGBoost:

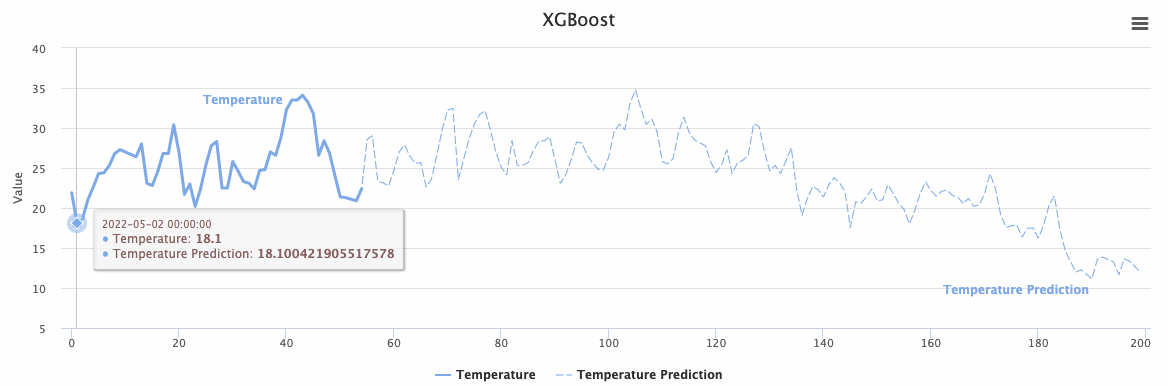


Ilustración 5: Representación de predicción usando XGBoost

# IMPLEMENTACIÓN Y DESARROLLO

En esta sección se va a realizar un análisis de toda la implementación y desarrollo del proyecto. Se explicará cómo se recolectaron los datos, que algoritmos se han usado o que proceso se ha realizado para crear el desarrollo de la interfaz. Además también se irán mostrando pequeños fragmentos de código donde se va creando la funcionalidad.

## Recolección de datos

El primer paso de todos fue recolectar los suficientes datos para crear un Dataset. Este conjunto de datos debe ser enorme ya que cuanto más grande mejor será el aprendizaje. También debe ser limpio, no puede contener errores y en el caso de contenerlos deberán eliminarse o modificarse en un posterior paso de preprocesado.

Para crear el Dataset se extrajeron datos de la página web Weather Underground (9). Para llevar a cabo dicha extracción se realizó un proceso de Web Scrapping[[1]](#footnote-1) mediante el uso del lenguaje de programación Python y su librería Selenium. Cabe destacar que todos estos datos que se recogen en esta página son recogidos mediante el uso de dispositivos IoT.

El intervalo de tiempo para el que se recogieron los datos fue desde el 1 de Enero de 2013 hasta el 31 de Abril de 2022.

En la siguiente imagen se puede ver el fragmento de código principal usado para realizar la extracción.



Ilustración 6: Fragmento de código del proceso de Web Scrapping

La función main() inicializa el web driver y declara mediante una lista de strings los años y los meses que van a ser extraídos.

Posteriormente se puede ver la variable “lista”, esta va a ser la que indique que campos va a tener el Dataset. Estos serán: año, mes, día, temperatura, humedad, velocidad del viento, presión y punto de rocío.

El funcionamiento del proceso consistirá en ir abriendo página por página para cada mes y año, cambiar la temperatura a Celsius y por ultimo ir obteniendo los datos de la tabla que se muestra a continuación y volcarlos en un CSV.

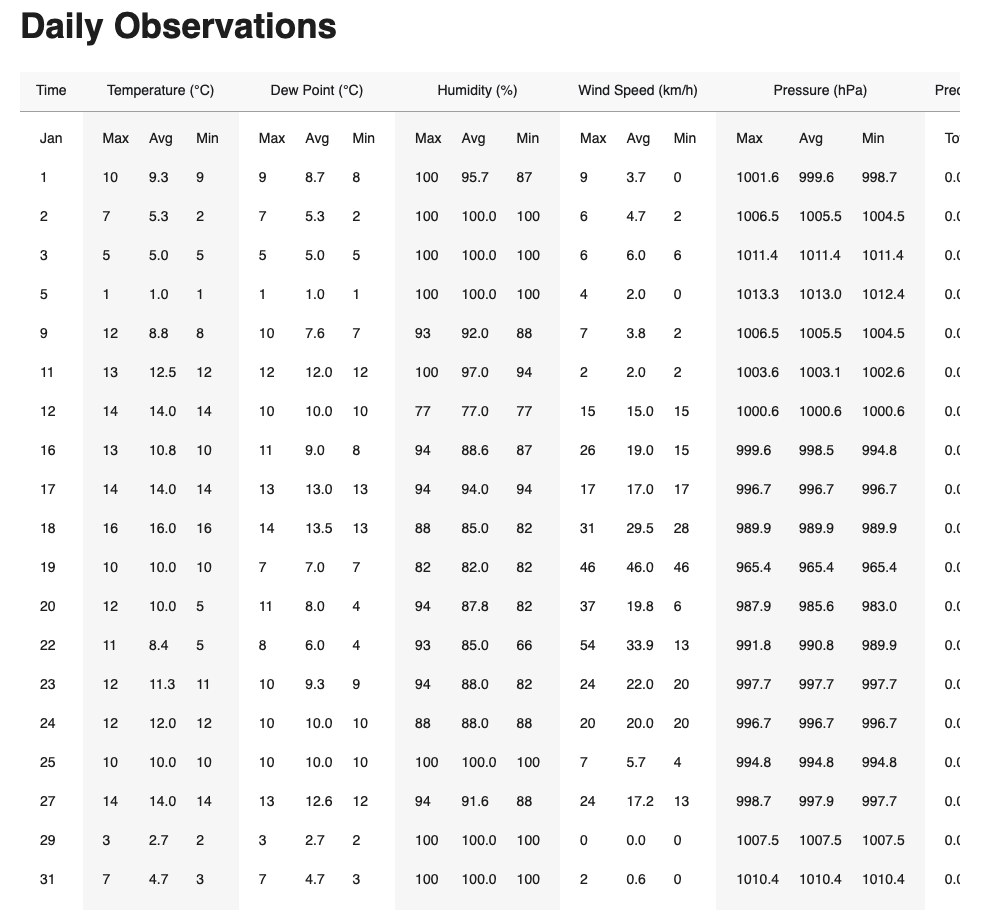


Ilustración 7: Tabla de datos de la web Weather Underground (9)

Todos los campos del csv son de tipo numérico, ya que tanto el campo año, mes y día se han guardado en el CSV como número, siendo el año 2013-2022, el mes 01-12 y el día 1-31. En la siguiente imagen se puede ver un pequeño ejemplo de algunos valores del CSV:

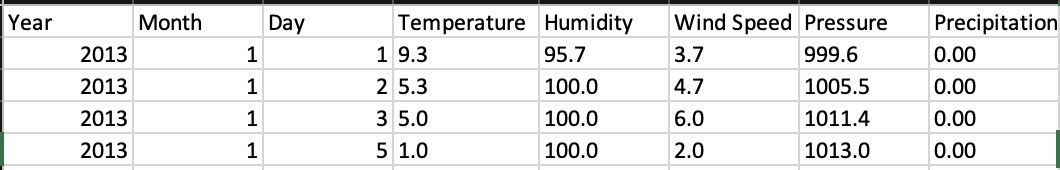


Ilustración 8; Fragmento de datos del csv

En cuanto al código, en la siguiente imagen se puede ver cómo se leen los datos del CSV en una variable y como posteriormente se realiza un eliminado de filas repetidas.

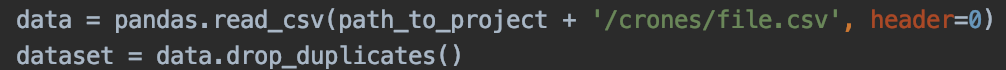


Ilustración 9: Fragmento de código de la lectura del CSV

## Selección de Features o atributos para el modelo de entrenamiento

Una vez se ha creado el CSV con los datos muestrales, se van a seleccionar las Features que se usaran para crear posteriormente el modelo de entrenamiento.

Como se comentaba en el apartado **introducción,** en este proyecto se va a realizar un proceso de Machine Learning para predecir la temperatura ambiental a partir de los datos que se han recogido. Para recordar, estos datos son: año, mes, día, temperatura, humedad, velocidad del viento, presión y punto de rocio. Con lo cual, se van a usar todos los campos (excepto el campo de temperatura) como Features de los diferentes algoritmos.

En la siguiente imagen se puede ver como se seleccionan las Features mediante código:

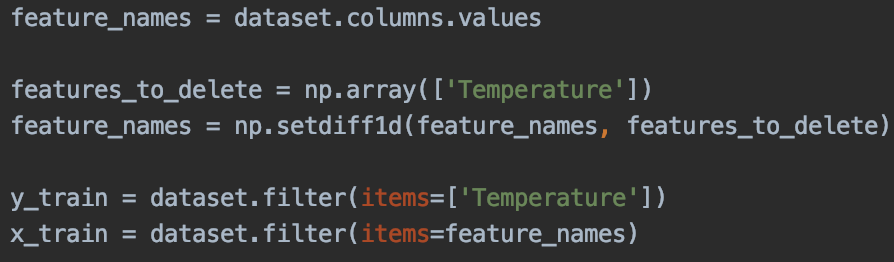


Ilustración 10: Fragmento de código de la selección de los Features

En la primera línea se puede ver que se extrae de la variable “dataset” (creada en la sección anterior) las columnas del CSV.

En la dos posteriores líneas se crean dos variables nuevas “features\_to\_delete” y “feature\_names”. En la primera se guarda un array con el campo temperatura. En la segunda se elimina mediante la función np.setdiff1d[[2]](#footnote-2) el campo temperatura y se guardan los demás.

