Sección de Matemáticas
Universidad de La Laguna

Roberto Calvo Cubas

Machine Learning aplicado a un problema de Regresión

Machine Learning in a Regression Problem

Trabajo Fin de Grado
Grado en Matemáticas
La Laguna, Mayo de 2024

Dirigido por
Belén Melián Batista
J. Marcos Moreno Vega

Belén Melián Batista
Departamento de Ingeniería Informática y de Sistemas
Universidad de La Laguna
38200 La Laguna, Tenerife

J. Marcos Moreno Vega
Departamento de Ingeniería Informática y de Sistemas
Universidad de La Laguna
38200 La Laguna, Tenerife

Agradecimientos

Gracias a Dack, a Betti, a Aidu, a Shiva y Uli, a la L, al Pisito, y, por último pero no menos importante, gracias a todas las personas de la Universidad que también formaron parte del proceso.

Roberto Calvo Cubas
La Laguna, 13 de mayo de 2024

Resumen · Abstract

Resumen

En el mundo actual se genera continuamente una inmensa cantidad de datos. Estos datos se pueden estructurar de tal forma que se pueda extraer información valiosa de ellos e incluso faciliten o permitan la resolución de problemas reales.
En este trabajo se pretende introducir al lector en el campo del Aprendizaje Automático o Machine Learning. Para ello, se propone estimar una solución para un problema real dado en un determinado puerto de España.
En primer lugar, se detallarán los aspectos teóricos y fundamentos de Machine Learning, así como de las tecnicas propias de Machine Learning aplicadas. Entonces, se presentará el problema y se aplicarán las técnicas de Machine Learning consideradas para el problema dado. Finalmente, se expondrán los resultados obtenidos junto a conclusiones prometedoras al respecto.

Palabras clave: *Ciencia de Datos – Machine Learning – Regresión – kNN – Random Forest – XGBoost – Multidimensional Scaling – Local Outlier Factor – Isolation Forest*

Abstract

Nowadays, an inmense amount of data is generated. This data can be structured in such a way that it can be extracted valuable information and make easier real-world problem solving.
In this work, the aim is to introduce the reader the field of Machine Learning. To this end, it is proposed to estimate a solution for a real-world problem in a given port of Spain.
First of all, the theoretical aspects and fundamentals of Machine Learning will be detailed, as well as the Machine Learning's techniques applied. Then, background will be described and abovementioned techniques will be applied. Finally, the results will be presented along to promising conclusions.

Keywords: *Data Science – Machine Learning – Regression – kNN – Random Forest – XGBoost - Multidimensional Scaling – Local Outlier Factor – Isolation Forest.*

Contenido

Introducción

La Ciencia de Datos es un campo interdisciplinario que combina Matemáticas y Ciencias de la Computación para descubrir información útil oculta en datos de cualquier tipo. Algunos de esos tipos de datos son: números, textos, imágenes, audios o vídeos. El eje sobre el que gira la Ciencia de Datos, y este trabajo, es el Machine Learning (ML) o Aprendizaje Automático (AA), concepto que surge de la combinación antes mencionada.

Tanto la Ciencia de Datos como el Machine Learning son campos relativamente recientes y que todavía se encuentran en expansión. Sin embargo, ambos se consideran herramientas fundamentales en el desarrollo de campos de especial relevancia en la era tecnológica actual, como por ejemplo la ciberseguridad y el análisis de datos masivos.

El objetivo que se propone este trabajo consiste en exponer, tanto de forma teórica como práctica, la construcción de una solución para un problema real dado en el Puerto de Valencia. Para ello, se procederá de la siguiente manera.

En el capítulo 1, se realizará una descripción sobre los fundamentos de Machine Learning y las técnicas que se utilizarán en la resolución del problema, distinguiendo entre Machine Learning supervisado y no supervisado. También se hará mención a la optimización de hiperparámetros.

En el capítulo 2, se comenzará presentando los aspectos del problema práctico. A continuación, se describen la población y muestra, variables y objetivo del estudio. Luego, se incluye una estadística descriptiva de los datos y se describe la aplicación de los modelos de Machine Learning supervisados aplicados, junto con los resultados obtenidos. Al contexto generado por todo lo relacionado a este capítulo se le denominará **Escenario 1**.

En el capítulo 3, en continuación al desarrollo en el Escenario 1, se aplicarán técnicas de Machine Learning no supervisado de donde se obtienen resultados prometedores. Se le denominará **Escenario 2**.

Finalmente, se aportan conclusiones respecto a los resultados obtenidos en ambos escenarios y se proponen una serie de acciones de mejora, acompañadas seguidamente del apéndice y la bibliografía.

