



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

RESUMEN

Este proyecto aporta una herramienta para realizar predicciones de valores meteorológicos (temperatura, humedad o velocidad del viento) a partir de un histórico de datos, obtenido desde Enero de 2013. Para crear esta herramienta se ha hecho uso de Machine Learning, complementándolo con el uso de los dispositivos IoT, que sirven como base para conseguir los datos históricos.

Para realizar las predicciones, se han usado varios algoritmos de regresión de Machine Learning que son: Decision Tree Regressor, Elastic Net, Lasso, Linear Regression, Random Forest, Ridge, Robust Regression, Stochastic Gradient Descent, Support Vector Regression y Xgboost.

Estos algoritmos son entrenados mediante un dataset, que contiene como características: año, mes, día, punto de rocío, humedad, velocidad del viento y presión; y tiene como finalidad predecir el valor de la temperatura. Este dataset fue extraído de una página web, que recoge datos mediante dispositivos IoT. Ha sido comparado con los datos del proyecto SmartPolitech (que cuenta con varios sensores para también recoger datos de tipo meteorológico) para seleccionar el que mayor fiabilidad aportaba, ya que los algoritmos son muy sensibles a valores erróneos.

Para representar las predicciones, que devuelven los algoritmos, se ha creado un entorno Frontend/Backend, el cual se encarga de representar, visualmente, la línea de tiempo actual comparándola con la línea de predicción en graficas intuitivas.

Los resultados obtenidos por los algoritmos han sido bastante buenos; esto, además de por las métricas de error, se puede comprobar por como la línea de predicción sigue la tendencia que debe seguir la temperatura al pasar por las diferentes estaciones del año.

ÍNDICE GENERAL DE CONTENIDOS

1	INTRODUCCIÓN.....	1
2	OBJETIVOS.....	3
3	ESTADO DEL ARTE.....	4
3.1	MACHINE LEARNING APLICADO A IoT	4
3.2	ANTECEDENTES DE MACHINE LEARNING APLICADO A IoT	5
	<i>Caso 1.....</i>	<i>5</i>
	<i>Caso 2.....</i>	<i>7</i>
	<i>Caso 3.....</i>	<i>9</i>
	<i>Caso 4.....</i>	<i>10</i>
4	METODOLOGÍA	12
4.1	MACHINE LEARNING	12
4.2	INTERNET DE LAS COSAS (IoT).....	14
4.3	METODOLOGÍA SEGUIDA	16
5	IMPLEMENTACIÓN Y DESARROLLO	19
5.1	RECOLECCIÓN DE DATOS	19
5.2	SELECCIÓN DE FEATURES O ATRIBUTOS PARA EL MODELO DE ENTRENAMIENTO	22
5.3	CONJUNTO DE ENTRENAMIENTO Y PRUEBA	23
5.4	ALGORITMOS IMPLEMENTADOS.....	24
5.5	INICIALIZACIÓN DEL ALGORITMO	34
5.6	ENTRENAMIENTO DEL MODELO Y SERIALIZACIÓN DEL MODELO RESULTANTE.....	35
5.7	PREDICCIÓN DE DATOS	36
5.8	CREACIÓN DE LA BASE DE DATOS	37
5.9	CREACIÓN DEL FRONTEND Y EL BACKEND.....	38
5.10	REPRESENTACIÓN DE LOS DATOS	43
6	RESULTADOS Y DISCUSIÓN.....	45
7	CONCLUSIONES.....	58
8	REFERENCIAS BIBLIOGRÁFICAS.....	60
	ANEXO 1: OTRA FUNCIONALIDAD DEL ENTORNO WEB.....	62
	ANEXO 2: INSTALACIÓN Y EJECUCIÓN DEL SERVIDOR APACHE	64
	ANEXO 3: REPOSITORIO Y ESTRUCTURA DEL PROYECTO.....	67

ÍNDICE DE FIGURAS

FIGURA 1: METODOLOGÍA USADA EN EL CASO 1 6

FIGURA 2: METODOLOGÍA DEL PROYECTO 16

FIGURA 3: EJEMPLO FUNCIONAMIENTO RANDOM FOREST 26

ÍNDICE DE ILUSTRACIONES

ILUSTRACIÓN 1: RESULTADOS RMSE Y MAE DEL CASO 2 (5)	7
ILUSTRACIÓN 2: RESULTADOS R2 DEL CASO 2 (5).....	8
ILUSTRACIÓN 3: RESULTADOS MSE, RMSE, MAE Y RMAE DEL CASO 3.....	9
ILUSTRACIÓN 4: METODOLOGÍA DEL CASO 4	10
ILUSTRACIÓN 5: REPRESENTACIÓN DE PREDICCIÓN USANDO XGBOOST.....	18
ILUSTRACIÓN 6: FRAGMENTO DE CÓDIGO DEL PROCESO DE WEB SCRAPPING	20
ILUSTRACIÓN 7: TABLA DE DATOS DE LA WEB WEATHER UNDERGROUND (9)	21
ILUSTRACIÓN 8; FRAGMENTO DE DATOS DEL CSV	21
ILUSTRACIÓN 9: FRAGMENTO DE CÓDIGO DE LA LECTURA DEL CSV	21
ILUSTRACIÓN 10: FRAGMENTO DE CÓDIGO DE LA SELECCIÓN DE LOS FEATURES	22
ILUSTRACIÓN 11: AÑADIENDO LIBRERÍA PARA CREAR CONJUNTO DE ENTRENAMIENTO Y PRUEBA..	23
ILUSTRACIÓN 12: SEPARACIÓN DE DATOS EN XTRAIN E YTRAIN.....	23
ILUSTRACIÓN 13: DIVISIÓN DE DATOS EN UN CONJUNTO DE PRUEBA Y OTRO DE ENTRENAMIENTO	23
ILUSTRACIÓN 14: EJEMPLO LÍNEA Y PUNTOS REGRESIÓN LINEAL.....	25
ILUSTRACIÓN 15: EJEMPLO FUNCIONAMIENTO DECISION TREE REGRESSION	32
ILUSTRACIÓN 16: TABLA DE LA DB DONDE SE ALMACENA EL NOMBRE DE LOS ALGORITMOS.....	37
ILUSTRACIÓN 17: TABLA DE LA DB DONDE SE ALMACENA EL CONJUNTO DE DATOS.....	37
ILUSTRACIÓN 18: TABLA DE LA DB DONDE SE ALMACENAN LOS DATASET A PREDECIR	37
ILUSTRACIÓN 19: ESTRUCTURA DEL FRONTEND Y BACKEND	38
ILUSTRACIÓN 20: FILTROS DEL FRONTEND	39
ILUSTRACIÓN 21: FILTROS DEL ENTORNO WEB	43
ILUSTRACIÓN 25: EJEMPLO REPRESENTACIÓN GRÁFICA.....	44
ILUSTRACIÓN 28: GRÁFICA DECISION TREE REGRESOR	47
ILUSTRACIÓN 29: GRÁFICA ELASTIC NET REGRESSION	48
ILUSTRACIÓN 30: GRÁFICA LASSO REGRESSION	49
ILUSTRACIÓN 31: GRÁFICA LINEAR REGRESSION.....	50
ILUSTRACIÓN 32: GRÁFICA RANDOM FOREST	51
ILUSTRACIÓN 33: GRÁFICA RIDGE REGRESSION	52
ILUSTRACIÓN 34: GRÁFICA ROBUST REGRESSION RANSAC.....	53
ILUSTRACIÓN 35: GRÁFICA STOCHASTIC GRADIENT DESCENT	54
ILUSTRACIÓN 36: GRÁFICA SUPPORT VECTOR REGRESSION	55
ILUSTRACIÓN 37: GRÁFICA XGBOOST	56

ÍNDICE DE TABLAS

TABLA 1: MODELOS Y ALGORITMOS USADOS EN EL CASO 1	5
TABLA 2: TABLA DE PREDICCIÓN PARA CONJUNTO DE TEST	36
TABLA 3: TABLA COMPARATIVA MAE	57
TABLA 4: TABLA COMPARATIVA MSE Y RMSE.....	58

1 INTRODUCCIÓN

El proyecto *Modelos Predictivos asociados a IoT* nace como una herramienta para estimar valores, como pueden ser temperatura, humedad, velocidad del viento u otras variables de este tipo. Puede servir de ayuda para hacerse una idea de lo que puede pasar en un futuro, gracias al histórico de datos.

Para llevar a cabo el proyecto se usa Machine Learning como base y se complementa con la ayuda de dispositivos IoT, que se encargan de medir y almacenar datos que van a servir de ayuda para crear el histórico de datos que necesita el aprendizaje automático para trabajar.

Para realizar todo el proceso de predicción se ha hecho uso de varios algoritmos de regresión (Decision Tree Regressor, Elastic Net, Lasso, Linear Regression, Random Forest, Ridge, Robust Regression, Stochastic Gradient Descent, Support Vector Regression y Xgboost). También se han analizado varias web, donde se cuelgan valores diarios relacionados con el clima. Algunas web consultadas son Wundergroun.com (1) o timeanddate.com (2). Estas webs contienen toda la información necesaria para construir el Dataset. Este se construye mediante un proceso de Web Scrapping, que extrae los datos y los incluye en un csv. Además de estas web, también se ha analizado el dataset formado por el proyecto SmartPolitech, de la Escuela Politécnica de Cáceres, que también aporta valores meteorológicos gracias a la cantidad de sensores repartidos por toda la facultad.

Los datos históricos que se recogen en las webs y en el proyecto de SmartPolitech proceden de dispositivos IoT, que están captando las 24 horas datos y los van almacenando para su posterior análisis.

Se ha seguido una metodología de trabajo en la que se lleva a cabo un proceso de pasos, que resumiéndolos, consisten en: creación de un conjunto de datos históricos (con valores como temperatura, humedad, velocidad del viento, etc), procesado para eliminar valores incorrectos, selección de una serie de algoritmos de Machine Learning con los que realizar el método de regresión, obtener estimaciones mediante los algoritmos a partir del conjunto histórico, creación de una base de datos y, finalmente, desarrollo de Frontend y Backend.

El desarrollo del Frontend y Backend es algo necesario, ya que la comparación visual también aporta valor, no solo la comparación que se realice mediante el uso de métricas de error.

Como funcionalidad añadida, el entorno Web permitirá, a partir de un fichero, representar sus datos en las gráficas ya creadas.

La herramienta está basada en una aplicación web desarrollada en HTML5, CSS3 y JavaScript 6 en el frontend, y en PHP, MySQL y Python en el backend.

2 OBJETIVOS

El objetivo principal de este proyecto es desarrollar una herramienta capaz de realizar una estimación aproximada de valores futuros con el menor error posible a partir de una serie de datos históricos. Los resultados de estas predicciones se mostrarán en un entorno Web, mediante gráficas, para poder visualizarlos punto por punto (por día) e interactuar con ellos.

A parte de esto, también se ha propuesto como objetivo conseguir, que a partir de un archivo de datos, el entorno web represente en las gráficas su contenido.

Para llevar a cabo el objetivo final, se deben cumplir los siguiente subobjetivos:

- Estudio y análisis del mundo del Machine Learning y del funcionamiento de los dispositivos IoT.
- Estudio de los algoritmos de regresión que se han de implantar en el proyecto.
- Creación de un conjunto de datos histórico de datos.
- Obtener, a partir del histórico y mediante el uso de los algoritmos, unas predicciones.
- Implementación de un entorno Web para representar las predicciones, junto con el valor real para poder realizar comparaciones.
- Realizar un análisis de los resultados en base a unas métricas de error para decidir que algoritmo es mejor.