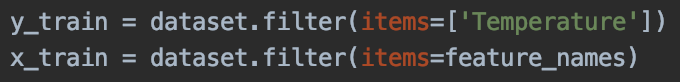
## Conjunto de entrenamiento y prueba

En esta sección se va proceder a explicar cómo se crea el conjunto de entrenamiento y el conjunto de prueba. Para ello primero se importa la librería necesaria, que es la siguiente:



Ilustración 11: Añadiendo librería para crear conjunto de entrenamiento y prueba

A continuación, mediante la siguientes dos líneas se obtienen en dos variables por un lado los datos correspondientes a la columna que se va a predecir, es decir la temperatura y por otro lado los datos correspondientes a las columnas que van a trabajar como Features o atributos, por ejemplo, la humedad o la velocidad del viento.



Teniendo esto claro, ahora se van a dividir estos datos entre un conjunto de prueba y un conjunto de entrenamiento, esto es como se comentaba en apartados anteriores, con la finalidad de que una vez creado el modelo se pueda usar dicho modelo de prueba para contractar los datos.

En el siguiente fragmento de código, se usa la librería que se ha importado. Esta divide los datos de la variable x\_train en y\_train en un conjunto de prueba y otro de entramiento. El parámetro test\_size=0.2 significa que el 20% de los datos van a ir al conjunto de prueba y el 80% al conjunto de entrenamiento.



De esta manera los datos quedan divididos así:

- x\_train = 80% del conjunto de datos de la columna a predecir para entrenamiento.

- y\_train = 80% del conjunto de datos de la columnas de atributos para crear el modelo.

- x\_test = 20% del conjunto de datos de la columna a predecir para pruebas.

- y\_test = 20% del conjunto de datos de la columnas de atributos para pruebas.

## Algoritmos implementados

En esta sección se van a describir los algoritmos que se han usado para realizar las predicciones.

**XGBoost** (10)

XGBoost o Extreme Gradient Boosting es uno de los algoritmos de Machine Learning que se clasifica dentro de los de tipo supervisado y es actualmente uno de los más usados. Es bastante conocido por conseguir muy buenas predicciones. Puede igualar o incluso mejorar los resultados de modelos más complejos. Como su propio nombre indica, este algoritmo usa el principio del boosting, que no es otra cosa que ir creando de forma secuencial varios modelos de predicción “débiles”, para que cada uno de esos modelos use los resultados del modelo anterior para generar otro modelo más “fuerte”, con mejores resultados predictivos y con mayor estabilidad en los resultados.

A continuación, se van a mostrar los fragmentos de código más importantes de la librería, que se usan para inicializar el algoritmo, entrenar el modelo y realizar una predicción:

Con los dos siguientes dos comandos, se importa la librería al proyecto y se inicializa el algoritmo:

*import xgboost as xgb*

*xgb.XGBRegressor(Parámetros de configuración=””)*

Después de inicializarlo, se realiza el entrenamiento con el comando:

*xg\_reg.fit(“Parámetros”)*

Para finalizar, con el comando de abajo se consigue una muestra de predicción:

*preds = xg\_reg.predict(“Parámetros”)*

Siendo preds el resultado.

**Linear Regression** (11)

El modelo de regresión lineal es probablemente una de las técnicas de regresión más usada en Machine Learning para realizar predicciones, ya que es uno de los métodos más simples. Una de sus principales ventajas es la facilidad con la que se pueden interpretar los resultados. Cabe destacar que es un algoritmo de tipo supervisado.

El funcionamiento consiste en que se busca la relación entre los diferentes puntos de datos y se dibuja una línea recta a través de ellos. Esta línea se puede utilizar para predecir datos futuros.

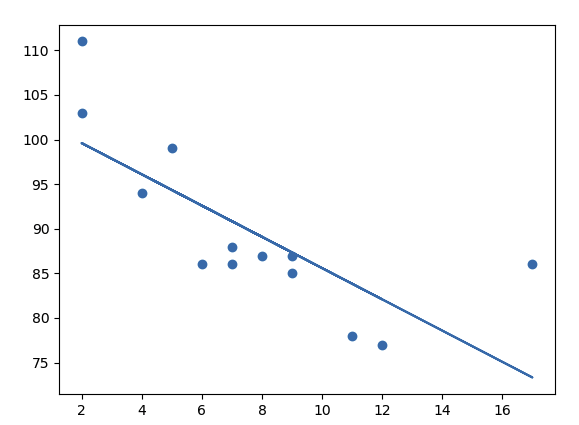
****

Ilustración 12: Ejemplo línea y puntos Regresión Lineal

Sabiendo todo esto, vamos a pasar a su implementación en Python.

Para inicializar el algoritmo se usa el siguiente comando:

*from sklearn.linear\_model import LinearRegression*

*lr\_reg = LinearRegression(Parámetros de configuración=””)*

Después de inicializarlo, se realiza el entrenamiento con el siguiente comando:

*lr\_reg.fit(“Parámetros”)*

Para finalizar, con el siguiente comando se consigue una muestra de predicción:

*preds = lr\_reg.predict(“Parámetros”)*

Siendo preds el resultado.

**Random Forest** (12)

El algoritmo Random Forest es un conjunto de árboles de decisión combinados con Bagging, esto significa que distintos árboles ven distintas porciones de los datos, por tanto, ningún árbol ve todos los datos de entrenamiento. Teniendo esto en cuenta, se sabe entonces que cada árbol se entrena con distintas muestras de datos para un mismo problema, entonces al combinar los resultados los errores se compensan unos con otros y conseguimos una mejor predicción. En la siguiente figura se puede ver un ejemplo del funcionamiento:

Figura 3: Ejemplo funcionamiento Random Forest

Cabe destacar que para problemas de regresión, la forma de combinar los resultados de los árboles de decisión al final del proceso es tomando la media aritmética.

Sabiendo todo esto, vamos a pasar a su implementación en Python.

Para inicializar el algoritmo se usa el siguiente comando:

*from sklearn.ensemble import RandomForestRegressor*

*rf\_reg = RandomForestRegressor(Parámetros de configuración=””)*

Después de inicializarlo, se realiza el entrenamiento con el siguiente comando:

*rf\_reg.fit(“Parámetros”)*

Para finalizar, con el siguiente comando se consigue una muestra de predicción:

*preds = rf\_reg.predict(“Parámetros”)*

Siendo preds el resultado.

**Support Vector Regression** (13)

El algoritmo de regresión Support Vector Regression es una modificación del modelo Support Vector Machine (utilizado para clasificar). Con esta modificación el modelo se utiliza como algoritmo de regresión para predecir valores.

En cuanto a SVM, se puede decir que es conjunto de algoritmos de aprendizaje de tipo supervisado, que están relacionados con problemas de clasificación y regresión. Su funcionamiento consiste en que a partir de un conjunto de datos de entrenamiento y con las clases etiquetadas se construye el modelo que pueda predecir una muestra nueva. De otra forma, en un conjunto de puntos dividido en dos categorías, en el que cada uno de ellos pertenece a una, este algoritmo genera un modelo capaz de predecir si un punto nuevo pertenece a una categoría o a la otra.

Sabiendo entonces cómo funciona SVM, se puede decir que SVR utiliza el mismo método con algunos pero con algunos cambios. Estos cambios son debido a que la salida de la regresión siempre es un valor real y no una etiqueta, con lo que se hace muy difícil predecir los valores.

Sabiendo todo esto, vamos a pasar a su implementación en Python.

Para inicializar el algoritmo se usa el siguiente comando:

*from sklearn.svm import SVR*

*svm\_reg = SVR(Parámetros de configuración=””)*

Después de inicializarlo, se realiza el entrenamiento con el siguiente comando:

*svm\_reg.fit(“Parámetros”)*

Para finalizar, con el siguiente comando se consigue una muestra de predicción:

*preds = svm\_reg.predict(“Parámetros”)*

Siendo preds el resultado.

**Stochastic Gradient Descent** (14)

El Stochastic Gradient Descent o descenso de gradiente[[3]](#footnote-3) estocástico es un algoritmo cada vez más popular y común que es utilizado en Machine Learning. Cabe destacar que forma la base de las redes neuronales.

Como recordatorio, el objetivo de la regresión es minimizar la suma de cuadrados de estimación de errores. Una función alcanza su valor mínimo cuando la pendiente es igual a 0. Este algoritmo es útil en los casos en los que no se pueden encontrar los puntos óptimos al igualar la pendiente de la función a 0.

El SGD es un algoritmo iterativo, que comienza desde un punto aleatorio en una función y van moviéndose por su pendiente en varios pasos hasta que consigue alcanzar el punto más bajo de esa función

Sabiendo todo esto, vamos a pasar a su implementación en Python.

Para inicializar el algoritmo se usa el siguiente comando:

*from sklearn.linear\_model import SGDRegressor*

*sgd\_reg = make\_pipeline(StandardScaler(), SGDRegressor(Parámetros de configuración=””))*

Después de inicializarlo, se realiza el entrenamiento con el siguiente comando:

*sgd\_reg.fit(“Parámetros”)*

Para finalizar, con el siguiente comando se consigue una muestra de predicción:

*preds = sgd\_reg.predict(“Parámetros”)*

Siendo preds el resultado.

**LASSO Regression** (15)

El algoritmo de Regression Lasso es un método de análisis de regresión que realiza tanto la regularización como la selección de variables. La regularización consiste en un proceso que cambia la respuesta del resultado para que sea más fácil de entender. De esta forma mejora la precisión de la predicción y hace más fácil interpretar el modelo resultante. Explicado de otra forma, consiste en que los valores de los datos se van reduciendo hasta llegar a un punto central. Este tipo de regresión es adecuado para modelos que muestran altos niveles de multicolinealidad[[4]](#footnote-4) o cuando se desea automatizar ciertas partes de la selección del modelo, como la selección de variables o eliminación de parámetros.