1

Preliminares

En este capítulo se realizará una introducción a los fundamentos del Machine Learning. A continuación, se llevará a cabo una descripción de los algoritmos utilizados en el estudio. Finalmente, se presentará un aspecto de especial importancia dentro del Machine Learning, la optimización de hiperparámetros.

1.1. Fundamentos del Machine Learning

A un nivel muy básico, se podría decir que el Aprendizaje Automático es la rama de la Inteligencia Artificial (IA) que tiene como objetivo desarrollar técnicas que permitan a las computadoras aprender. Gran parte de la estructura de esta sección está basada en [1].

Se presentará el ML como una combinación de tres componentes:

- Los datos
- El espacio de hipótesis
- La función de pérdida

Nota 1.1: Cabe mencionar que la definición y estructura de las tres componentes anteriores es una elección de diseño y corresponde al analista asumir el compromiso entre las complejidades computacionales y las propiedades estadísticas de los métodos de ML resultantes.

1.1.1. Los datos

La mayoría de los métodos de ML construyen estimaciones/predicciones sobre un conjunto de datos de referencia (entrenamiento), y que se vuelven más precisas a medida que aumenta el número de datos utilizados y se itera sobre el proceso de optimización que conlleva la construcción de una estimación.

Cada dato se caracteriza por su vector de características $x \in X \subseteq \mathbb{R}^n$, con $x = (x_1, ..., x_n)$, siendo n el número de atributos del objeto y $X \equiv$ el espacio de características.

La mayoría de los métodos de ML trabajan con valores numéricos reales, por lo que hay procedimientos estandarizados para convertir características no numéricas en características numéricas.

Un concepto de especial importancia es el de *etiqueta* de un dato [2]. La etiqueta de un dato consiste en el valor real del atributo que se quiere estimar sobre el mismo dato. La etiqueta de un dato puede ser conocida o no. Además, el conjunto de todos los posibles valores de etiqueta de los datos, es decir, de todos los posibles resultados, se denomina espacio de etiquetas Y.

Definición 1.1.1.- Se considera que un dato está etiquetado si, además de sus características, $x \in X$, se conoce el valor de su etiqueta, $y \in Y$.

Esta definición da lugar a la siguiente clasificación dentro del *Machine Learning o Aprendizaje Automático*:

- **Aprendizaje supervisado**. El cual está basado en datos etiquetados.
- **Aprendizaje no supervisado**. El cual está basado en datos no etiquetados.
- **Aprendizaje semi-supervisado**. Mezcla de las dos clasificaciones anteriores, combinando datos etiquetados y no etiquetados.
- **Aprendizaje por refuerzo**. Donde el algoritmo aprende reforzando aquellas acciones que reciben una respuesta positiva en el sistema.

1.1.2. El modelo o espacio de hipótesis

El principio de los métodos de ML consiste en aprender una hipótesis

$$h: \ X \to Y, \quad \text{tal que} \quad y \approx h(x), \quad \forall x \in X \tag{1.1}$$

(el ideal sería $y = h(x), \forall x \in X$)

El conjunto de todas las posibles aplicaciones desde el espacio de características X hasta el espacio de etiquetas Y es donde, en principio, se debería buscar la mejor hipótesis. Pero, en general, este espacio es demasiado grande para que se pueda buscar en él con métodos prácticos.

En comparación, los métodos prácticos de ML solo pueden buscar y evaluar un subconjunto (minúsculo) de todas las hipótesis posibles, H, que se denomina espacio de hipótesis o modelo subyacente a un método de ML.

1.1.3. La función de pérdida

Cuando se aplica un modelo de *Machine Learning*, es natural preguntarse si el modelo está funcionando correctamente o, si se está equivocando, cuánto se está equivocando. Es por ello que se vuelve fundamental poder medir el error.

Definición 1.1.2.- Una función de pérdida es una aplicación

$$L: \ X \times Y \times H \to \mathbb{R}^+$$

$$(x, y, h) \rightarrow L((x, y), h) \tag{1.2}$$

El valor de pérdida $L((x, y), h)$ cuantifica la discrepancia entre la etiqueta verdadera, y, y la etiqueta predicha, $h(x)$. Un valor $L((x, y), h)$ cercano a 0 indica una discrepancia baja, por lo que se podría considerar como un indicador sobre el funcionamiento del algoritmo.