3 ESTADO DEL ARTE

Para realizar un análisis del estado del arte en el Machine Learning aplicado a IoT, es necesario introducir en qué consiste el Machine Learning y el IoT. También es fundamental llevar a cabo un resumen de otros estudios que se han realizado, anteriormente, en cuanto al trabajo conjunto de ambos.

3.1 Machine Learning aplicado a IoT

Actualmente, se pueden encontrar muchos ejemplos de Machine Learning aplicado a dispositivos IoT. Se puede ver, por ejemplo, en dispositivos que se encargan de realizar un reconocimiento facial, sensores de coches inteligentes o, incluso, predicciones del tráfico. Para todo esto, el Machine Learning hace uso de una serie de algoritmos que son los que se encargan de llevar a cabo toda la lógica.

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.

3.2 Antecedentes de Machine Learning aplicado a IoT

En este punto se describen 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:

Tabla 1: Modelos y algoritmos usados en el caso 1

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)

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

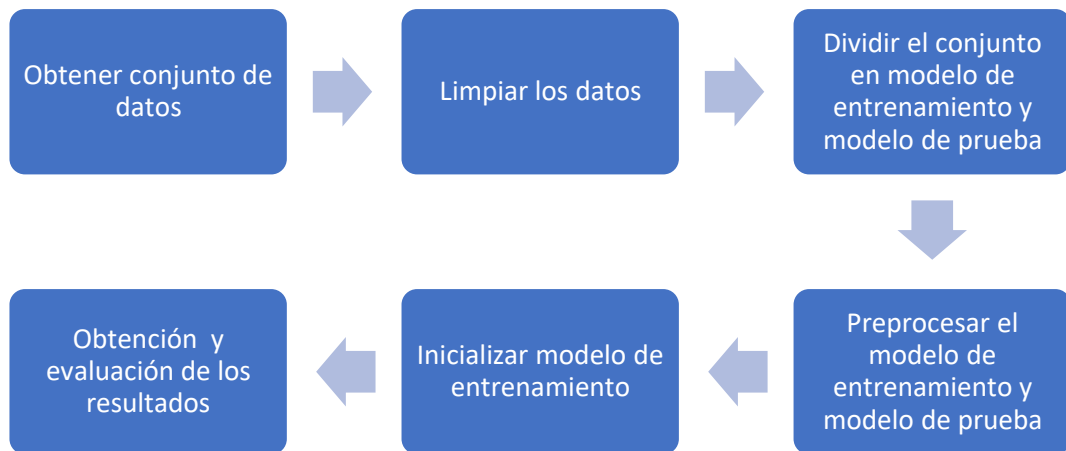


Figura 1: Metodología usada en el caso 1

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 1**, 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 Neighbors, y Hieraclhical cluster.

Como punto de partida, la metodología que se siguió es la misma que la de **Figura 1**.

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, NO_x, 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 recogen los resultados obtenidos para cada Dataset y algoritmo:

Techniques		Bagging		Random Committee		Random Forest		M5P Tree		KNN	
Data	Years	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
Alc	2013	8.31	10.99	8.01	10.83	7.65	10.20	8.48	11.19	8.87	12.33
	2014	8.03	10.65	7.66	10.26	7.33	9.77	8.66	11.47	9.43	13.07
Alc2	2014	8.22	10.97	8.00	10.78	7.68	10.24	8.58	11.41	8.87	12.33
Alj	2013	9.17	12.14	8.62	11.68	8.34	11.11	8.57	11.36	9.32	12.98
	2014	7.63	9.79	7.35	9.59	7.10	9.16	8.06	10.29	7.70	10.37
Lorca	2013	8.95	11.72	8.61	11.50	8.25	10.89	9.46	12.31	10.04	13.64
	2014	8.53	11.09	8.13	10.77	7.87	10.30	9.25	11.92	9.05	12.36
Car	2013	8.50	11.07	8.13	10.80	7.90	10.35	9.19	11.94	8.93	12.33
	2014	8.54	11.12	7.96	10.68	7.77	10.29	9.30	12.06	9.71	13.27

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 R^2 , 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:

Techniques		Bagging	Random Committee	Random Forest	M5P Tree	KNN
Datasets	Years	R^2	R^2	R^2	R^2	R^2
Alcantarilla	2013	0.895	0.898	0.910	0.891	0.870
	2014	0.908	0.911	0.920	0.900	0.870
Alcantarilla2	2014	0.913	0.919	0.927	0.899	0.871
Aljorra	2013	0.792	0.807	0.826	0.897	0.766
	2014	0.801	0.808	0.826	0.780	0.779
Lorca	2013	0.832	0.839	0.856	0.815	0.780
	2014	0.849	0.858	0.870	0.835	0.817
Caravaca	2013	0.679	0.695	0.722	0.626	0.616
	2014	0.742	0.761	0.780	0.752	0.645

Ilustración 2: Resultados R^2 del Caso 2 (5)

En este caso, lo sobrepasaron todos los algoritmos, ya que todos dieron un R^2 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.

	Proposed	RF	CART	BPNN	XGBoost	SVR
MSE	434.06	520.20	948.51	738.97	542.11	1174.73
RMSE	20.83	22.81	31.51	27.18	23.28	34.27
MAE	13.74	14.40	19.39	20.37	16.92	23.37
RMAE	3.71	3.79	4.40	4.51	4.11	4.83

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 artículo 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:

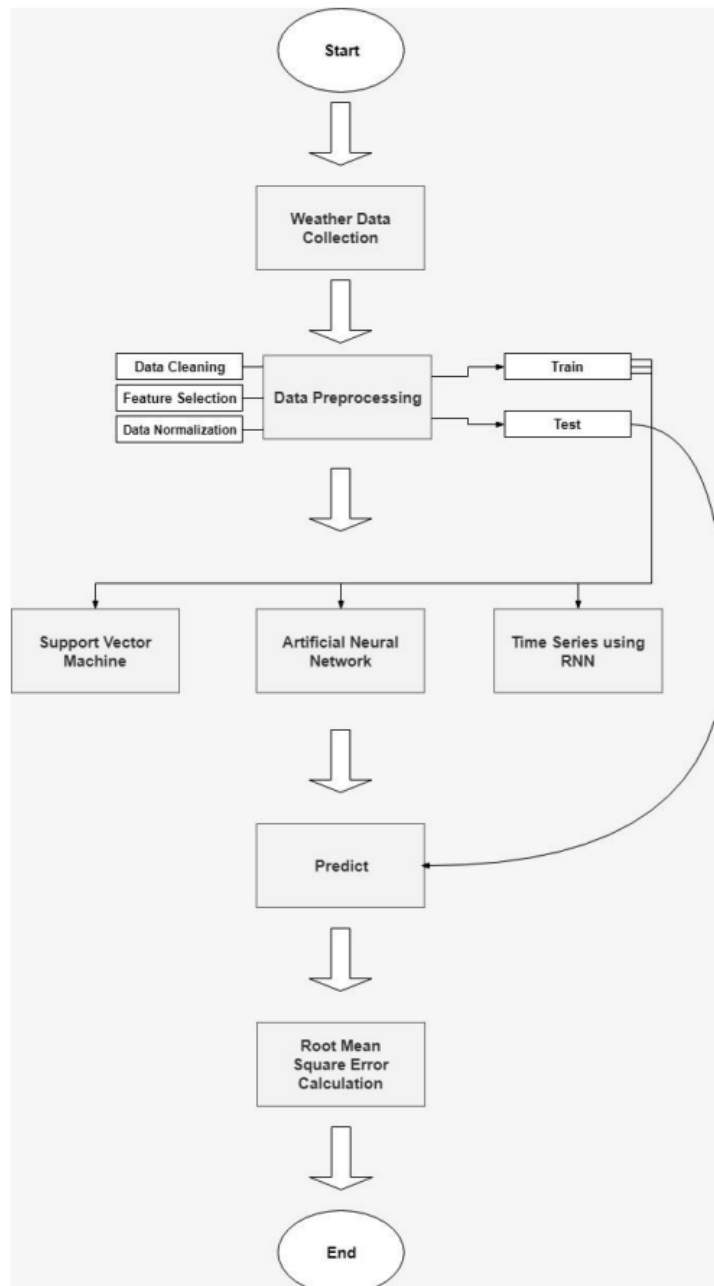


Ilustración 4: Metodología del caso 4

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 RMSE:

Modelo	Tiempo de predicción	RMSE
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 RMSE de 1,41.

4 METODOLOGÍA

En este apartado se va a explicar, por partes, en qué consisten y cómo funcionan el Machine Learning y los dispositivos IoT. También, se van a describir la serie de pasos que se han realizado para implementar el caso de estudio de este trabajo.

4.1 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 la máquina, una vez tiene el conjunto de datos, es escoger una fórmula matemática que se aplica al conjunto de datos o input de entrenamiento, que 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:

$$\frac{\sum_{i=1}^n (P_i - O_i)^2}{n}$$

Donde P_i son los valores predichos, O_i los valores reales y n el número de muestras totales.

La fórmula del RMSE es:

$$\sqrt{\frac{\sum_{i=1}^n (P_i - O_i)^2}{n}}$$

Al igual que en el MSE, P_i son los valores predichos, O_i 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; por ejemplo, la temperatura real frente a la temperatura prevista.

La fórmula del MAE es:

$$\frac{\sum_{i=1}^n |P_i - O_i|}{n}$$

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

4.2 Internet de las cosas (IoT)

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.

4.3 Metodología seguida

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 muestran los pasos que se han ido llevando a cabo para el desarrollo de todo el análisis