Sabiendo todo esto, vamos a pasar a su implementación en Python.

Para inicializar el algoritmo se usa el siguiente comando:

*from sklearn.linear\_model import Lasso*

*lasso = Lasso(Parámetros de configuración=””)*

Después de inicializarlo, se realiza el entrenamiento con el siguiente comando:

*lasso.fit(“Parámetros”)*

Para finalizar, con el siguiente comando se consigue una muestra de predicción:

*preds = lasso.predict(“Parámetros”)*

Siendo preds el resultado.

**Ridge Regression** (16)

El método de regresión Ridge es un método para estimar los coeficientes de modelos de regresión múltiple en escenarios donde las variables linealmente independientes están altamente correlacionadas

La regresión de Ridge se creó como una solución a la nula capacidad de los estimadores de mínimos cuadrados cuando se encontraba que los modelos de regresión lineal tenían algunas variables independientes multicolineales. Esto proporciona una estimación más precisa de los parámetros del algoritmo, ya que su varianza y el estimador cuadrático medio suelen ser más pequeños que los estimadores de mínimos cuadrados derivados anteriormente.

Sabiendo todo esto, vamos a pasar a su implementación en Python.

Para inicializar el algoritmo se usa el siguiente comando:

*from sklearn.linear\_model import Ridge*

*ridge = Ridge(Parámetros de configuración=””)*

Después de inicializarlo, se realiza el entrenamiento con el siguiente comando:

*ridge.fit(“Parámetros”)*

Para finalizar, con el siguiente comando se consigue una muestra de predicción:

*preds = ridge.predict(“Parámetros”)*

Siendo preds el resultado.

**Elastic Net Regression** (17)

El algoritmo de regresión lineal Elastic Net utiliza los errores de las técnicas de Lasso y Ridge para regularizar los modelos de regresión. La técnica combina los métodos de regresión de Lasso y Ridge aprendiendo de sus deficiencias para mejorar la regularización de los modelos.

Este algoritmo mejora las deficiencias de lasso y proporciona la inclusión de un número de variables hasta la saturación. Si las variables son de tipos altamente correlacionados, Lasso normalmente elige una variable de cada grupo e ignora el resto.

Sabiendo todo esto, vamos a pasar a su implementación en Python.

Para inicializar el algoritmo se usa el siguiente comando:

*from sklearn.linear\_model import ElasticNet*

*en\_regr = ElasticNet( Parámetros de configuración=””))*

Después de inicializarlo, se realiza el entrenamiento con el siguiente comando:

*en\_regr.fit(“Parámetros”)*

Para finalizar, con el siguiente comando se consigue una muestra de predicción:

*preds = en\_regr.predict(“Parámetros”)*

Siendo preds el resultado.

**Decision Tree Regressor** (18)

El algoritmo Decision Tree es una método de regresión que funciona mediante la toma de decisiones y utiliza una estructura de árbol como si fuese a un diagrama de flujo. Este algoritmo está dentro de la categoría de algoritmos de aprendizaje supervisado y se puede usar tanto para variables de salida continuas como categóricas. Las ramas pueden ser de dos tipos de nodos: nodo de condiciones, conocidos como nodos de decisión y nodos de resultados, conocidos como nodos finales.

En la siguiente imagen se puede ver un ejemplo de cómo funciona.

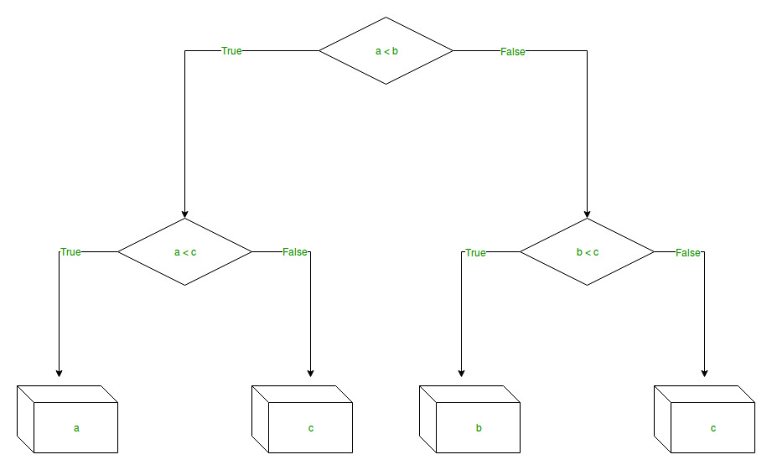
****

Ilustración 13: Ejemplo funcionamiento Decision Tree Regression

Como se puede ver, en cada nodos se van tomando una serie de decisiones hasta llegar al final. Sabiendo todo esto, vamos a pasar a su implementación en Python.

Para inicializar el algoritmo se usa el siguiente comando:

*from sklearn.linear\_model import SGDRegressor*

*sgd\_reg = SGDRegressor(Parámetros de configuración=””)*

Después de inicializarlo, se realiza el entrenamiento con el siguiente comando:

*sgd\_reg.fit(“Parámetros”)*

Para finalizar, con el siguiente comando se consigue una muestra de predicción:

*preds = sgd\_reg.predict(“Parámetros”)*

Siendo preds el resultado.

**Robust Regression RANSAC** (19)

El algoritmo RANSAC o Random Sample Consensus es un modelo de tipo iterativo que se encarga de calcular los parámetros necesario de un modelo que contiene una serie de datos con valores atípicos.

Se considera un algoritmo no determinista ya que produce un resultado ciertamente razonable, con cierta probabilidad de acierto, que es mejor a medida que se permiten más iteraciones.

La entrada al algoritmo RANSAC está formado por un conjunto de datos. El objetivo de RANSAC es que a partir de un subconjunto aleatorio de los datos que tenemos se forma un modelo contra el que se prueban todos los demás valores.

El modelo estimado es bueno si se han conseguido suficientes puntos como parte del conjunto de prueba. El modelo puede ser mejorado volviendo a estimar usando todos los valores del conjunto de prueba.

Sabiendo todo esto, vamos a pasar a su implementación en Python.

Para inicializar el algoritmo se usa el siguiente comando:

*from sklearn.linear\_model import RANSACRegressor*

*ransac\_reg = (Parámetros de configuración”=””)*

Después de inicializarlo, se realiza el entrenamiento con el siguiente comando:

*ransac\_reg.fit(“Parámetros”)*

Para finalizar, con el siguiente comando se consigue una muestra de predicción:

*preds = ransac\_reg.predict(“Parámetros”)*

Siendo preds el resultado.

## Inicialización del algoritmo

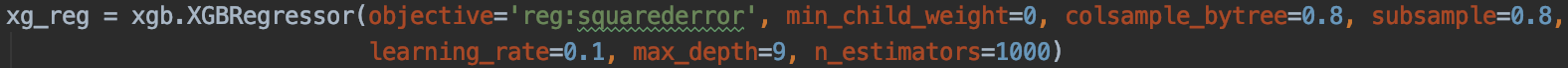
En esta sección se va a explicar cómo se inicializa un algoritmo. En la sección anterior se ha podido ver como para cada algoritmo había una serie de comando específicos para inicializarlo, entrenarlo y obtener predicciones.

Este ejemplo se va a basar en Xgboost.

Lo primero que se debe hacer en el proyecto es importar la librería:



Una vez importada la librería, se procede a inicializar el modelo con el siguiente comando:



Como se puede ver en la imagen, para inicializar el algoritmo se llama al constructor de la clase XGBRegressor y se introducen algunos parámetros de configuración para mejorar los resultados que el algoritmo puede conseguir. Al igual que en XGBoost, en los demás algoritmos también hay una serie de parámetros de configuración, que se han ido ajustando hasta encontrar los mejores valores y así conseguir las predicciones más acertadas.

Una vez inicializado el algoritmo, se puede proseguir con el entrenamiento de los datos.

## Entrenamiento del modelo

Una vez se ha recopilado el conjunto de datos, se ha creado el conjunto de entrenamiento y test y se ha inicializado el algoritmo, se pasa a realizar el entrenamiento. Para ello se usa el siguiente comando:



A la función *fit()*, se le introduce por parámetros ambos conjuntos de entrenamiento y devuelve un modelo ya entrenado, listo para predecir datos.

## Predicción de datos

Una vez realizado el entrenamiento de los datos y creado el modelo de entrenamiento, este se puede usar para predecir datos. Para ello se usa la función *predict().*

En el siguiente fragmento de código se puede ver un ejemplo donde se le pasa el conjunto de datos de prueba que tiene de contenido el 20% del conjunto de datos total, creado anteriormente:



Esta función devolverá un conjunto de datos que hacen referencia a la temperatura predicha. En la siguiente tabla podemos ver el un pequeño fragmento del conjunto de test con una columna para la predicción que ha devuelto:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Día** | **Mes** | **Humedad** | **Precipitación** | **Presión** | **Velocidad del viento** | **Temperatura predicha por el algoritmo** |
| 17 | 1 | 78.3 | 0 | 1006.4 | 2.1 | 7.93 |
| 24 | 3 | 63.9 | 0 | 989.7 | 21.1 | 11.82 |
| 12 | 5 | 43.1 | 0 | 992.4 | 11.2 | 25.15 |
| 31 | 5 | 62.7 | 0 | 996.7 | 10.2 | 21.82 |
| 3 | 9 | 50.1 | 0 | 994.0 | 11.2 | 25.45 |
| 14 | 10 | 54.4 | 0 | 997.3 | 2.9 | 20.55 |
| 16 | 10 | 66.1 | 0 | 999.9 | 2.4 | 18.26 |
| 27 | 10 | 55.5 | 0 | 996.1 | 5 | 18.10 |

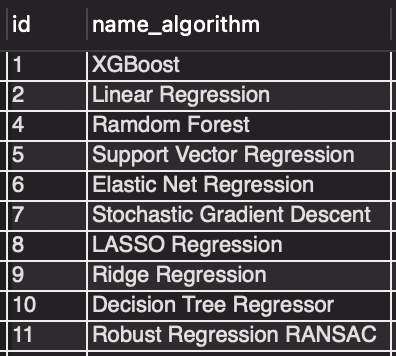
Tabla 2: Tabla de predicción para conjunto de test

Como se comprobar cuanto más cercano a los meses de verano, la temperatura aumenta, y cuando más se aleja, disminuye.