Existen numerosas funciones de pérdida y su uso dependerá de la naturaleza de la variable que se quiera estimar. En este caso, se utilizarán dos opciones ampliamente utilizadas para el tipo de problema que se pretende resolver (regresión). Se establecen a continuación:

- **Pérdida de Error Cuadrático**:

$$L((x, y), h(x)) = (y - h(x))^2 \tag{1.3}$$

- **Pérdida de Error Absoluto**:

$$L((x, y), h(x)) = |y - h(x)| \tag{1.4}$$

El modelo probabilístico más básico y utilizado interpreta los datos (x^i, y^i) como realizaciones de variables aleatorias independientes e idénticamente distribuidas (i.i.d.) que siguen una distribución de probabilidad común $p(x, y)$.

La ML más práctica se centra en los métodos que no requieren conocer $p(x, y)$, y el método más utilizado consiste en aproximar la pérdida esperada mediante una media empírica (muestral) sobre un conjunto finito de datos etiquetados (datos de test).

Definición 1.1.3.- Se define el Riesgo Empírico de una hipótesis $h \in H$ para un conjunto de datos $D = \{(x^1, y^1), ..., (x^m, y^m)\}$ como:

$$\widehat{L}(h|D) = \frac{1}{m} \sum_{i=1}^{m} L((x^i, y^i), h) \tag{1.5}$$

Obsérvese que, en general, el riesgo empírico depende tanto de h como de D. Si los datos utilizados para calcular el riesgo empírico se pueden modelizar como variables aleatorias i.i.d. cuya distribución común es $p(x, y)$, la Ley de los grandes números mantiene que, para un tamaño de muestra m suficientemente grande se tiene que:

$$E[L((x, y), h(x))] \approx \frac{1}{m} \sum_{i=1}^{m} L((x^i, y^i), h) \tag{1.6}$$

En este trabajo se consideran dos formas de medir el riesgo empírico, basadas en las funciones de pérdida 1.3 y 1.4, respectivamente:

- **RMSE** (Error Cuadrático Medio o *Root Mean Squared Error*):

$$\text{RMSE} = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(y^i - h(x^i)\right)^2} \tag{1.7}$$

  siendo m el número total de observaciones.
- **MAE** (Error Absoluto Medio o *Mean Absolute Error*):

$$\text{MAE} = \frac{1}{m}\sum_{i=1}^{m}\left|y^i - h(x^i)\right| \tag{1.8}$$

  siendo m el número total de observaciones.

1.2. Underfitting y Overfitting

El objetivo de un modelo de Machine Learning consiste en aprender de datos que ya se tienen para poder generalizar a futuros nuevos datos de una forma apropiada [4]. Uno de los principales problemas con los que se ha de lidiar es conseguir un equilibrio entre underfitting y overfitting, o lo que es lo mismo, entre Bias y varianza.

Cabe mencionar que una práctica generalizada en Machine Learning consiste en dividir los datos de la muestra en dos conjuntos: el de entrenamiento y el de test. Con el conjunto de entrenamiento se construye el modelo y con el conjunto de test se validan los resultados, es decir, se comprueba el comportamiento del modelo con datos nuevos.

En el caso de tener datos etiquetados, el riesgo empírico se puede medir tomando los datos de entrenamiento o los datos de test. Cabe esperar que el modelo se ajuste mejor a los datos de entrenamiento ya que son datos conocidos. Sin embargo, puede ocurrir todo lo contrario. O también puede darse que un modelo sea muy preciso en el entrenamiento, pero en el test se obtenga un error significativamente grande. Este tipo de problemas se clasifican en dos tipos, *underfitting* y *overfitting*, y están ligados a la siguientes definiciones.

Definición 1.2.1.- Se define el **Bias** o sesgo estadístico como la incapacidad de un modelo de ML para calcular la verdadera relación entre los datos.

Definición 1.2.2.- Se define la **varianza** de un modelo de ML como la diferencia de precisión en la estimación de diferentes conjuntos de datos.

Se puede interpretar el Bias como el error cometido. Entonces, se tiene que si se mide un Bias alto para los datos de entrenamiento, el modelo está siendo incapaz de capturar la relación entre los datos. En este caso, se tiene un problema de **underfitting** y suele estar dado por falta de datos o excesiva simplicidad en los modelos.

Por otro lado, un modelo puede ser extremadamente preciso con los datos de entrenamiento, pero luego con los datos de test cometer errores inaceptables. Esto ocurre cuando el modelo se aprende de memoria los datos de entrenamiento, con un bajo Bias, pero no es capaz de capturar la naturaleza de los datos y, por tanto, fracasa en la estimación de nuevos datos. A este problema se le conoce como **overfitting** y viene acompañado de una alta varianza del modelo.