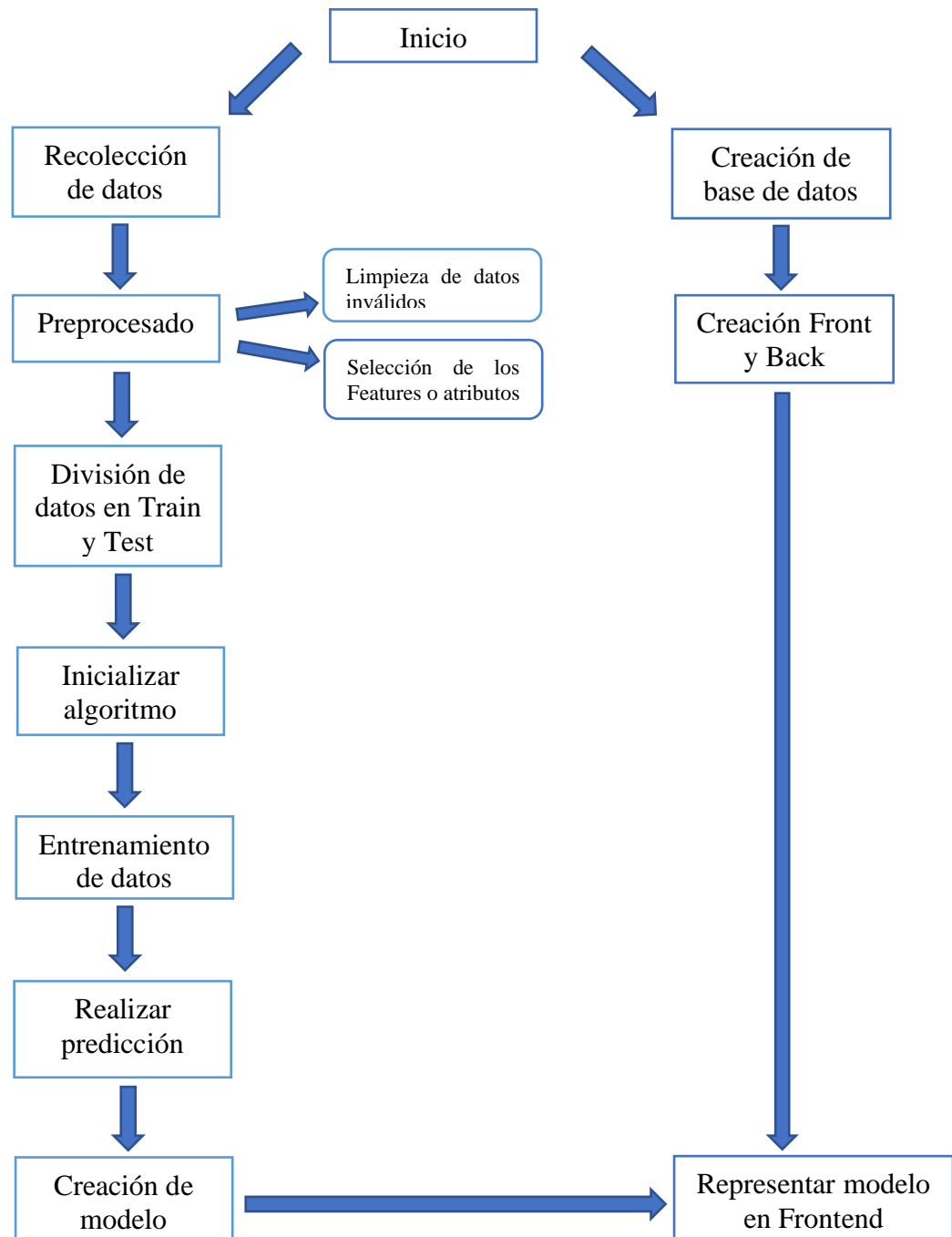


Figura 2: Metodología del proyecto

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 pasos 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 al 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 Backend y un Frontend 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 Backend y representada por el Frontend.

-Creación Frontend y Backend: se crea un Backend 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 Frontend, este último creado mediante el uso de HTML, CSS y JavaScript

-Representar modelo en Frontend: una vez se han realizado todos los pasos de la metodología desde el JavaScript del Frontend, se hace un solicitud al PHP del Backend, que a su vez hace una solicitud a Python y devuelve una predicción con los datos que ha recogido del formulario del Frontend. 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:

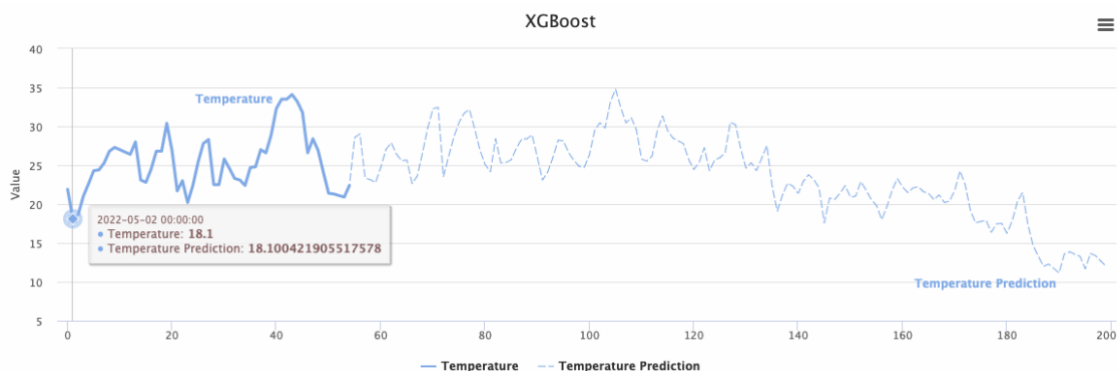


Ilustración 5: Representación de predicción usando XGBoost

5 IMPLEMENTACIÓN Y DESARROLLO

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

5.1 Recolección de datos

El primer paso de todos es 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 han extraído datos de la página web Weather Underground (9). Para llevar a cabo dicha extracción se ha realizado un proceso de Web Scrapping¹ 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 (se puede ver el código completo en **Anexo 4**).

¹ Técnica utilizada mediante programas de software para extraer información de sitios web. (22)

```

def run(driver, list_atr, year, month):
    sleep(5)
    acceptCookies(driver)
    changeToCelsius(driver)
    table = getTableInfo(driver)
    return operateTable(table, list_atr, year, month)

def main():
    driver = initWebdriver()
    years = ["2013", "2014", "2015", "2016", "2017", "2018", "2019", "2020", "2021", "2022"]
    months = ["01", "02", "03", "04", "05", "06", "07", "08", "09", "10", "11", "12"]
    list_atr = ["Year", "Month", "Day", "Temperature", "Dew Point", "Humidity", "Wind Speed", "Pressure"]
    for year in years:
        for month in months:
            openUrl(driver, "https://www.wunderground.com/history/monthly/es/badajoz/LEBZ/date/"
                           + year + "-" + month)
            list = run(driver, list_atr, year, month)
    driver.close()
    clear_list(list)
    np.savetxt("prueba.csv", list, delimiter=",", fmt="% s")

main()

```

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 *list_atr*, esta indica qué 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 consiste 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.

Daily Observations

Time	Temperature (°C)			Dew Point (°C)			Humidity (%)			Wind Speed (km/h)			Pressure (hPa)			Precipitation (mm)
Jan	Max	Avg	Min	Max	Avg	Min	Max	Avg	Min	Max	Avg	Min	Max	Avg	Min	Total
1	10	9.3	9	9	8.7	8	100	95.7	87	9	3.7	0	1001.6	999.6	998.7	0.0
2	7	5.3	2	7	5.3	2	100	100.0	100	6	4.7	2	1006.5	1005.5	1004.5	0.0
3	5	5.0	5	5	5.0	5	100	100.0	100	6	6.0	6	1011.4	1011.4	1011.4	0.0
5	1	1.0	1	1	1.0	1	100	100.0	100	4	2.0	0	1013.3	1013.0	1012.4	0.0
9	12	8.8	8	10	7.6	7	93	92.0	88	7	3.8	2	1006.5	1005.5	1004.5	0.0
11	13	12.5	12	12	12.0	12	100	97.0	94	2	2.0	2	1003.6	1003.1	1002.6	0.0
12	14	14.0	14	10	10.0	10	77	77.0	77	15	15.0	15	1000.6	1000.6	1000.6	0.0
16	13	10.8	10	11	9.0	8	94	88.6	87	26	19.0	15	999.6	998.5	994.8	0.0
17	14	14.0	14	13	13.0	13	94	94.0	94	17	17.0	17	996.7	996.7	996.7	0.0
18	16	16.0	16	14	13.5	13	88	85.0	82	31	29.5	28	989.9	989.9	989.9	0.0
19	10	10.0	10	7	7.0	7	82	82.0	82	46	46.0	46	965.4	965.4	965.4	0.0
20	12	10.0	5	11	8.0	4	94	87.8	82	37	19.8	6	987.9	985.6	983.0	0.0
22	11	8.4	5	8	6.0	4	93	85.0	66	54	33.9	13	991.8	990.8	989.9	0.0
23	12	11.3	11	10	9.3	9	94	88.0	82	24	22.0	20	997.7	997.7	997.7	0.0
24	12	12.0	12	10	10.0	10	88	88.0	88	20	20.0	20	996.7	996.7	996.7	0.0
25	10	10.0	10	10	10.0	10	100	100.0	100	7	5.7	4	994.8	994.8	994.8	0.0
27	14	14.0	14	13	12.6	12	94	91.6	88	24	17.2	13	998.7	997.9	997.7	0.0
29	3	2.7	2	3	2.7	2	100	100.0	100	0	0.0	0	1007.5	1007.5	1007.5	0.0
31	7	4.7	3	7	4.7	3	100	100.0	100	2	0.6	0	1010.4	1010.4	1010.4	0.0

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:

Year	Month	Day	Temperature	Humidity	Wind Speed	Pressure	Precipitation
2013		1	9.3	95.7	3.7	999.6	0.00
2013		1	5.3	100.0	4.7	1005.5	0.00
2013		1	5.0	100.0	6.0	1011.4	0.00
2013		1	1.0	100.0	2.0	1013.0	0.00

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.

```
data = pandas.read_csv(path_to_project + '/crones/file.csv', header=0)
dataset = data.drop_duplicates()
```

Ilustración 9: Fragmento de código de la lectura del CSV

5.2 Selección de Features o atributos para el modelo de entrenamiento

Una vez se ha creado el CSV con los datos muestrales, se seleccionan las Features que se usarán para crear el modelo de entrenamiento.

Como se comentaba en el apartado **INTRODUCCIÓN**, en este proyecto se realiza 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 rocío. Con lo cual, se usan todos los campos (excepto el campo de temperatura) como Features de los diferentes algoritmos.

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

```
feature_names = dataset.columns.values  
  
features_to_delete = np.array(['Temperature'])  
feature_names = np.setdiff1d(feature_names, features_to_delete)
```

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 el campo temperatura y se guardan los demás.

² Encuentra la diferencia de conjunto de dos arreglos. Devuelve los valores únicos en ar1 que no están en ar2.

5.3 Conjunto de entrenamiento y prueba

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

```
from sklearn.model_selection import train_test_split
```

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

A continuación, mediante las siguientes dos líneas se obtienen en dos variables: por un lado, los datos correspondientes a la columna que se va a predecir (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).

```
y_train = dataset.filter(items=['Temperature'])  
x_train = dataset.filter(items=feature_names)
```

Ilustración 12: Separación de datos en xtrain e ytrain

Teniendo esto claro, se dividen 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` e `y_train` en un conjunto de prueba y otro de entrenamiento. 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.

```
x_train, x_test, y_train, y_test = train_test_split(x_train, y_train, shuffle=True, test_size=0.2, random_state=123)
```

Ilustración 13: División de datos en un conjunto de prueba y otro 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.

5.4 Algoritmos implementados

En esta sección se describen 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 muestran 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 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.

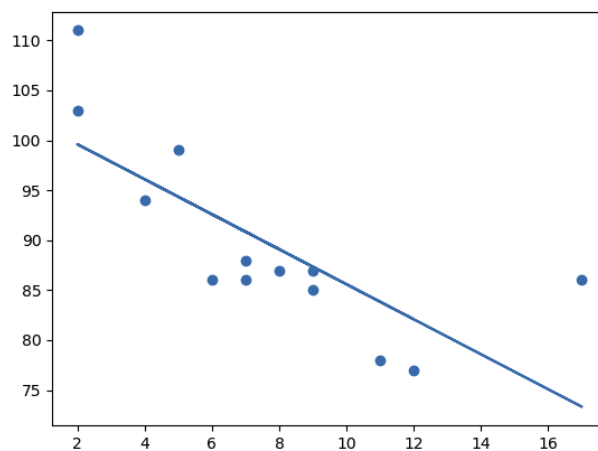


Ilustración 14: Ejemplo línea y puntos Regresión Lineal

Sabiendo todo esto, se puede 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 este 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 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:

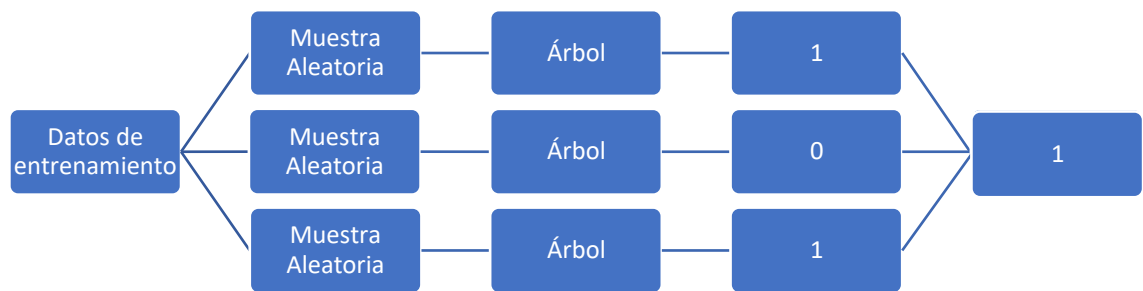


Figura 3: Ejemplo funcionamiento Random Forest

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, se puede 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 este 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 un 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, 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, se puede 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 este 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 gradiente³ 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 va moviéndose por su pendiente en varios pasos hasta que consigue alcanzar el punto más bajo de esa función.