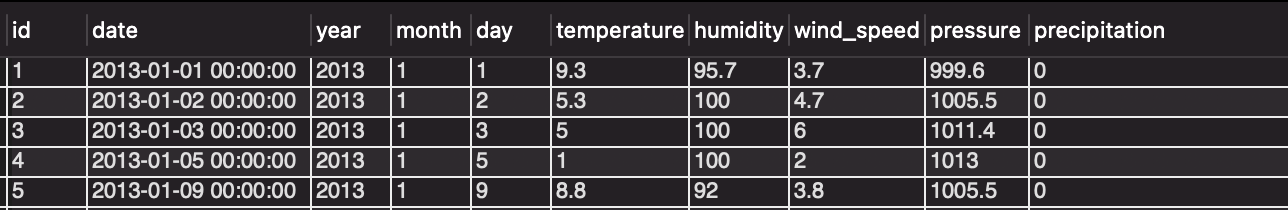
## Creación de la base de datos

Para almacenar algunos datos, se ha creado una base de datos con varias tablas. La base de datos se ha llamado SmartPolitech, y consta de tres tablas que son las siguientes:

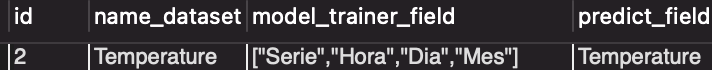
smart\_algorithms: Es la tabla donde se almacena el nombre de los algoritmos que posteriormente se usa para la representación en el Front.



smart\_data: Es la tabla donde se almacena el conjunto de los datos extraídos de la web con Web Scrapping antes de procesarlos y limpiarlos.



smart\_dataset: Es la tabla donde se almacenan los atributos que se van a predecir, se usará para mostrarlo en uno de los campos del Front.



## Creación del Front y el Back

Para la representación visual del resultado de los algoritmos se ha creado un entorno web que se ejecuta bajo un servidor apache donde se mantiene toda la lógica.

Para el back se han usado PHP y Python combinados. PHP se usa como intermediario para la comunicación entre Python y JavaScript. Cuando se realiza un función en el Front que requiere de alguna modificación, el JavaScript mediante Ajax lanza la petición a PHP con los datos del formulario que van a servir para construir el conjunto de datos con los Features que se van a pasar a Python para que este devuelva los datos predichos a partir de ellos. Estos datos irán de Python a PHP y de PHP a JavaScript mediante la solicitud que se hizo con Ajax. En la siguiente figura se representa el funcionamiento:

FRONT BACK

DB

HTML CCSS

JavaScript

PHP

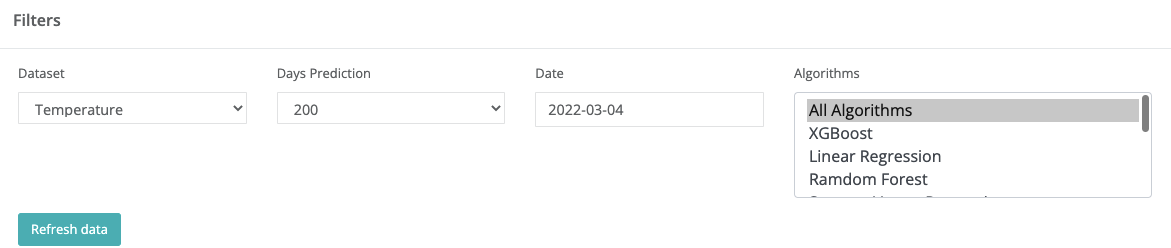
Python

## Representación de los datos

En esta sección se va a mostrar un ejemplo de representación de los datos en el Front. Pero para ello se va a realizar una explicación paso a paso de cómo funciona todo el proceso.

**Paso 1: Selección de los filtros:**

Para empezar lo primero que se hace es seleccionar en los filtros del formulario los valores que queremos:



* Dataset: se debe seleccionar el parámetro a predecir, en este caso solo hay uno que es Temperatura.
* Days Prediction: se debe seleccionar cuantos días de predicción queremos que nos devuelva el algoritmo.
* Date: se debe seleccionar a partir de que día queremos que se muestre la predicción.
* Algorithms: se debe seleccionar que algoritmo se quiere representar.

Una vez seleccionados los filtros se pulsa sobre el botón refresh data que ejecutará toda la lógica interna.

**Paso 2: Solicitud de JavaScript mediante Ajax a PHP:**

Al pulsar botón el refresh data, se inicia todo el proceso para obtener una predicción. En JavaScript se la lanza la petición Ajax a PHP. A continuación se muestra un fragmento de código de la petición de JavaScript con Ajax:



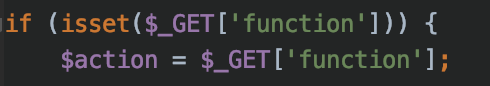
Como parámetros importantes tiene:

* Url: el archivo PHP al que se hace la petición, en este caso main.php.
* Type: método http, en este caso se usa GET.
* Data: hace referencia a los datos que se quieren enviar en la petición, en este caso, se envía el json con los datos recogidos del formulario
* Datatype: el tipo de los datos que se envían, en este caso tipo json.
* Submit: al enviarlo con valor true, le estamos diciendo que vamos a recibir una respuesta.

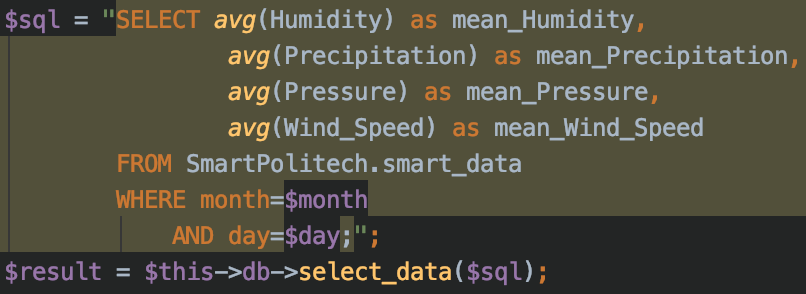
Una vez se ejecuta la llamada, el resultado será devuelto en la variable “response”.

**Paso 3: Recogida de la solicitud Ajax por PHP:**

La forma con la que PHP recoge los datos de Ajax es con el siguiente fragmento de código:

****

El parámetro “function” viene en el json que manda Ajax y tiene que ver con el algoritmo que se ha seleccionado anteriormente en el filtro. A partir de aquí se lanza el procesado de datos que hay que mandar a Python para que devuelva la predicción. Hay que recordar, que para que se devuelva esa predicción hay que pasarle una serie de datos al modelo ya entrenado (los features). Entonces para cada día que se quiere predecir, se realiza una consulta a la tabla de la base de datos donde se solicita que para ese día devuelva la media de los datos teniendo en cuenta todos años de datos que hay registrados. Se puede ver la consulta en la siguiente imagen:



Esta consulta devolverá la media de la humedad, precipitación, presión y velocidad el viento. Estos datos se juntaran en un array junto al día y el mes y se le enviaran a Python para que procese el modelo ya entrenado con ellos.

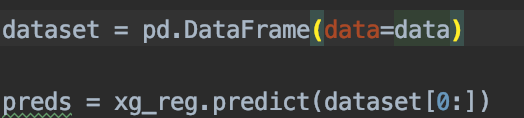
**Paso 3: Recogida de los Features por Python y devolución de predicción:**

Cada algoritmo a parte de un archivo donde se implementa el código para realizar los entrenamientos, también cuenta con un archivo para cargar el modelo ya creado juntos con los datos que le llegan de PHP.

Por ejemplo para el algoritmo XGBoost, para cargar el modelo se usa el comando load y devuelve en la variable toda la información del modelo.



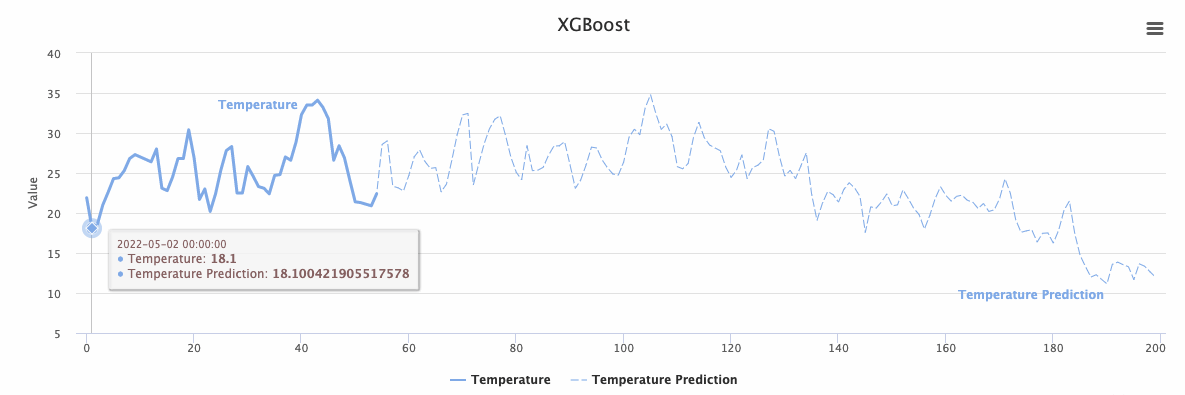
Una vez se tiene en la variable el modelo, se crea una variable con los datos de los Features que se han recogido de PHP y se pasan como parámetro a la función de predicción predict():



Esto generará un valor que corresponde con la temperatura predicha para el día que seleccionamos en el formulario. Esta temperatura será devuelta a PHP que a su vez la devolverá a la petición de Ajax.