Para obtener resultados prometedores es necesario encontrar un equilibrio entre underfitting y overfitting; es decir, construir modelos que tengan el menor Bias y varianza posible.

1.3. Algoritmos de Machine Learning supervisado

El Machine Learning supervisado abarca todos aquellos modelos cuyo objetivo es aprender una función que relacione las variables predictoras (que se encuentran en el espacio de características X) con la variable de salida Y, y para los cuales son necesarios datos etiquetados. Cabe mencionar que, dentro del ML supervisado, surge la siguiente clasificación:

- Modelos de **clasificación**. Cuando la variable de salida, Y, es discreta.
- Modelos de **regresión**. Cuando la variable de salida, Y, es continua.

A continuación, se describirán los siguientes algoritmos clásicos de Machine Learning supervisado aplicados en este trabajo:

- k Nearest Neighbour (kNN).
- Árbol de decisión.
- Random Forest.
- XGBoost.

1.3.1. kNN (k Nearest Neighbour)

Sea k un entero positivo prefijado, el algoritmo k Nearest Neighbour (kNN) es un algoritmo de clasificación o regresión, supervisado y no paramétrico [5]. En particular, kNN se basa en la idea de que los nuevos ejemplos serán clasificados con la misma clase que tengan la mayor cantidad de vecinos más parecidos a ellos del conjunto de entrenamiento. En el caso de tratarse de un problema de regresión, el valor predicho pasa a ser la media aritmética o ponderada de las etiquetas (valores de y) de los k vecinos más cercanos [6].

kNN incluye una condición que ha de cumplirse entre los datos que se tengan, y es que ha de existir una forma de medir el grado de similaridad entre dos cualesquiera de ellos (en el apéndice A, se incluyen las definiciones de distancia y similaridad obtenidas en [7]).

Por lo tanto, para poder aplicar el algoritmo kNN es necesario poder definir una distancia o similaridad en X, el espacio de características. Una vez definida

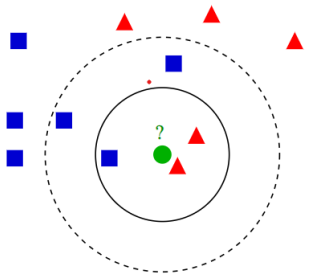


Figura 1.1. Ejemplo algoritmo clasificador kNN.

la forma de medir la distancia, a la hora de clasificar un nuevo dato, x, los pasos que realiza el algoritmo se podrían resumir en:

- Se seleccionan las k observaciones más cercanas o similares a x dentro del conjunto de datos de entrenamiento, $(x_1, ..., x_k)$. A cada una de las k observaciones se le proporciona un peso w_i $(i = 1, ..., k)$. Esto permite elegir que todas las observaciones ponderen igual o que tengan mayor peso aquellas que se encuentren más cerca de x. En este último caso, un ejemplo puede ser

$$w_i = \frac{1}{d(x, x_i)^2} \quad , \quad i = 1, ..., k$$

  siendo $x_i \in X$ y $d(x, x_i)$ la distancia o similaridad entre x y x_i, con $i = 1, ..., k$.
- Se distinguirá entre regresión y clasificación:
  1. Para un problema de **clasificación**, la clase asignada a x será aquella que verifique que la suma de los pesos de sus representantes sea máxima, es decir:

$$\arg\max_{v \in V} \sum_{i=1}^{k} w_i$$

     con $v \in V$ las posibles clases de clasificación.
  2. Para un problema de **regresión** [6], el valor predicho se calculará como la media ponderada de los valores de etiqueta de los k vecinos:

$$\widehat{y} = \frac{1}{k} \sum_{i=1}^{k} w_i y_i$$

     con (x_i, y_i) $\forall i = 1, ..., k$, los datos de entrenamiento etiquetados.

1.3.2. Árbol de decisión

Los árboles de decisión son modelos predictivos y no paramétricos formados por nodos cuyo objetivo es repartir las observaciones en función de sus atributos y predecir así el valor de la variable respuesta. Se consideran de especial

relevancia al ser los elementos fundamentales de un conjunto de algoritmos de *Machine Learning*, tales como *Random Forest*, *XGBoost* e *Isolation Forest*. A continuación, se tratará de resumir la construcción de un árbol de regresión, basada en información obtenida en [8], [9] y [10].