Sabiendo todo esto, se puede 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 este comando se consigue una muestra de predicción:

```
preds = sgd_reg.predict("Parámetros")
```

Siendo preds el resultado.

³ Pendiente o inclinación de una superficie.

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 adecuada para modelos que muestran altos niveles de multicolinealidad⁴ o cuando se desea automatizar ciertas partes de la selección del modelo, como la selección de variables o eliminación de parámetros.

Sabiendo todo esto, se puede 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 este comando se consigue una muestra de predicción:

```
preds = lasso.predict("Parámetros")
```

Siendo preds el resultado.

⁴ Surge cuando las variables explicativas del modelo están altamente correlacionadas entre sí.

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, se puede 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 este 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, se puede 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 este 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 un 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.

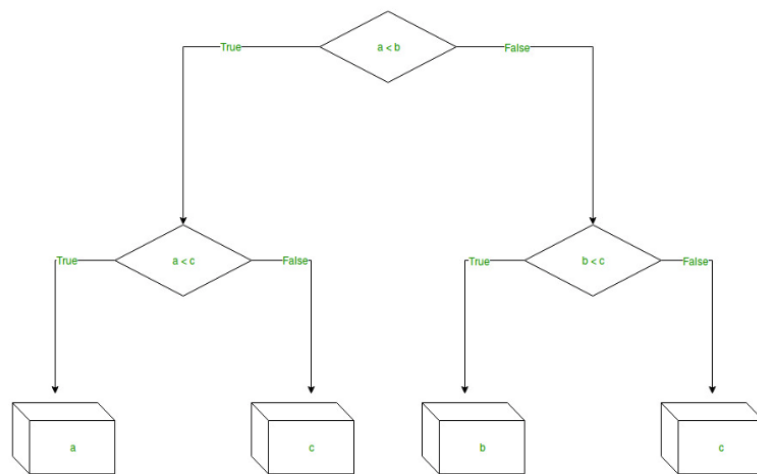


Ilustración 15: Ejemplo funcionamiento Decision Tree Regression

Como se puede ver, en cada nodo se van tomando una serie de decisiones hasta llegar al final. Sabiendo todo esto, se puede 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 este 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 forme 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, se puede 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 este comando se consigue una muestra de predicción:

```
preds = ransac_reg.predict("Parámetros")
```

Siendo preds el resultado.

5.5 Inicialización del algoritmo

En esta sección se explica cómo se inicializa un algoritmo. En la sección anterior se ha podido ver como para cada algoritmo había una serie de comandos 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:

```
import xgboost as xgb
```

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

```
xg_reg = xgb.XGBRegressor(objective='reg:squarederror', min_child_weight=0, colsample_bytree=0.8, subsample=0.8,  
learning_rate=0.1, max_depth=9, n_estimators=1000)
```

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.

5.6 Entrenamiento del modelo y serialización del modelo resultante

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:

```
xg_reg.fit(x_train, y_train)
```

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

Para poder exportar el modelo resultante se puede hacer uso de la librería *Picke* (20), esta librería permite la serialización de los datos de una variable para poder exportar los datos en un objeto y poder cargarlos en otro proyecto incluso o poder usar el modelo para realizar más predicciones en otro momento.

5.7 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:

```
preds = xg_reg.predict(x_test, ntree_limit=0)
```

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

Tabla 2: Tabla de predicción para conjunto de test

Día	Mes	Humedad	Punto de rocío	Presión	Velocidad del viento	Temperatura predicha por el algoritmo
17	1	78.3	2.3	1006.4	2.1	7.93
24	3	63.9	8.9	989.7	21.1	11.82
12	5	43.1	7.2	992.4	11.2	25.15
31	5	62.7	11.2	996.7	10.2	21.82
3	9	50.1	12.8	994.0	11.2	25.45
14	10	54.4	9.4	997.3	2.9	20.55
16	10	66.1	11.3	999.9	2.4	18.26
27	10	55.5	8.8	996.1	5	18.10

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

5.8 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 Frontend.

id	name_algorithm
1	XGBoost
2	Linear Regression
4	Random Forest
5	Support Vector Regression
6	Elastic Net Regression
7	Stochastic Gradient Descent
8	LASSO Regression
9	Ridge Regression
10	Decision Tree Regressor
11	Robust Regression RANSAC

Ilustración 16: Tabla de la db donde se almacena el nombre de los algoritmos

-*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.

id	date	year	month	day	temperature	humidity	wind_speed	pressure	precipitation
1	2013-01-01 00:00:00	2013	1	1	9.3	95.7	3.7	999.6	0
2	2013-01-02 00:00:00	2013	1	2	5.3	100	4.7	1005.5	0
3	2013-01-03 00:00:00	2013	1	3	5	100	6	1011.4	0
4	2013-01-05 00:00:00	2013	1	5	1	100	2	1013	0
5	2013-01-09 00:00:00	2013	1	9	8.8	92	3.8	1005.5	0

Ilustración 17: Tabla de la db donde se almacena el conjunto de datos

-*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.

id	name_dataset	model_trainer_field	predict_field
2	Temperature	["Serie", "Hora", "Dia", "Mes"]	Temperature

Ilustración 18: Tabla de la db donde se almacenan los dataset a predecir

5.9 Creación del Frontend y el Backend

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 backend se usa PHP y Python combinados. PHP se usa como intermediario para la comunicación entre Python y JavaScript. Cuando se realiza una función en el Frontend, que requiere de alguna modificación, el JavaScript (mediante Ajax) lanza la petición a PHP con los datos del formulario, que sirven 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 van 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:

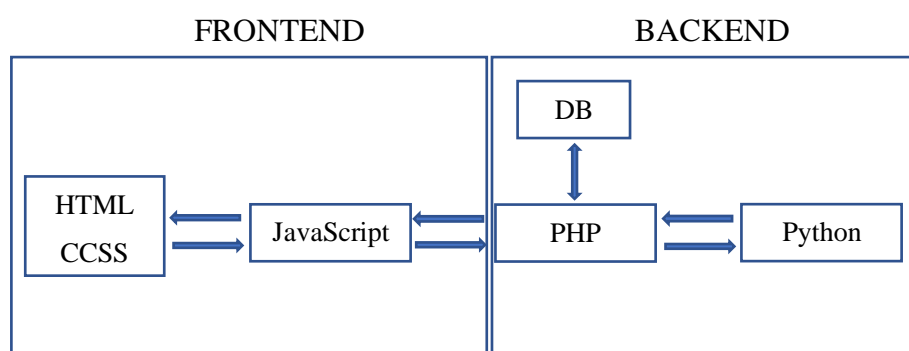


Ilustración 19: Estructura del Frontend y Backend

A continuación, se explica, detalladamente, como se comunican los distintos niveles a través de código:

-**HTML/CSS**: en este primer nivel se implementa el código de lo que el usuario va a ver cuándo accede al entorno web. Aquí se encuentran definidas todas las etiquetas necesarias para dar funcionalidad a la página posteriormente con JavaScript. Gracias a CSS se puede dar estilo a la página y ofrecer una vista intuitiva al usuario final. Por ejemplo, en la página principal se muestran los siguientes filtros:

Filters

Dataset Temperature	Days Prediction 200	Date 2022-02-28	Algorithms All Algorithms XGBoost Linear Regression Random Forest
------------------------	------------------------	--------------------	---

Refresh data

Show configuration options

Ilustración 20: Filtros del Frontend

Aquí debajo se muestra el código que los define:

```
<div class="col-lg-2 col-md-6 col-sm-12 m-b">
  <label>Dataset</label>
  <select class="form-control chosen" id='dataset' data-placeholder="Choose an option please">
</select>
</div>
<div class="col-lg-2 col-md-6 col-sm-12 m-b">
  <label>Days Prediction</label>
  <select class="form-control chosen" id='days_prediction' data-placeholder="Choose an option please">
</select>
</div>
<div class="col-lg-2 col-md-6 col-sm-12 m-b">
  <label>Date</label>
  <input class="form-control active" id="date_picker" type="text" name="daterange">
</div>
<div class="col-lg-3 col-md-6 col-sm-12 m-b">
  <label>Algorithms</label>
  <select class="custom-select" multiple id="algorithms" type="text" name="Choose an option please">
</select>
</div>
```

-JavaScript: se utiliza para realizar las operaciones como enviar y recibir información del Backend con la ayuda de Ajax. Además, también se usa en el HTML para rellenar algunos contenidos de las etiquetas.

A continuación, por ejemplo, se muestra el fragmento usado para pintar las gráficas. Para evitar la duplicidad de código se puede usar un mismo fragmento con JavaScript para pintarlas todas.

La etiqueta declarada en HTML es el `<div>` con id *graphs* que es posteriormente usado por JavaScript.

```
<div class="row" id="graphs" style='display:none'></div>
```

En JavaScript se usa el siguiente código:

```
function newContainer(index) {

    var html = "<div class='col-lg-6' id='graph" + index + "'>\n" +
        "    <div class='ibox-content' style='margin-top: 20px;'>\n" +
        "        <div class='highcharts-figure'>\n" +
        "            <div id='container" + index + "'></div>\n" +
        "        </div>\n" +
        "    </div>\n" +
        "</div>"

    $("#graphs").append(html);

}
```

Esta función crea, dinámicamente, un código HTML para cada gráfico y se lo va insertando a la etiqueta *graphs*.

Además, también con JavaScript, se envían los datos recogidos del formulario mostrado por el Front al Back. Al pulsar el botón *Refresh data*, se inicia todo el proceso de recogida de datos de los campos del formulario. Mediante el siguiente fragmento de código se realiza la petición a :

```
$.ajax({
    url: "main.php",
    type: "GET",
    data: data,
```

```

dataType: "json",
submit: "true",
success: function (response) {
    }
});

```

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, se indica que se va a recibir una respuesta.
- PHP**: el código que se ejecuta aquí ya se encarga de realizar las conexiones a la base de datos y también de comunicarse con Python para obtener las predicciones.

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

```

if (isset($_FILES['file'])) {
    $file = $_FILES['file']['tmp_name'];
}

```

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 procesamiento 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 (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:

```

$sql = "SELECT avg(dew_point),
            avg(humidity),

```



```

        avg(wind_speed),
        avg(pressure)
    FROM SmartPolitech.smart_data
    WHERE month=$month
        AND day=$day
        AND year > '2020';
$result = $this->db->select_data($sql);

```

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

Para comunicarse con Python se usa el comando de PHP *shell_exec* y se le pasa por parámetro la locación del archivo de Python que se quiere ejecutar junto con el array de datos creado anteriormente como argumento. El código es el siguiente:

```

$array_data = json_encode($array_data);

$command = "cd .. && cd algorithms/$this->type/model/ && /usr/local/bin/python3
charge_model.py " . $array_data;

$result = shell_exec($command);

```

Este comando ejecutará el archivo de Python del algoritmo que se escogió en el filtro del Front y devolverá en la variable *result*, el resultado.