**Paso 4: Representación visual Web**

Cuando JavaScript ya tiene los valores, los representa en el entorno web mediante una librería llamada Highcharts (20) quedando una visualización de una gráfica donde se pinta la línea de temperatura real hasta el día actual y a partir de ahí otra línea con la predicción.



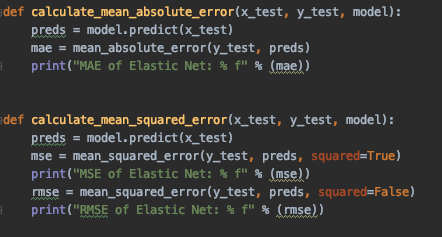
# RESULTADOS Y DISCUSIÓN

En esta sección se va a realizar una comparación de todos los algoritmos y para ello se van a utilizar las métricas Mean Squared Error (MSE) y Mean Absolute Error (MAE).

Para calcular estas métricas se han usado dos clases de la librería de Python Sklearn (21), las cuales son las de la siguiente imagen:



Se han creado dos funciones dentro de cada archivo de entrenamiento del modelo en Python, una para calcular el MAE y otra para calcular el MSE y RMSE:

****

Los pasos que siguen las funciones son:

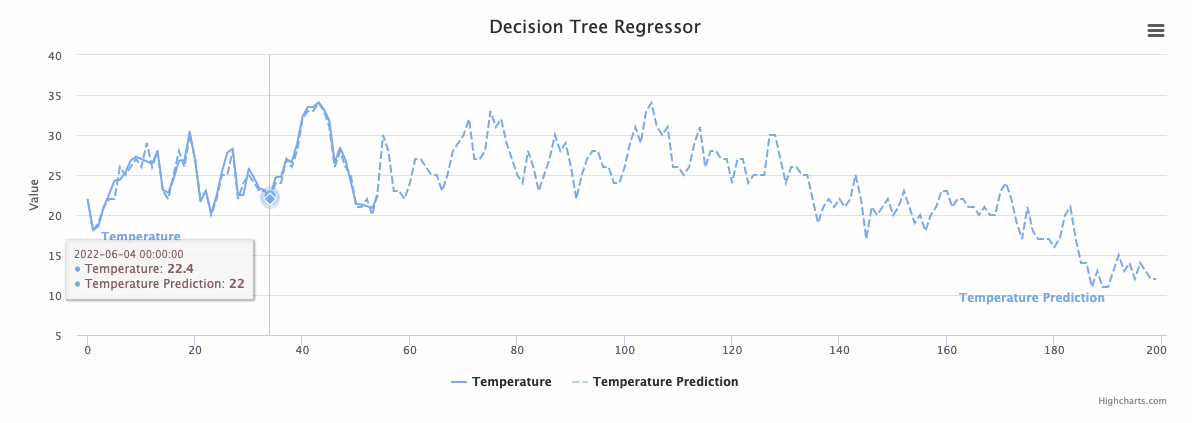
Con el modelo ya previamente entrenado, se realiza una predicción usando el conjunto de test con las Features creado anteriormente con el 20% de la muestra total. Con el resultado, se usa las función *mean\_squared\_error*, pasando por parámetros el conjunto resultado de la predicción y el conjunto de test con la temperatura real. Esto comparará ambos conjuntos y nos devolverá el MAE. Usando estos dos mismo conjuntos, se llama a la función *mean\_squared\_error* para obtener el MSE o el RMSE dependiendo de si se le pasa el parámetro *squared* a False o a True, que se resultaría en MSE y RMSE respectivamente.

A continuación, se van a ir mostrando capturas de una gráfica para cada algoritmo junto con el resultado de las métricas. Posteriormente se va mostrarán en tablas todos los resultados juntos para ver la comparación más clara.

Como observación, para todos los algoritmos se ha realizado la predicción a partir del día 1 de Mayo de 2022 y se han solicitado 200 días de predicción.

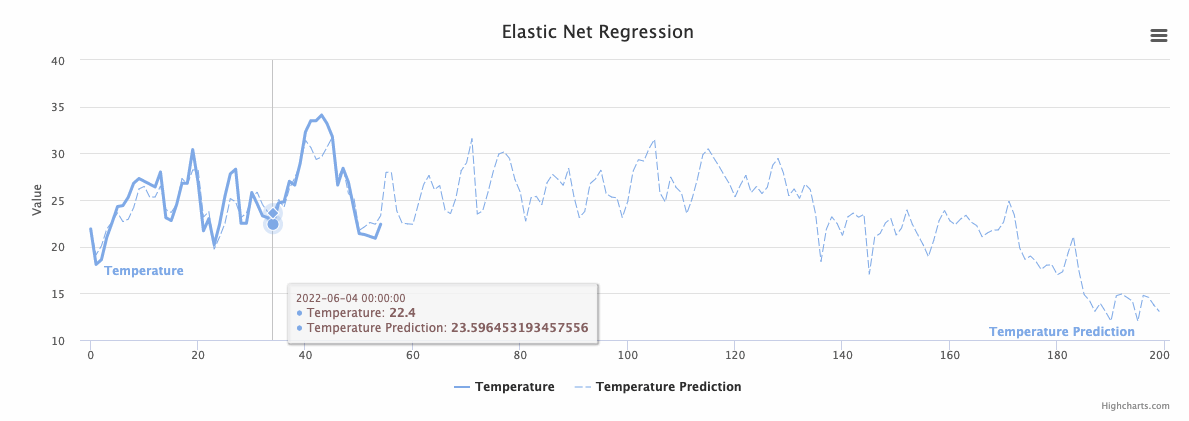
Como se podrá ver en cada gráfica, se ha seleccionado el día 4 de Junio para mostrar el recuadro que se forma cuando pones el ratón encima con ambos valores, tanto el real como el de predicción.

**Decision Tree Regresor**



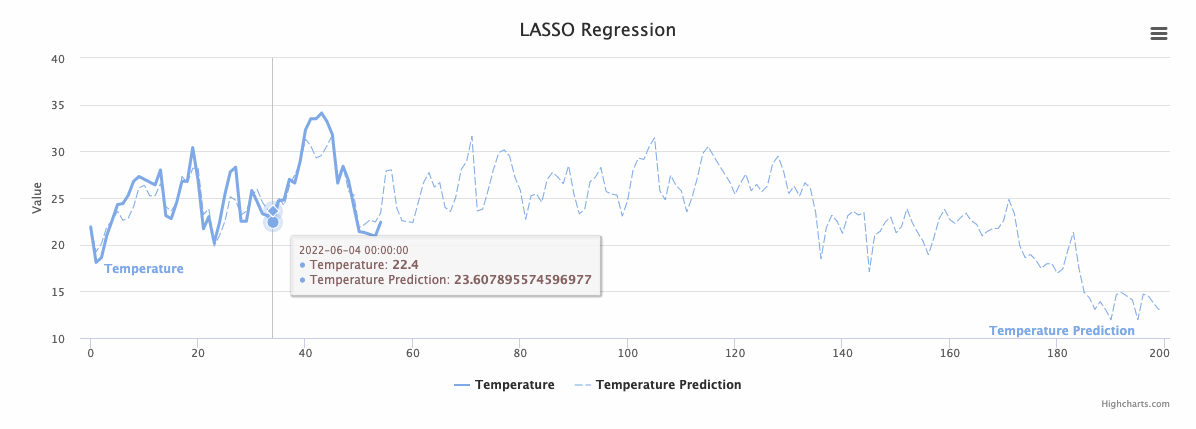
|  |  |  |
| --- | --- | --- |
| **MAE** | **MSE** | **RMSE** |
| 0.797337 | 1.431751 | 1.196558 |

**Elastic Net Regression**



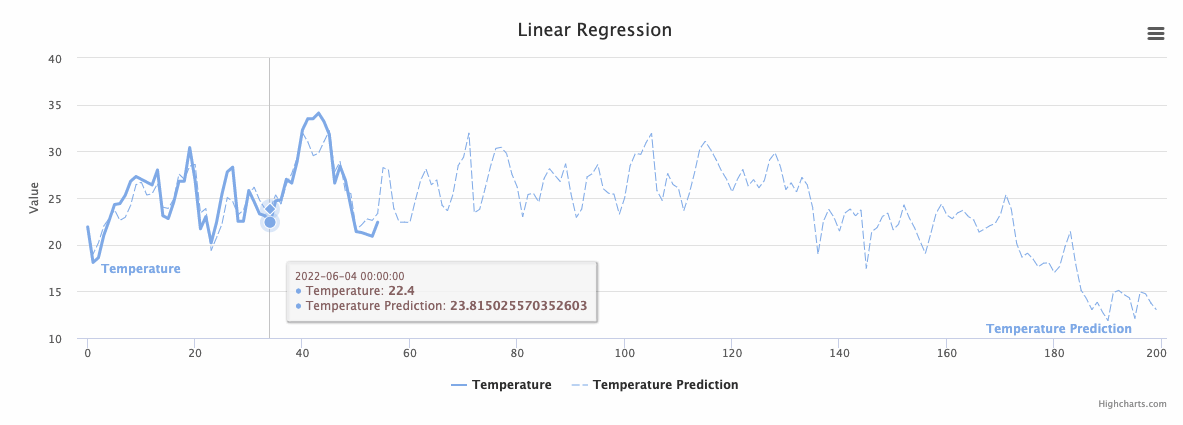
|  |  |  |
| --- | --- | --- |
| **MAE** | **MSE** | **RMSE** |
| 0.902506 | 1.288683 | 1.135202 |

**Lasso Regression**



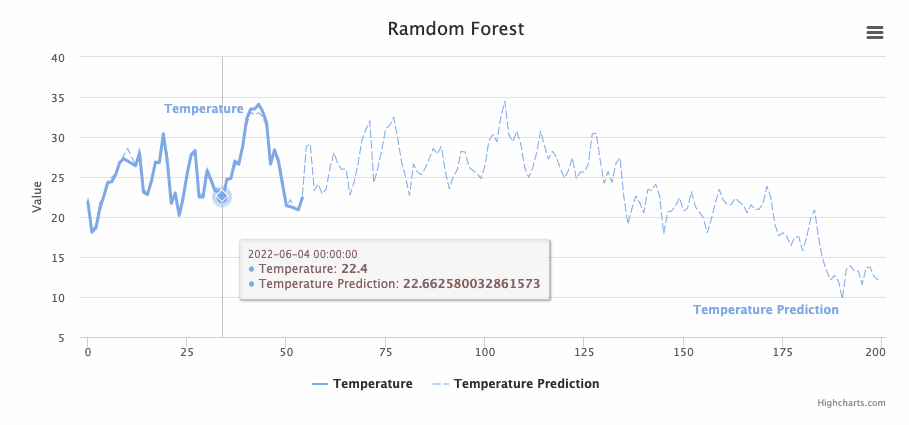
|  |  |  |
| --- | --- | --- |
| **MAE** | **MSE** | **RMSE** |
| 0.906175 | 1.299675 | 1.140033 |

**Linear Regression**



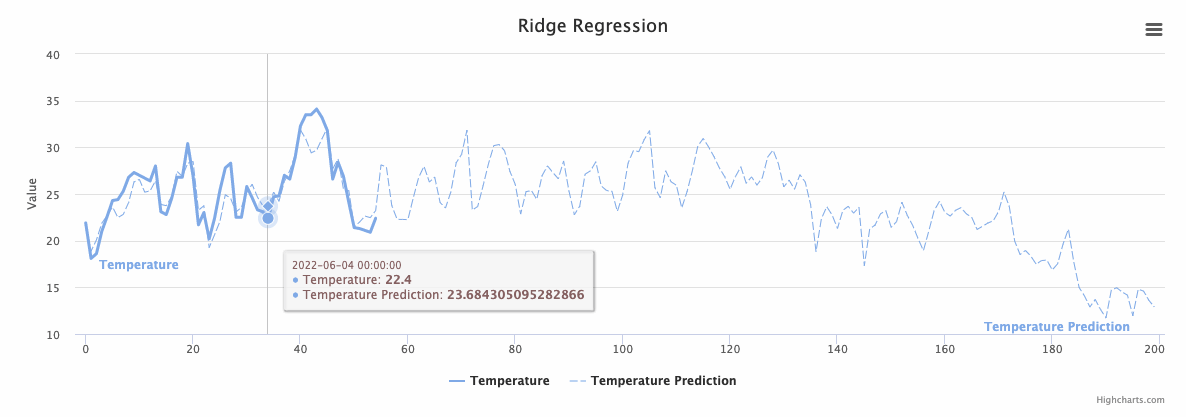
|  |  |  |
| --- | --- | --- |
| **MAE** | **MSE** | **RMSE** |
| 0.849584 | 1.142320 | 1.068794 |

**Random Forest**



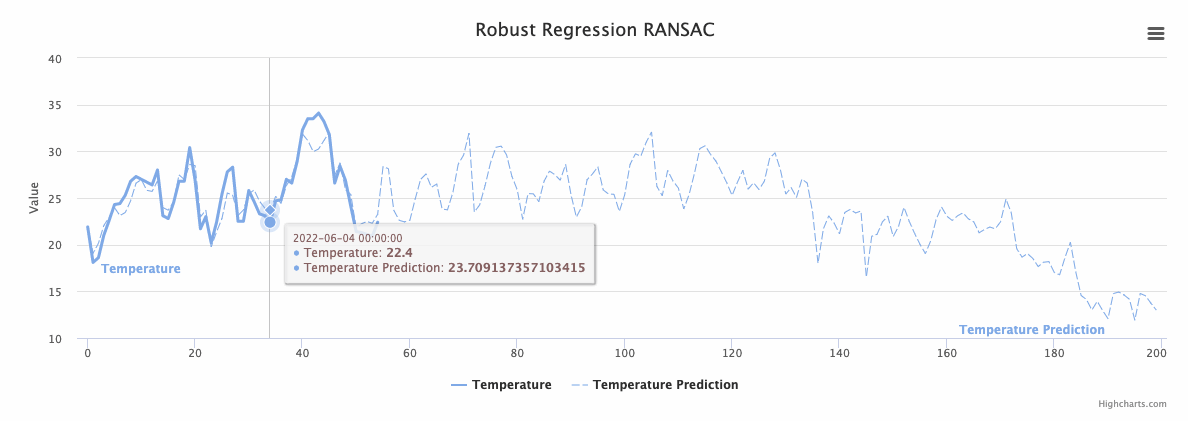
|  |  |  |
| --- | --- | --- |
| **MAE** | **MSE** | **RMSE** |
| 0.420766 | 0.395857 | 0.629171 |

**Ridge Regression**



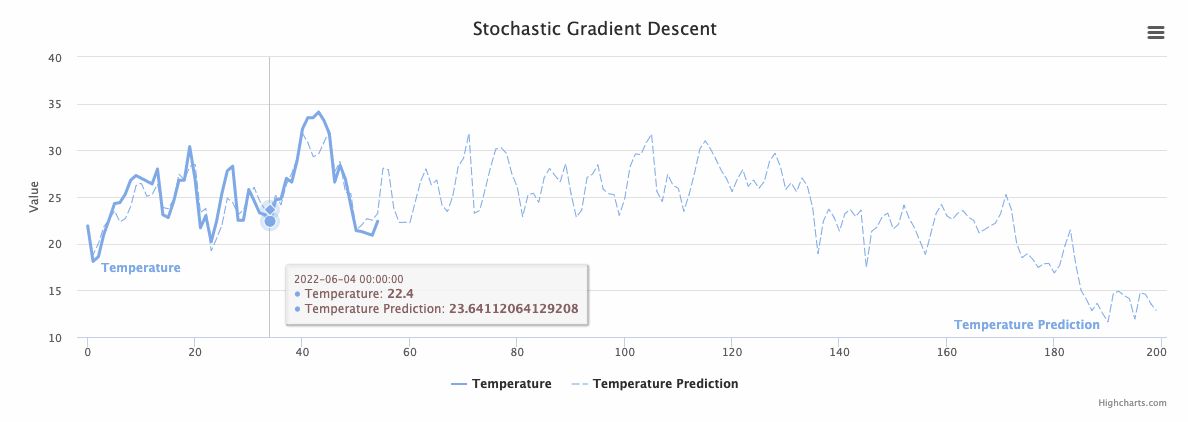
|  |  |  |
| --- | --- | --- |
| **MAE** | **MSE** | **RMSE** |
| 0.849591 | 1.142341 | 1.068803 |

**Robust Regression RANSAC**



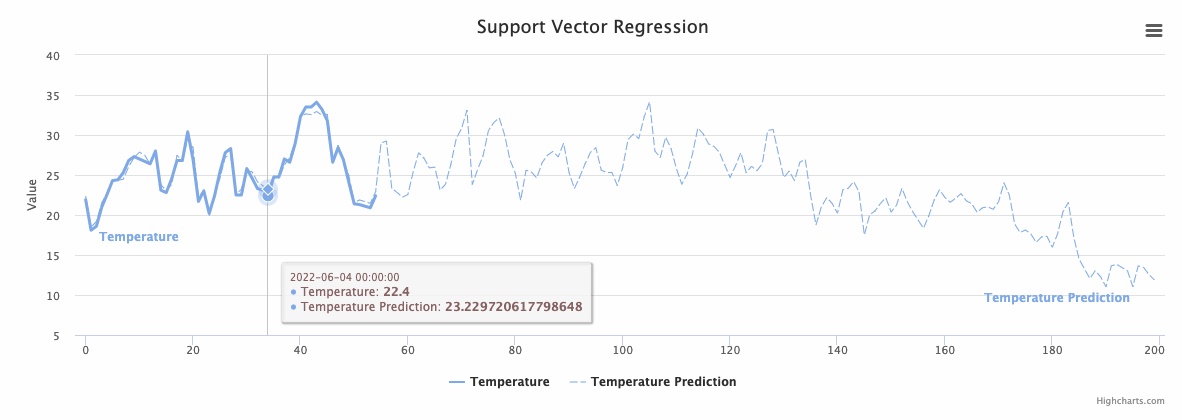
|  |  |  |
| --- | --- | --- |
| **MAE** | **MSE** | **RMSE** |
| 0.857084 | 1.159062 | 1.076597 |

**Stochastic Gradient Descent**



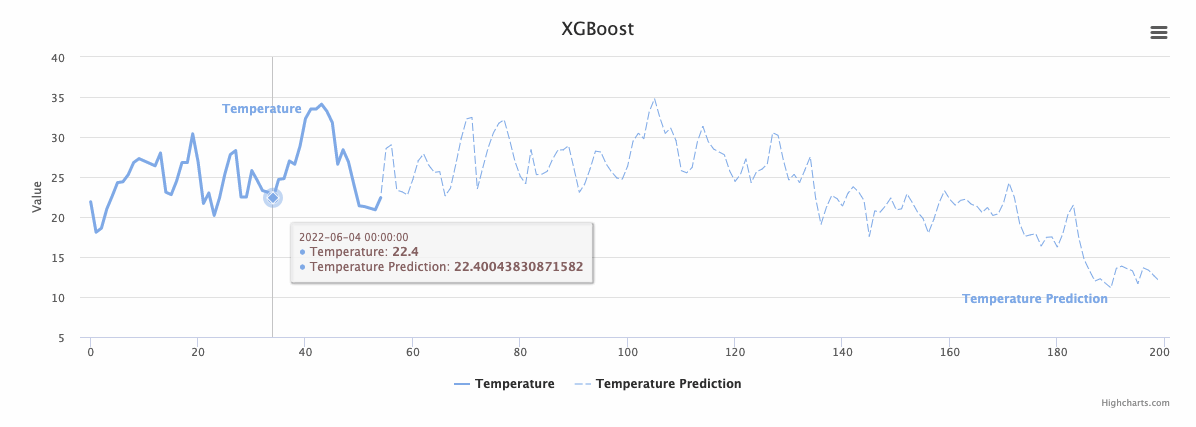
|  |  |  |
| --- | --- | --- |
| **MAE** | **MSE** | **RMSE** |
| 0.845497 | 1.139959 | 1.067689 |