-Python: se encarga de ejecutar tanto el entrenamiento del algoritmo de Machine Learning, como la carga de los datos una vez entrenados.

Partiendo de que ya está realizado el entrenamiento y creado el objeto serializado, se usa el comando *load* (de la librería *Pickle*) El comando es:

```
var = pickle.load()
```

```

name = 'train.dat'

xg_reg = pickle.load(open(name, "rb"))

```

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*:

```
var = model.predict()
```

```
data = json.loads(sys.argv[-1])

dataset = pd.DataFrame(data=data)

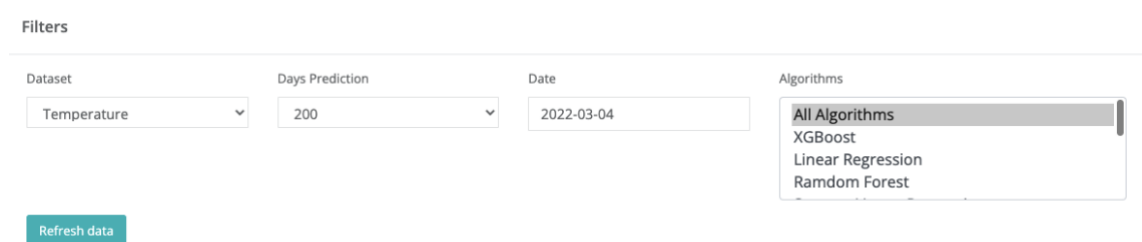
preds = xg_reg.predict(dataset[0:])
```

Esto genera un valor que corresponde con la temperatura predicha para el día que seleccionó en el formulario. Esta temperatura es devuelta a PHP, que a su vez la devuelve a la petición de Ajax.

5.10 Representación de los datos

En esta sección se muestra un ejemplo de representación de los datos en el Frontend.

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



Filters

Dataset	Days Prediction	Date	Algorithms
Temperature	200	2022-03-04	All Algorithms XGBoost Linear Regression Random Forest

Refresh data

Ilustración 21: Filtros del entorno web

-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.

Al pulsar botón el *Refresh data*, se inicia todo el proceso para obtener una predicción.

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.

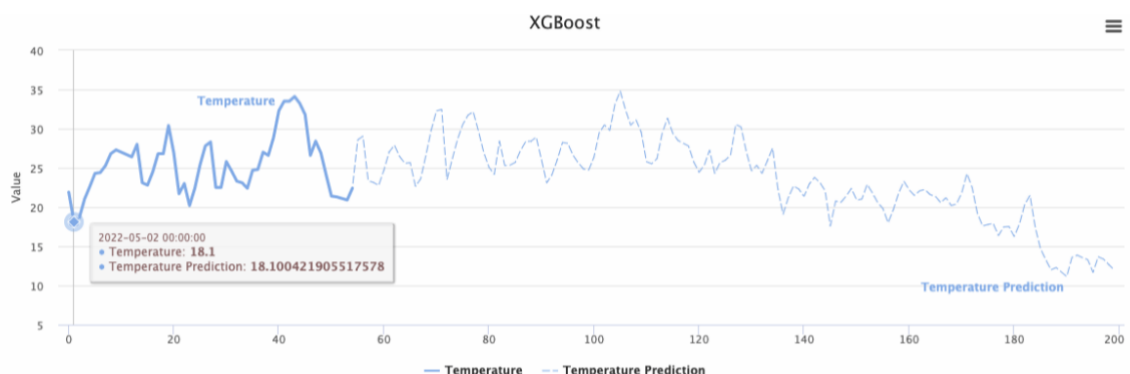


Ilustración 22: Ejemplo representación gráfica

6 RESULTADOS Y DISCUSIÓN

En esta sección se realiza una comparación de todos los algoritmos y, para ello, se utilizan 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):

```
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
```

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:

```
def calculate_mean_absolute_error(x_train, y_train, model):
    preds = model.predict(x_train)
    mae = mean_absolute_error(y_train, preds)
    print("MAE of SGD: % f" % (mae))
```

```
def calculate_mean_squared_error(x_train, y_train, model):  
    preds = model.predict(x_train)  
    mse = mean_squared_error(y_train, preds, squared=True)  
    print("MSE of SGD: % f" % (mse))  
    rmse = mean_squared_error(y_train, preds, squared=False)  
    print("RMSE of SGD: % f" % (rmse))
```

Los pasos que siguen las funciones son:

Con el modelo, 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 la función *mean_squared_error*, pasando por parámetros el conjunto resultado de la predicción y el conjunto de test con la temperatura real. Esto compara ambos conjuntos y nos devuelve 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 muestran capturas de una gráfica para cada algoritmo, junto con el resultado de las métricas. Posteriormente, podemos observar 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 puede 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

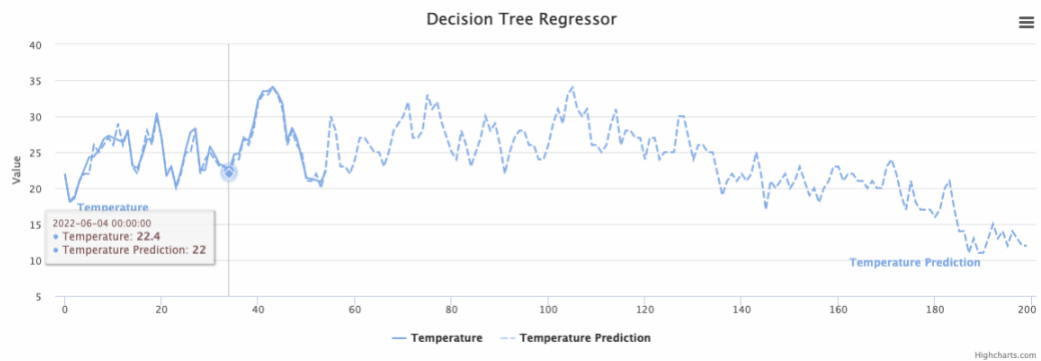


Ilustración 23: Gráfica Decision Tree Regresor

MAE	MSE	RMSE
0.835799	1.750000	1.322876

Este algoritmo, como ya se explicaba en un apartado anterior, es un algoritmo basado en árboles de decisión. Para series temporales, los algoritmos de árboles de decisión son uno de las mejores opciones.

MAE: se puede comprobar que el error medio absoluto está por debajo de un grado de temperatura entre los datos predichos y los datos observados, con lo cual es un buen resultado.

MSE: para el error cuadrático medio, los resultados no son tan buenos, comparados con otros algoritmos, esto es debido a que para algunos días no es buena la predicción. El valor real y el predicho son bastante diferentes para esos días, lo que hace que al elevar al cuadrado la diferencia entre ambos, el resultado sea excesivamente elevado y realizando la media de todos los puntos el resultado es alto por culpa de lo expuesto anteriormente.

Elastic Net Regression

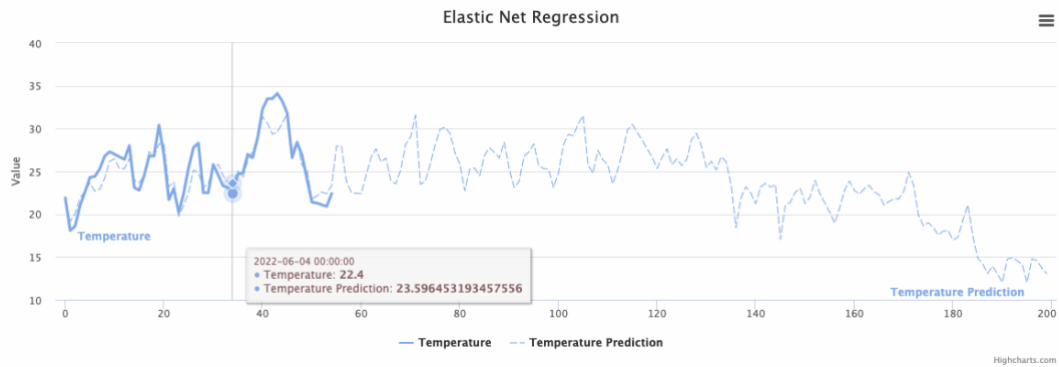


Ilustración 24: Gráfica Elastic Net Regression

MAE	MSE	RMSE
0.923130	1.587156	1.259824

Este algoritmo está basado en la regresión lineal. Para series temporales, los algoritmos de este tipo no son los mejores, ya que donde mejores resultados dan son en conjunto de datos lineales.

MAE: se puede comprobar que el error medio absoluto está por debajo de un grado de temperatura entre los datos predichos y los datos observados, con lo cual es un buen resultado, aun siendo un algoritmo de regresión lineal.

MSE: al igual que con el MAE, para el error cuadrático medio, los resultados no son buenos, esto es debido a lo que se comentaba un poco más arriba, al ser un algoritmo de regresión de lineal, los resultados con series temporales no suelen ser buenos.

Lasso Regression

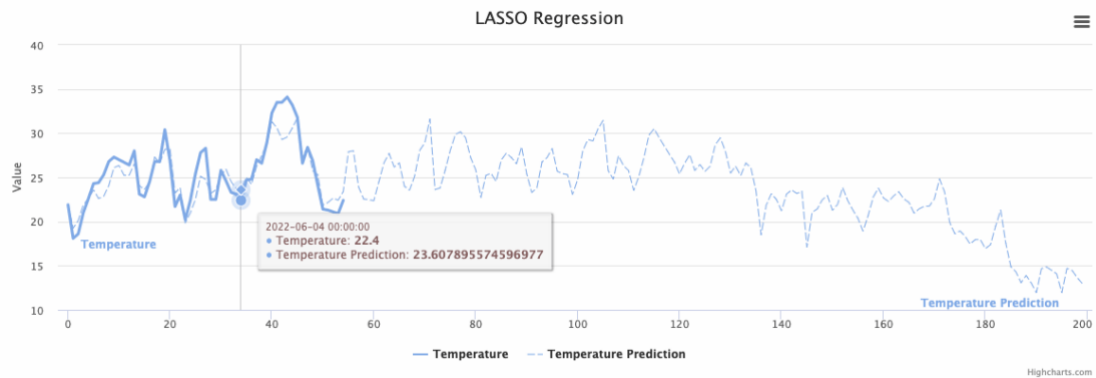


Ilustración 25: Gráfica Lasso Regression

MAE	MSE	RMSE
0.932046	1.602886	1.266052

Este algoritmo, al igual que Elastic Net, ha sido creado usando la base del algoritmo de regresión lineal. Para series temporales, los algoritmos de este tipo no son los mejores, ya que estos algoritmos donde mejores resultados dan son en conjunto de datos lineales.

MAE: se puede comprobar que el error medio absoluto está por debajo de un grado de temperatura entre los datos predichos y los datos observados, con lo cual es un buen resultado, aun siendo un algoritmo de regresión lineal. Se puede también comprobar que los valores son prácticamente iguales a los que devuelve Elastic Net o Linear Regression.

MSE: al igual que con el MAE, para el error cuadrático medio, los resultados no son buenos, esto es debido a lo que se comentaba un poco más arriba, al ser un algoritmo de regresión de lineal, los resultados con series temporales no suelen ser buenos.