**Support Vector Regression**



|  |  |  |
| --- | --- | --- |
| **MAE** | **MSE** | **RMSE** |
| 0.379213 | 0.300039 | 0.547758 |

**XGBoost**



|  |  |  |
| --- | --- | --- |
| **MAE** | **MSE** | **RMSE** |
| 0.470588 | 0.427915 | 0.654152 |

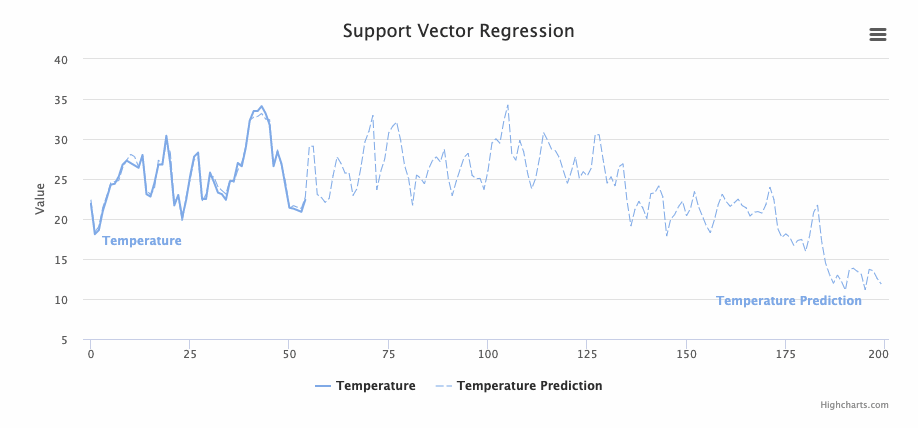
A continuación se muestran dos tablas comparativa con el valor de la métricas MAE en la primera y MSE y RMSE en la segunda para todos los algoritmos. Las tablas está ordenada de mejor a peor resultado en función del MAE y MSE

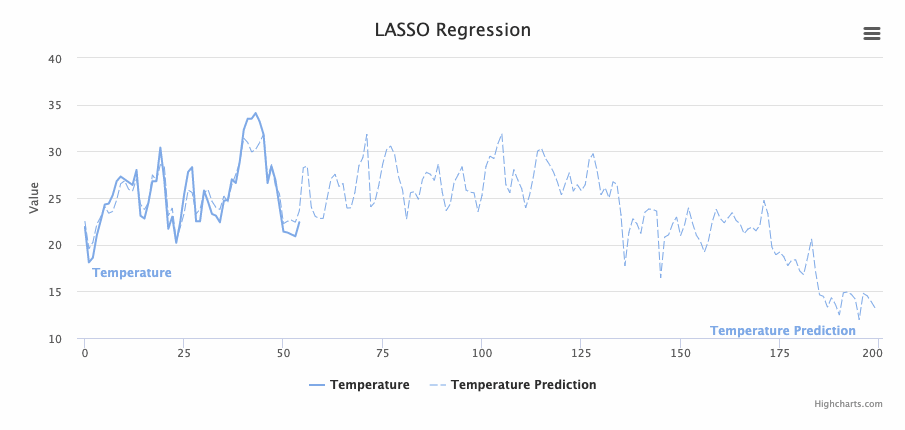
**Tabla comparativa MAE**

|  |  |
| --- | --- |
| **Nombre algoritmo** | **MAE** |
| Support Vector Regression | 0.379213 |
| Random Forest | 0.420766 |
| XGBoost | 0.470588 |
| Decision Tree Regresor | 0.793769 |
| Stochastic Gradient Descent | 0.845497 |
| Linear Regression | 0.849584 |
| Ridge | 0.849591 |
| RANSAC | 0.857084 |
| Elastic Net Regression | 0.902506 |
| Lasso Regression | 0.906175 |

Como se puede ver en la tabla el algoritmo que mejor resultado ha conseguido con esta métrica es Support Vector Regression seguido de cerca por Random Forest y XGBoost. Lasso Regression es el que peor resultado ha conseguido. A continuación se va a mostrar la gráfica de Support Vector Regression y la de Lasso Regression donde se puede ver que la predicción del primero es más precisa como apuntan los resultados.

**Support Vector Regression vs Lasso Regression**



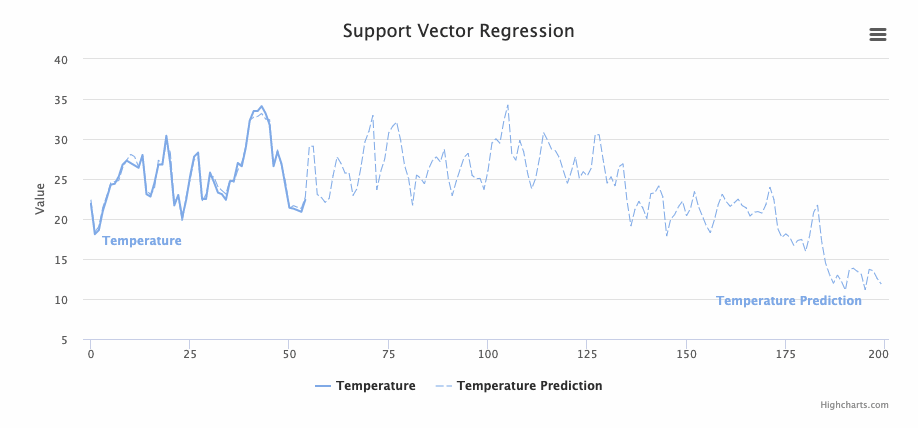
****

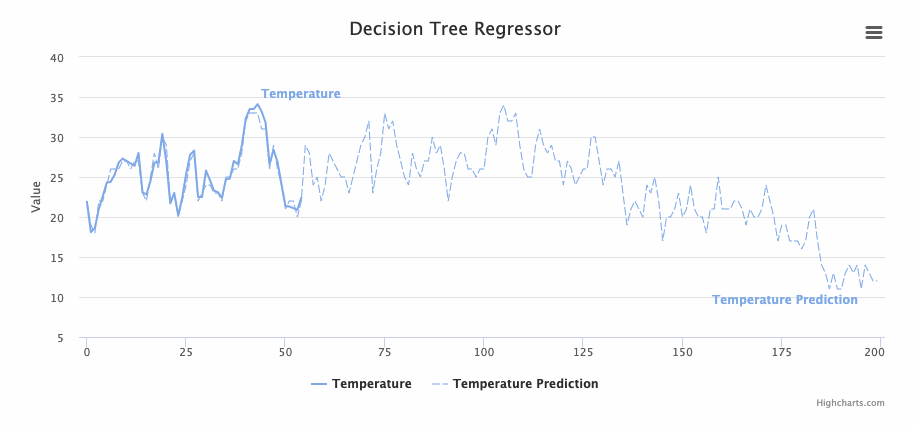
**Tabla comparativa MSE, RMSE**

|  |  |  |
| --- | --- | --- |
| **Nombre algoritmo** | **MSE** | **RMSE** |
| Support Vector Regression | 0.300039 | 0.547758 |
| Random Forest | 0.395857 | 0.629171 |
| XGBoost | 0.427915 | 0.654152 |
| Stochastic Gradient Descent | 1.139959 | 1.067689 |
| Linear Regression | 1.142320 | 1.068794 |
| Ridge | 1.142341 | 1.068803 |
| RANSAC | 1.159062 | 1.076597 |
| Elastic Net Regression | 1.288683 | 1.135202 |
| Lasso Regression | 1.299675 | 1.140033 |
| Decision Tree Regresor | 1.431751 | 1.196558 |

Como se puede ver en la tabla el algoritmo que mejor resultado ha conseguido con esta métrica es también, Support Vector Regression, seguido como en la anterior métrica por Random Forest y XGBoost. Decision Tree Regressor es el que peor resultado ha conseguido. A continuación se va a mostrar la gráfica de Support Vector Regression y la de Decision Tree Regressor donde se puede ver que la predicción del primero es más precisa como apuntan los resultados.

**Support Vector Regression vs Decision Tree Regressor**





# CONCLUSIONES

# REFERENCIAS BIBLIOGRÁFICAS

1. **Iberdrola.** Descubre los principales beneficios del Machine Learning. [Online] 2019. https://www.iberdrola.com/innovacion/machine-learning-aprendizaje-automatico.

2. **Burkov, Andriy.** The Hundred-Page Machine Learning Book. 2019.

3. **sitiobigdata.** sitiobigdata.com. [En línea] https://sitiobigdata.com/2018/08/27/machine-learning-metricas-regresion-mse/#:~:text=Ahora%2C%20es%20muy%20importante%20entender,una%20funci%C3%B3n%20que%20no%20disminuye..

4. **Visser, Lennard, AlSkaif, Tarek y van Sark, Wildfried.** Elveiser.com. [En línea] 2022. https://reader.elsevier.com/reader/sd/pii/S0960148121015688?token=280D13A6F4479E4B92F62354AF166484F7451982D4F5B6F488C28139DAFC90A4CD62B2552EBE9D7905CDD16085AA1BFB&originRegion=eu-west-1&originCreation=20220605172019.