Linear Regression

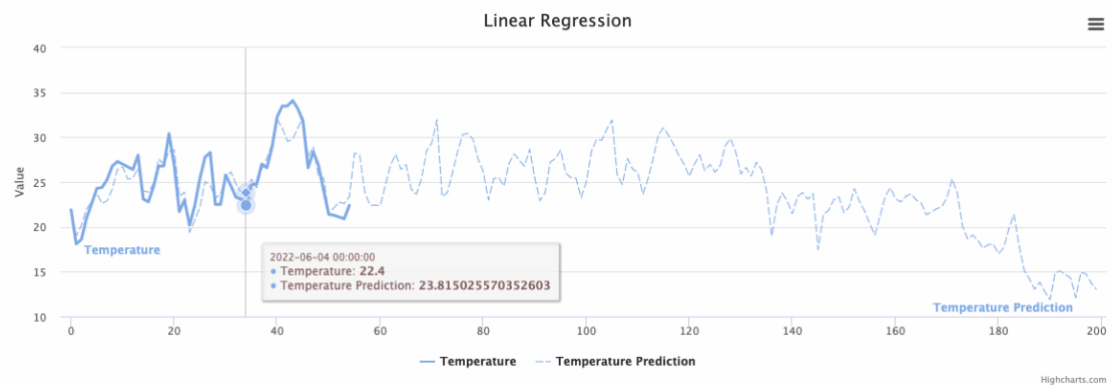


Ilustración 26: Gráfica Linear Regression

MAE	MSE	RMSE
0.880313	1.466668	1.211060

Este algoritmo usa la técnica de regresión lineal para realizar las estimaciones a partir de los valores reales. Para series temporales, los algoritmos de este tipo no son los mejores, ya que donde mejores resultados dan son en conjunto de datos lineales.

MAE: se puede comprobar que el error medio absoluto está por debajo de un grado de temperatura entre los datos predichos y los datos observados, con lo cual es un buen resultado, aun siendo un algoritmo de regresión lineal. Se puede también comprobar que los valores son prácticamente iguales a los que devuelve Elastic Net o Lasso Regression

MSE: al igual que con el MAE, para el error cuadrático medio, los resultados no son buenos, esto es debido a lo que se comentaba un poco más arriba, al ser un algoritmo de regresión lineal, los resultados con series temporales no suelen ser buenos.

Random Forest

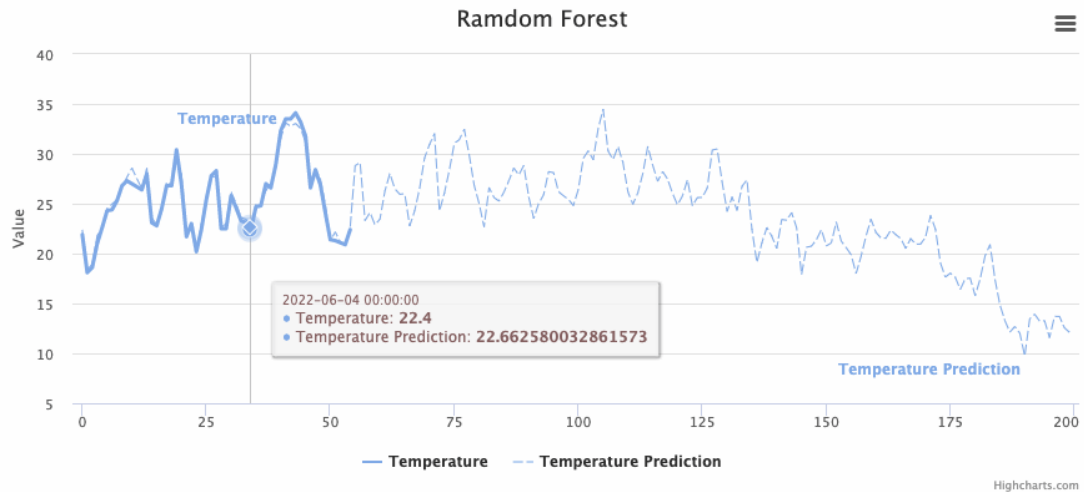


Ilustración 27: Gráfica Random Forest

MAE	MSE	RMSE
0.447718	0.616301	0.785048

Este algoritmo, como ya se explicaba en un apartado anterior, es un algoritmo basado en árboles de decisión. Además, es una mejora del algoritmo de árboles de decisión, ya que crea varios árboles para poder llegar a resultado final. Esto hace que el resultado que devuelve sea más real. Para series temporales, los algoritmos de árboles de decisión son uno de las mejores opciones.

MAE: se puede comprobar que el error medio absoluto está por debajo del medio grado de temperatura entre los datos predichos y los datos observados. Es uno de los mejores resultados entre los algoritmos implementados.

MSE: para el error cuadrático medio, los resultados también son bastante buenos, quedando por debajo del grado de diferencia.

Se puede comprobar, por tanto, comparando con los resultados de Decision Tree Regressor, que el uso de varios árboles conlleva, como dice la teoría, a conseguir mejores resultados.

Ridge Regression

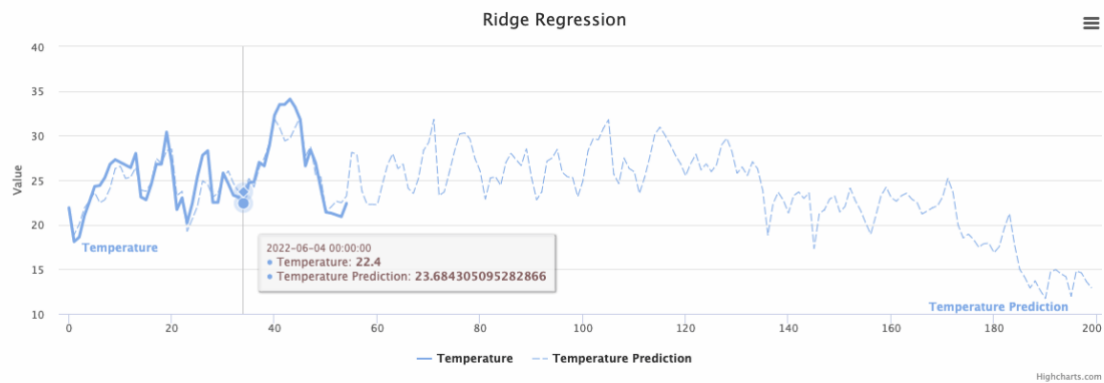


Ilustración 28: Gráfica Ridge Regression

MAE	MSE	RMSE
0.880319	1.466679	1.211065

Este algoritmo, al igual que Elastic Net y Lasso, ha sido creado usando la base del algoritmo de regresión lineal. Como ya se comentaba anteriormente, para series temporales, los algoritmos de este tipo no son los mejores. Estos algoritmos donde mejores resultados obtienen son en conjunto de datos lineales. Además, se puede ver en la gráfica que se muestra, que la línea de tiempo real tiene bastantes diferencias con la línea de predicción.

MAE: se puede comprobar que el error medio absoluto está por debajo de un grado de temperatura entre los datos predichos y los datos observados, con lo cual es un buen resultado, aun siendo un algoritmo de regresión lineal. Se puede también comprobar que los valores son prácticamente iguales a los que devuelve Elastic Net, Lasso o Linear Regression.

MSE: al igual que con el MAE, para el error cuadrático medio, los resultados no son buenos, esto es debido a lo que se comentaba un poco más arriba, al ser un algoritmo de regresión lineal, los resultados con series temporales no suelen ser buenos.

Robust Regression RANSAC

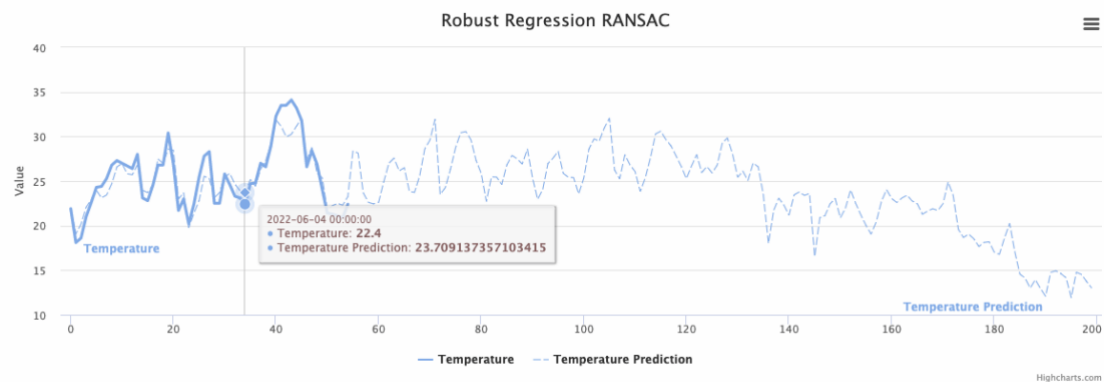


Ilustración 29: Gráfica Robust Regression RANSAC

MAE	MSE	RMSE
0.882776	1.469930	1.212407

El algoritmo ha ofrecido un buen resultado, a pesar de la forma en la que trabaja. Realiza sus iteraciones evitando usar valores atípicos del conjunto de datos. Esto hace que para un dataset con algún dato extraño, no lo tiene en cuenta y crea una línea con los valores interiores del conjunto. Este algoritmo es mejor para predecir datos lineales.

MAE: se puede comprobar que el error medio absoluto está por debajo de un grado de temperatura entre los datos predichos y los datos observados, con lo cual es un buen resultado.

MSE: para el error cuadrático medio, los resultados no son tan buenos, esto puede ser porque para algunos días no es buena la predicción. Esto es debido a que el valor real y el predicho son bastante diferentes para esos días, lo que hace que al elevar al cuadrado la diferencia entre ambos, el resultado sea excesivamente elevado y al realizar la media para obtener el MSE, aumente la puntuación.

Stochastic Gradient Descent

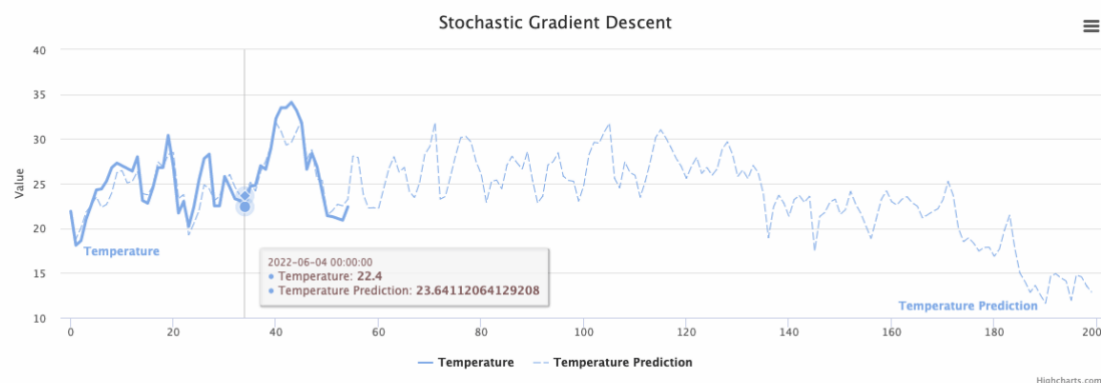


Ilustración 30: Gráfica Stochastic Gradient Descent

MAE	MSE	RMSE
0.885141	1.482844	1.217721

Ya que este algoritmo usa la base de la regresión lineal, se encuentra con el mismo problema que los demás algoritmos de este tipo, que para conseguir resultados lo más fiable posible, deben usarse con un conjunto de datos lineales.

MAE: se puede comprobar que el error medio absoluto está por debajo de un grado de temperatura entre los datos predichos y los datos observados, con lo cual es un buen resultado.

MSE: al igual que con el MAE, para el error cuadrático medio, los resultados no son buenos, esto es debido a lo que se comentaba un poco más arriba, al ser un algoritmo de basado en regresión lineal, los resultados con series temporales no suelen ser buenos.

Support Vector Regression

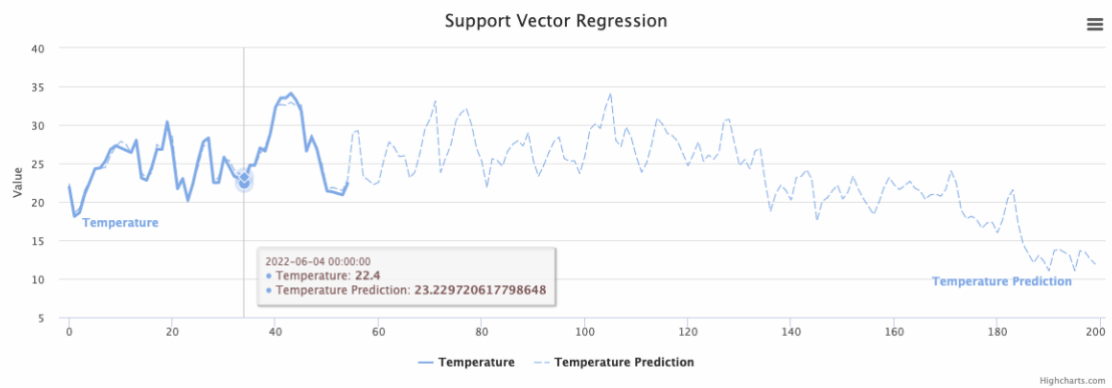


Ilustración 31: Gráfica Support Vector Regression

MAE	MSE	RMSE
0.403035	0.603295	0.776720

Este algoritmo ha sido el que mejor resultado ha conseguido en cuanto a la métrica de error absoluto medio y uno de los mejores en cuanto al error cuadrático medio. Esto coincide con la teoría y con otros casos de estudio (es usado con buenos resultados en los casos de la sección **ESTADO DEL ARTE**), ya que es ampliamente usado para series de datos temporales.

MAE: ha conseguido obtener la mejor puntuación en esta métrica, posicionándose con solo un error de 0.4 grados entre el valor predicho y el valor real. Se puede comprobar además por las gráficas, que la línea de predicción es prácticamente igual a la línea real.

MSE: aunque no ha sido el mejor para esta métrica, siendo superado por XGBoost y Random Forest, ha obtenido muy buen resultado, posicionándose con solo 0.6 grados de error.

XGBoost

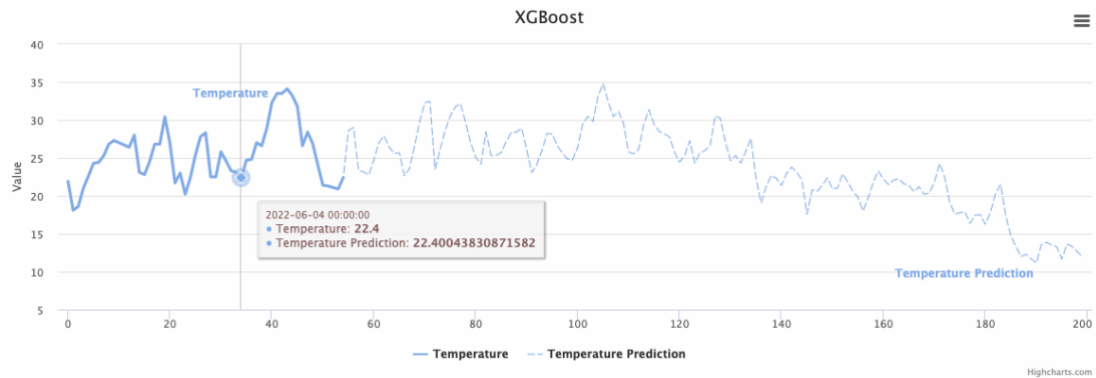


Ilustración 32: Gráfica XGBoost

MAE	MSE	RMSE
0.459350	0.514622	0.717371

Este algoritmo, como ya se explicaba en un apartado anterior, es un algoritmo basado en árboles de decisión. Para series temporales, los algoritmos de árboles de decisión son uno de las mejores opciones. Según la teoría, este algoritmo es una versión mejorada de Decision Tree Regresor y Random Forest, esto es debido a que, por ejemplo, el primero solo crea un árbol de decisión y el segundo crea varios árboles pero con una extensión limitada por el usuario. Por su parte, el funcionamiento de XGBoost consiste en ir agregando secuencialmente más árboles, con el fin de aprender del resultado de estos e ir corregir el error producido hasta que ya no se pueda corregir más dicho error (esto conocido como descenso de gradiente).

MAE: se puede comprobar con el error medio que es uno de los mejores algoritmos, ya que da un resultado muy bajo, el cual no llega a medio grado de error. Aunque según la teoría Random Forest es peor algoritmo, ha conseguido mejor resultado, aun teniendo la limitación de la extensión de los árboles, esto es debido a que el número introducido ha sido suficiente para llegar al mínimo error.

MSE: para el error cuadrático medio, los resultados son incluso mejores, ya que es el algoritmo que mejores resultados ha conseguido.

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

Tabla comparativa MAE

Tabla 3: Tabla comparativa MAE

Nombre algoritmo	MAE
Support Vector Regression	0.403035
Random Forest	0.447718
XGBoost	0.459350
Decision Tree Regresor	0.835799
RANSAC	0.872776
Linear Regression	0.880313
Ridge	0.880319
Stochastic Gradient Descent	0.885141
Elastic Net Regression	0.923130
Lasso Regression	0.932046

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. Esto, como se comentaba anteriormente, coincide con la teoría, ya que tanto el algoritmo Support Vector como los algoritmos basados en arboles de decisión (RF y XGBoost), son buenos para series temporales. Lasso Regression es el que peor resultado ha conseguido, coincidiendo igual con la teoría, ya que es un algoritmo de regresión lineal.

Tabla comparativa MSE, RMSE

Tabla 4: Tabla comparativa MSE y RMSE

Nombre algoritmo	MSE	RMSE
XGBoost	0.514622	0.717371
Support Vector Regression	0.603295	0.776720
Random Forest	0.616301	0.785048
Linear Regression	1.466668	1.211060
Ridge	1.466679	1.211065
RANSAC	1.469930	1.212407
Stochastic Gradient Descent	1.482844	1.217721
Elastic Net Regression	1.587156	1.259824
Lasso Regression	1.602886	1.266052
Decision Tree Regresor	1.750000	1.322876

El algoritmo que mejor resultado ha conseguido con esta métrica es XGBoost, seguido por Support Vector Regression y Random Forest. Decision Tree Regresor es el que peor resultado ha conseguido, aun siendo un algoritmos basado en arboles de decisión.

7 CONCLUSIONES

Para empezar, cabe destacar que el Machine Learning aplicado a dispositivos IoT es tan útil como se esperaba y se puede aplicar a numerosos ámbitos y temáticas. Gracias

a esto, se llegó a la conclusión de que podía ser interesante crear una herramienta donde aplicar tanto las virtudes del aprendizaje automático, como la utilidad de los datos que almacenan los dispositivos IoT.

Al comenzar este proyecto se establecieron una serie de objetivos, los cuales se puede decir que se han conseguido.

Se ha logrado obtener, a partir del histórico de datos, y, mediante el uso de algoritmos, unas predicciones que se ajustan a lo deseado. Se ha conseguido, también, un entorno web en el que representar, adecuadamente, y de forma intuitiva los datos. Además, se ha añadido una funcionalidad en la cual se puede cargar un archivo y representar su contenido en las gráficas. Asimismo, se ha hecho un estudio de los algoritmos y se han comparados los resultados con métricas de error, donde se ha visto que cuanto menor era el error mejores predicciones se representaban gráficamente.

A pesar de esto, siempre hay aspectos en los que se puede hacer más hincapié o ampliar; en este caso se podría haber realizado todo el backend en Python sin usar PHP como intermediario; también, haber creado un proceso mediante el uso de un cron que extraiga datos, los procese, los almacene y haga un entrenamiento nuevo diario; se podría haber añadido nuevos Dataset a predecir, como por ejemplo la contaminación del Aire; también hubiese estado acertada una gráfica en la que se hubiesen podido simular dos algoritmos con sus predicciones para poder comparar visualmente; y, por último, se podría haber añadido las métricas de error en el Frontend, junto con cada gráfica.

En general, se puede concluir que se han obtenido muy buenos resultados, ya que las predicciones son muy semejantes a los valores reales y, al fin y al cabo, era el objetivo principal que se esperaba conseguir en este proyecto.

8 REFERENCIAS BIBLIOGRÁFICAS

1. **wunderground.com.**

<https://www.wunderground.com/history/monthly/es/badajoz/LEBZ/date/>.

2. **timeanddate.com.** www.timeanddate.com/weather/spain/.

3. **Visser, Lennard, AlSkaif, Tarek y van Sark, Wildfried. Elsevier.com. 2022.**

<https://reader.elsevier.com/reader/sd/pii/S0960148121015688?token=280D13A6F4479E4B92F62354AF166484F7451982D4F5B6F488C28139DAFC90A4CD62B2552EBE9D7905CD16085AA1BFB&originRegion=eu-west-1&originCreation=20220605172019>.

4. **Martínez, Raquel, y otros. www.jucs.org. 2018.**

https://www.jucs.org/jucs_24_3/air_pollution_prediction_in/jucs_24_03_0261_0276_espana.pdf.

5. **Zheng, Huan y Wu, Yanghui. www.researchgate.net.**

https://www.researchgate.net/publication/334711620_A_XGBoost_Model_with_Weather_Similarity_Analysis_and_Feature_Engineering_for_Short-Term_Wind_Power_Forecasting.

6. **Singh, Siddharth, y otros. deliverypdf.ssrn.com.**

<https://deliverypdf.ssrn.com/delivery.php?ID=190081025071014118094007117025066121042037029042091050024118019086002125092082010075126060062111031022046118098094122106010098049015010026004066095115099094071080108023077048127111007017087072100003093115020102>.

7. **Iberdrola. Descubre los principales beneficios del Machine Learning. 2019.**

<https://www.iberdrola.com/innovacion/machine-learning-aprendizaje-automatico>.

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

9. **sitiobigdata. sitiobigdata.com.** <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>.

10. **Wikipedia.** <https://es.wikipedia.org/wiki/Metodolog%C3%ADa>.

11. **Weather Underground.** <https://www.wunderground.com/>.

12. **Vega, Juan Bosco Mendoza. medium.com.**

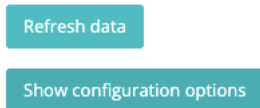
<https://medium.com/@jboscomendoza/tutorial-xgboost-en-python-53e48fc58f73>.

13. **w3schools.com.** https://www.w3schools.com/python/python_ml_linear_regression.asp.
14. **iaartificial.net.** <https://www.iaartificial.net/random-forest-bosque-aleatorio/>.
15. **jacobsoft.com.mx.** https://www.jacobsoft.com.mx/es_mx/support-vector-regression/.
16. **realpython.com.** <https://realpython.com/gradient-descent-algorithm-python/>.
17. **en.wikipedia.org.** [https://en.wikipedia.org/wiki/Lasso_\(statistics\)](https://en.wikipedia.org/wiki/Lasso_(statistics)).
18. **en.wikipedia.org.**
https://en.wikipedia.org/wiki/Ridge_regression#:~:text=Ridge%20regression%20is%20a%20method,econometrics%2C%20chemistry%2C%20and%20engineering.
19. **corporatefinanceinstitute.com.**
<https://corporatefinanceinstitute.com/resources/knowledge/other/elastic-net/>.
20. **geeksforgeeks.org.** <https://www.geeksforgeeks.org/python-decision-tree-regression-using-sklearn/>.
21. **wikipedia.org.** <https://es.wikipedia.org/wiki/RANSAC>.
22. **docs.python.org.** <https://docs.python.org/3/library/pickle.html>.
23. **highcharts.com.** <https://www.highcharts.com/>.
24. **scikit-learn.org.** <https://scikit-learn.org/stable/>.
25. **Wikipedia.** https://es.wikipedia.org/wiki/Web_scraping.