5. **Martínez, Raquel, y otros.** www.jucs.org. [En línea] 2018. https://www.jucs.org/jucs\_24\_3/air\_pollution\_prediction\_in/jucs\_24\_03\_0261\_0276\_espana.pdf.

6. **Zheng, Huan y Wu, Yanghui.** www.researchgate.net. [En línea] https://www.researchgate.net/publication/334711620\_A\_XGBoost\_Model\_with\_Weather\_Similarity\_Analysis\_and\_Feature\_Engineering\_for\_Short-Term\_Wind\_Power\_Forecasting.

7. **Singh, Siddharth, y otros.** deliverypdf.ssrn.com. [En línea] https://deliverypdf.ssrn.com/delivery.php?ID=190081025071014118094007117025066121042037029042091050024118019086002125092082010075126060062111031022046118098094122106010098049015010026004066095115099094071080108023077048127111007017087072100003093115020102.

8. **Wikipedia. [En línea] https://es.wikipedia.org/wiki/Metodolog%C3%ADa.**

**9. Weather Underground. [En línea] https://www.wunderground.com/.**

**10. Vega, Juan Bosco Mendoza. *medium.com.* [En línea] https://medium.com/@jboscomendoza/tutorial-xgboost-en-python-53e48fc58f73.**

**11. *w3schools.com.* [En línea] https://www.w3schools.com/python/python\_ml\_linear\_regression.asp.**

**12. *iartificial.net.* [En línea] https://www.iartificial.net/random-forest-bosque-aleatorio/.**

**13. *jacobsoft.com.mx.* [En línea] https://www.jacobsoft.com.mx/es\_mx/support-vector-regression/.**

**14. *realpython.com.* [En línea] https://realpython.com/gradient-descent-algorithm-python/.**

**15. *en.wikipedia.org.* [En línea] https://en.wikipedia.org/wiki/Lasso\_(statistics).**

**16. *en.wikipedia.org.* [En línea] https://en.wikipedia.org/wiki/Ridge\_regression#:~:text=Ridge%20regression%20is%20a%20method,econometrics%2C%20chemistry%2C%20and%20engineering..**

**17. *corporatefinanceinstitute.com.* [En línea] https://corporatefinanceinstitute.com/resources/knowledge/other/elastic-net/.**

**18. *geeksforgeeks.org.* [En línea] https://www.geeksforgeeks.org/python-decision-tree-regression-using-sklearn/.**

**19. *wikipedia.org.* [En línea] https://es.wikipedia.org/wiki/RANSAC.**

**20. Wikipedia. [En línea] https://es.wikipedia.org/wiki/Web\_scraping.**

# Anexo 1: Otra funcionalidad del entorno web

# Anexo 2: Instalación del entorno

Para poder ejecutar nuestro proyecto, vamos a necesitar instalar apache como servidor. Pero, ¿qué es apache? Apache es un software de servidor web gratuito y de código abierto para plataformas Unix(Linux, Mac por ejemplo) con el cual se ejecutan el 46% de los sitios web de todo el mundo. Es mantenido y desarrollado por la Apache Software Foundation.

Le permite a los propietarios de sitios web o a un particular en su propio servidor o pc servir contenido en la web o de forma local, de ahí el nombre de servidor web. Es uno de los servidores web más antiguos y confiables. Su primera versión fue lanzada hace más de 20 años, en 1995.

Cuando alguien quiere visitar un sitio web, ingresa un nombre de dominio en la barra de direcciones de su navegador. Luego, el servidor web envía los archivos solicitados actuando como un repartidor virtual.

Sabiendo que es Apache, proseguimos con los pasos para realizar la instalación, en esta documentación el proyecto se ha realizado sobre un Mac, con lo cual los pasos para la instalación van a ser en lenguaje de consola de Mac OS

El comando de instalación que vamos a usar es:

brew install httpd

Una vez instalado vamos a indicar en el archivo httpd.conf cuál va a ser la ruta que debe abrir.

Primero abrimos el archivo para editarlo con el siguiente comando:

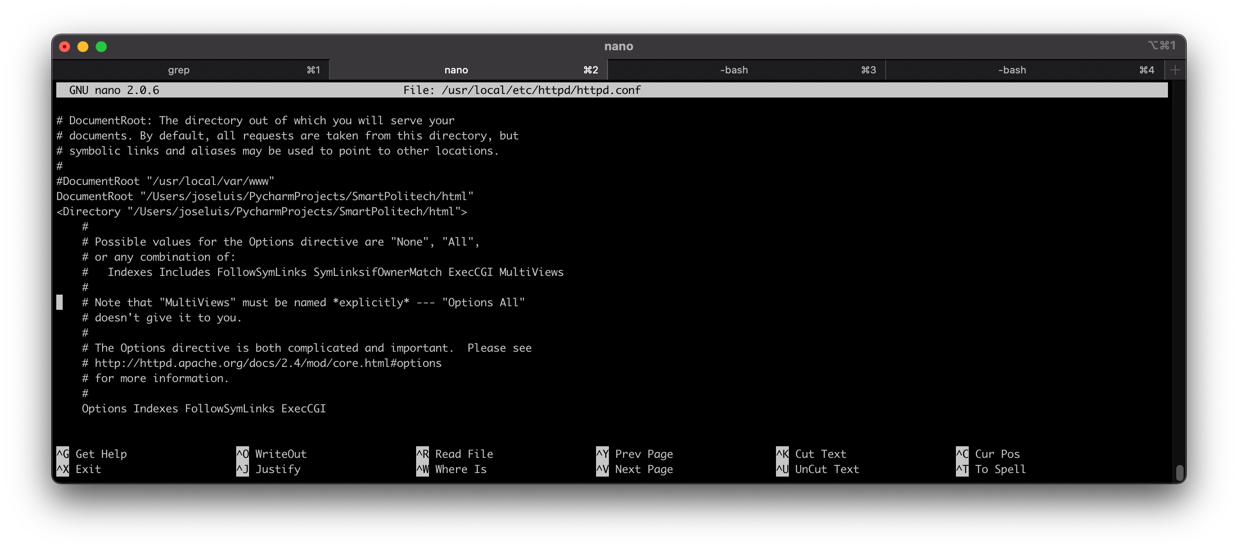
nano /usr/local/etc/httpd/httpd.conf

Bajamos hasta la linea donde pone:

# DocumentRoot: The directory out of which you will serve your

# documents. By default, all requests are taken from this directory, but

# symbolic links and aliases may be used to point to other locations.



Justo debajo de esta línea se indica donde se encuentra nuestra carpeta con el main principal. Por defecto aparecerá #DocumentRoot “/usr/local/var/www”, con lo cual, tendríamos que cambiar “/usr/local/var/www” por nuestra ruta. En mi caso será “/Users/joseluis/PycharmProjects/SmartPolitech/html”, quedando la linea así:

**#DocumentRoot “/Users/joseluis/PycharmProjects/SmartPolitech/html”**.

Una vez hecho esto, debajo de esta linea hay otra que también debemos modificar:

<Directory “/usr/local/var/www”> y haremos lo mismo, sustituiremos “/usr/local/var/www” por nuestra ruta, en mi caso quedando así:

**<Directory “/Users/joseluis/PycharmProjects/SmartPolitech/html”>**

Hecho esto, bajamos un poco hasta donde encontraremos el fragmento de código siguiente:

<IfModule dir\_module>

DirectoryIndex index.html

</IfModule>

Si nuestro main, se llama index.html, lo dejaremos como está, sin por el contrario se llama de otra forma, deberemos cambiarlo. En mi caso se llama main.html, por lo que mi fragmento de código quedaría así:

<IfModule dir\_module>

DirectoryIndex main.html

</IfModule>

Una vez terminado, se pulsa control + X para guardar los cambios y salir del archivo. Ya está configurado el archivo de configuración para poder ejecutar nuestro servidor local con el proyecto.

Ahora se tiene que levantar el servidor, para ello se usan los siguientes comandos:

Inicio de httpd: **brew services start httpd**

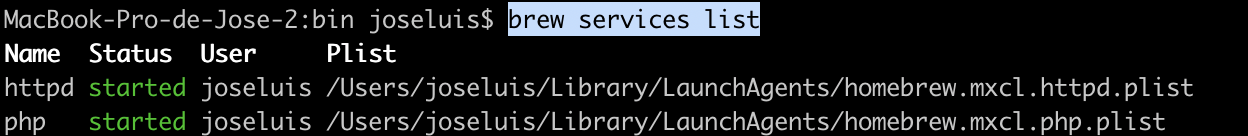
Interrupción de httpd: **brew services stop httpd**

Reinicio de httpd: **brew services restart httpd**

Para comprobar que se ha iniciado el servidor se puede usar el siguiente comando

**brew services list**

Este comando mostrará todos los servicios iniciados en el sistema y su estado.

****

Cómo se puede ver en la captura anterior está el servicio httpd en estado started.

# Anexo 3: Librerías usadas

Durante el Proyecto, se han usado varias librerías en los diferentes lenguajes con lo que se ha programado:

Librería XGBoost -> se instala con el comando pip3 install xgboost

Librería Sklearn -> se instala con el comando pip3 install sklearn

Librería Statsmodels -> se instala con el comando pip3 install statsmodels

Librería Bartpy -> se instala con el comando pip3 install bartpy

# 

1. Técnica utilizada mediante programas de software para extraer información de sitios web. (20) [↑](#footnote-ref-1)
2. Encuentre la diferencia de conjunto de dos arreglos. Devuelve los valores únicos en ar1 que no están en ar2. [↑](#footnote-ref-2)
3. Pendiente o inclinación de una superficie. [↑](#footnote-ref-3)
4. Surge cuando las variables explicativas del modelo están altamente correlacionadas entre sí. [↑](#footnote-ref-4)