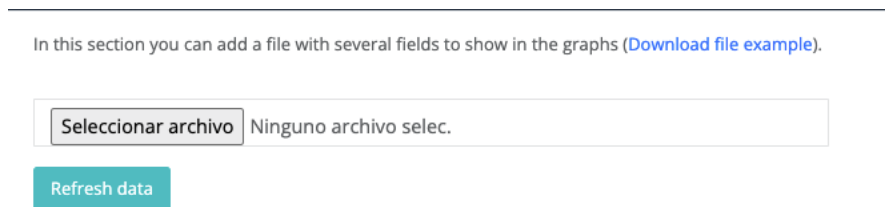
Anexo 1: Otra funcionalidad del entorno web

Se ha añadido otra funcionalidad al entorno web, en la cual se puede cargar un archivo csv con datos que se pueden representar en las gráficas.

Para ello, se ha añadido un botón debajo del botón *Refresh data*, llamado *Show configuration options*:



Cuando se pulsa, se desplegará una sección como la que se puede ver a continuación:



Si se pulsa sobre *Download file example*, se descarga un archivo de prueba para poder ver que formato hay que seguir:



Si se abre el archivo, se puede ver lo siguiente:

Algorithm	Date	Temperature	Temperature Prediction	Algorithm	Date	Temperature	Temperature Prediction	Algorithm	Date	Temperature	Temperature Prediction
Support Vector Regression	1/5/22	21,9	22,39121036	Random Forest	1/5/22	21,9	22,22099222	XGBoost	1/5/22	21,9	21,89990807
	2/5/22	18,1	18,36425306		2/5/22	18,1	18,38487771		2/5/22	18,1	18,10037613
	3/5/22	18,6	19,0621235		3/5/22	18,6	18,95363655		3/5/22	18,6	18,16291618
	4/5/22	21	21,4553631		4/5/22	21	21,50130927		4/5/22	21	21,52323532
	5/5/22	22,6	22,90372849		5/5/22	22,6	22,67601617		5/5/22	22,6	22,60012817
	6/5/22	24,3	24,52091443		6/5/22	24,3	24,54462862		6/5/22	24,3	24,30023003
	7/5/22	24,4	24,680241		7/5/22	24,4	25,27795417		7/5/22	24,4	24,06258202
	8/5/22	25,3	24,80331058		8/5/22	25,3	25,35568387		8/5/22	25,3	24,71976471
	9/5/22	26,8	26,39286251		9/5/22	26,8	26,71623139		9/5/22	26,8	26,80032539
	10/5/22	27,3	27,34033338		10/5/22	27,3	27,660371		10/5/22	27,3	27,29999733
	11/5/22	27	28,08519259		11/5/22	27	27,94743402		11/5/22	27	27,00007248
	12/5/22	26,7	27,79825644		12/5/22	26,7	27,64197684		12/5/22	26,7	26,69968605
	13/5/22	26,4	26,52169585		13/5/22	26,4	26,48259748		13/5/22	26,4	26,40006828
	14/5/22	28	27,93007968		14/5/22	28	28,61836045		14/5/22	28	27,16400337
	15/5/22	23,1	23,63726086		15/5/22	23,1	23,66660564		15/5/22	23,1	23,09973335
	16/5/22	22,8	23,03125572		16/5/22	22,8	22,93493231		16/5/22	22,8	22,800457
	17/5/22	24,5	23,79798781		17/5/22	24,5	24,24610992		17/5/22	24,5	24,50025749

Cada algoritmo representará cuatro columnas, en la primera se añadirá el nombre, en la segunda la fecha del dato, en la tercera el dato real y en la cuarta la predicción.

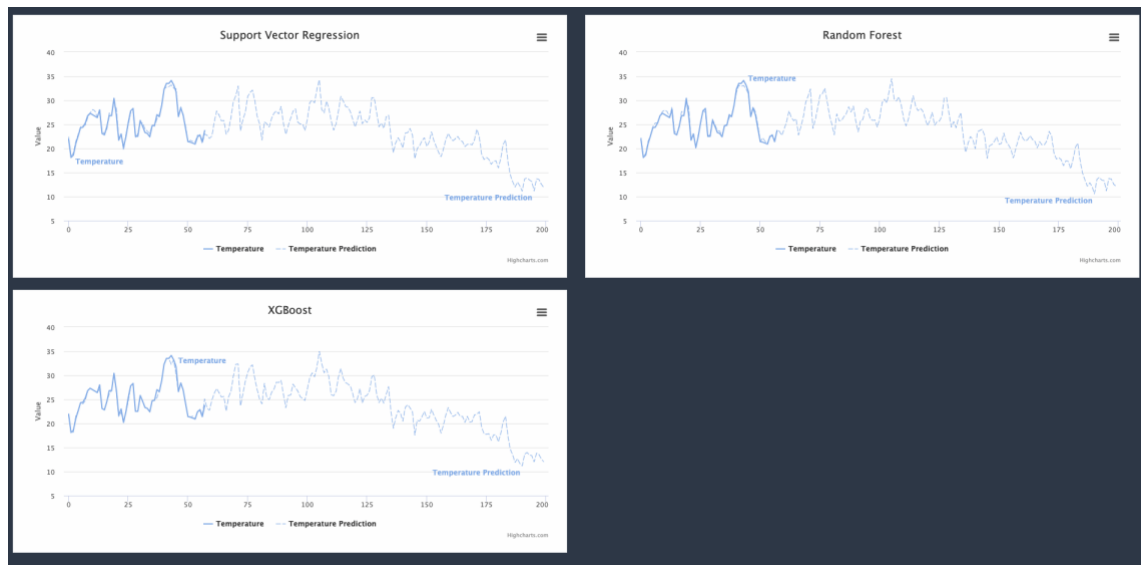
El siguiente paso es seleccionar este archivo desde el frontend, pulsando en el botón *seleccionar archivo*, hecho esto, se selecciona en el sistema el archivo que se acaba de descargar y se pulsa sobre el botón de abajo *Refresh data*.

Seleccionar archivo

algorithm.csv

Refresh data

Después, se cargarán las gráficas con los datos que contenía el csv:



Anexo 2: Instalación y ejecución del servidor Apache

Para poder ejecutar el proyecto, se va a necesitar instalar apache como servidor.

Pero, ¿qué es apache? Apache es un servidor web open source o de código abierto. Se puede usar en plataformas Unix (Linux, Mac por ejemplo). Actualmente, se estima que el 46% de los sitios web se lanzan bajo este.

Apache permite a los dueños de sitios web (ya sea en un servidor o una maquina local) mostrar contenido. Se trata de uno de los servidores web más antiguo.

Para visitar un sitio web, se debe ingresar un nombre de dominio en la barra de direcciones de su navegador, entonces el servidor actúa como repartidor virtual y envía los archivos solicitados.

Sabiendo que es Apache, se puede proseguir con los pasos para realizar la instalación.

En este 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.

Lo primero se instala apache, el comando de instalación que se debe usar es:

```
brew install httpd
```

Una vez instalado, se debe indicar en el archivo *httpd.conf* cuál va a ser la ruta que debe abrir. Se abre el archivo para editarlo con el siguiente comando:

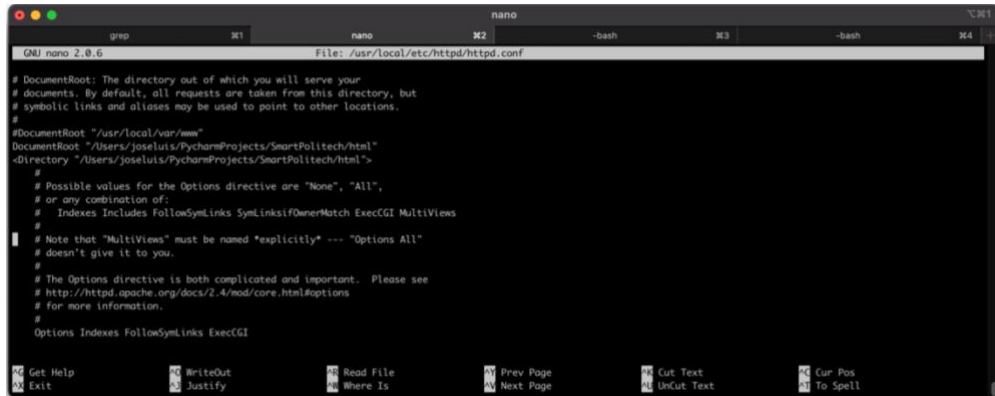
```
nano /usr/local/etc/httpd/httpd.conf
```

Se baja hasta la línea 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 dónde se encuentra la carpeta con el main principal. Por defecto aparecerá `#DocumentRoot "/usr/local/var/www"`, con lo cual, se debe que cambiar `"/usr/local/var/www"` por la ruta donde esté localizado el proyecto. En este caso será `"/Users/joseluis/Documents/MPAIOT/html"`, quedando la línea así:

`#DocumentRoot "/Users/joseluis/Documents/MPAIOT /html"`.

Una vez hecho esto, debajo de esta línea hay otra que también se debe modificar:

`<Directory "/usr/local/var/www">` y se debe hacer lo mismo, se sustituye `"/usr/local/var/www"` por la ruta que se use, en este caso queda así:

`<Directory "/Users/joseluis/PycharmProjects/SmartPolitech/html">`

Hecho esto, se baja un poco en el fichero hasta donde se encuentra el fragmento de código siguiente:

`<IfModule dir_module>`

`DirectoryIndex index.html`

`</IfModule>`

Si el main que se usa en el proyecto se llama `index.html`, se dejará como está, si por el contrario se llama de otra forma, se debe cambiar. En este caso se llama `main.html`, por lo que el 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 modificado el archivo de configuración para poder ejecutar el 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.

```
MacBook-Pro-de-Jose-2:bin joseluis$ brew services list
Name  Status  User      Plist
httpd  started joseluis  /Users/joseluis/Library/LaunchAgents/homebrew.mxcl.httpd.plist
php    started joseluis  /Users/joseluis/Library/LaunchAgents/homebrew.mxcl.php.plist
```

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

Anexo 3: Repositorio y estructura del proyecto

El proyecto se puede encontrar en Github en la url:

<https://github.com/jperezlf/MPAIOT>

A continuación, se explica cómo se estructura el proyecto:

La carpeta *Algorithm* contiene toda la lógica de los algoritmos, el script de entrenamiento (*model_trainer.py*), el script de carga del modelo (*model_trainer.py*), y el archivo con el entrenamiento (*train.dat*).

La carpeta *Data_extraction* contiene el script de extracción de datos (*extractData.py*) con el que se recoge la información de la web para crear el modelo de datos.

La carpeta *Db* contiene un script con las funciones que sirven de ayuda para conectarse a la base de datos, hacer inserciones, borrado o actualizaciones.

La carpeta *Doc* contiene la memoria del proyecto.

La carpeta *Files* contiene el Dataset que se inserta en la base de datos y que se usa para los entrenamientos.

La carpeta *HTML* contiene toda la lógica del frontend

La carpeta *Process* contiene el proceso para insertar el Dataset en la base de datos.

La carpeta *Training* contiene los script para lanzar todos los entrenamientos a la vez.