La idea de un árbol de decisión consiste en agrupar las observaciones que tengan atributos similares. Para ello, se parte de un nodo primario en el que se encuentran todas las observaciones del conjunto de datos. Entonces, se realiza una división de los datos en función de un determinado valor en un atributo de los mismos. Al grupo previo a la división se le conoce como nodo principal y los grupos que se obtienen tras la división se les conoce como nodos secundarios. El proceso de dividir las observaciones se repite hasta que se alcanza una regla de parada y se tiene, finalmente, el árbol construido.

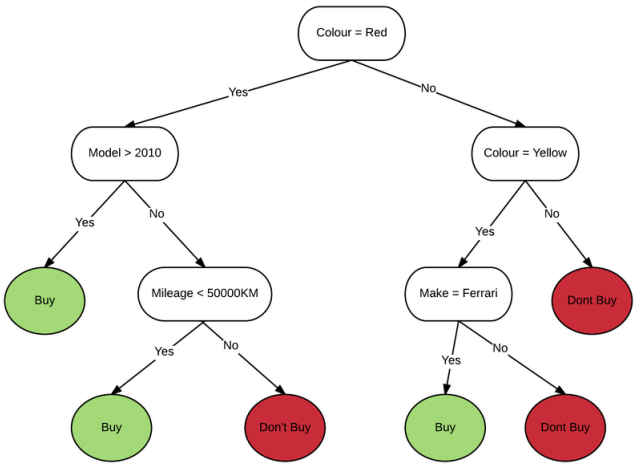


Figura 1.2. Ejemplo de un árbol de decisión.

Por simplicidad y para reducir el espacio de búsqueda combinatoria, la mayoría de las bibliotecas implementan árboles de decisión binarios, donde cada nodo principal es dividido en dos nodos secundarios, D_L y D_R.

Sean $X = (X_1, ..., X_n)$ el espacio de características o variables predictoras ($n \in \mathbb{N}$), e Y el espacio de etiquetas o variable respuesta. La función objetivo a maximizar en cada división es la ganancia de información:

$$IG(D_p, X_i) = I(D_p) - I(D_L) - I(D_R), \tag{1.9}$$

con:

D_p : el nodo principal.

X_i : la variable predictora para realizar la división, con $i = 1, ..., n$.

I : la medida de impureza.

Como se puede observar, cuanto menor es la impureza de los nodos secundarios, mayor es la ganancia de información. En un problema de regresión, la forma más frecuente (y la utilizada en este trabajo) para medir la impureza de un nodo D_j es la suma residual de cuadrados (RSS) en tal nodo:

$$I(D_j) = \sum_{i \in D_j} \left(y_i - \bar{y}_{D_j}\right)^2, \tag{1.10}$$

donde $\bar{y}_{D_j}$ es el promedio de la variable de etiqueta Y dentro del nodo D_j.

El método utilizado para construir el árbol de decisión es *recursive binary splitting* (división binaria recursiva).

El objetivo del método consiste en encontrar, en cada iteración, el predictor X_j y el punto de corte s tal que, si se distribuyen las observaciones en las regiones $D_L = \{X|X_j < s\}$ y $D_R = \{X|X_j \geq s\}$, se consiga la mayor ganancia de información posible. El algoritmo seguido es:

1. Se identifican todos los posibles puntos de corte s para cada una de las variables predictoras $(X_1, ..., X_n)$. En el caso de ser cualitativas, los posibles puntos de corte pueden ser cada uno de sus niveles. Para variables continuas, se ordenan de menor a mayor sus valores y el punto intermedio entre cada par de valores se emplea como punto de corte.
2. Se calcula la ganancia de información total que se consigue con cada posible división identificada en el paso 1; es decir,

$$I(D_L) + I(D_R) \tag{1.11}$$

3. Se selecciona la variable X_j y el punto de corte s que minimice la suma anterior; es decir, la impureza ambos nodos. Si existen dos o más divisiones que cumplan dicha condición, la elección entre ellas es aleatoria.
4. Se repiten de forma iterativa los 3 primeros pasos para cada una de las regiones que se han creado en la iteración anterior hasta que se alcanza alguna regla de parada, normalmente la profundidad máxima prefijada para el árbol o la ganancia de información mínima requerida.
5. Se distinguirá entre regresión y clasificación:
   - Para un problema de **clasificación**, una vez construido el árbol, se distribuirá la observación x que se quiere estimar a través del árbol hasta alcanzar un nodo terminal. Entonces, la clase asignada a x será la clase que más se repita dentro de dicho nodo, es decir, la moda de la variable de salida Y en el nodo terminal
   - Para un problema de **regresión**, el proceso que asocia la observación x a su correspondiente nodo terminal es el mismo. La diferencia es que la estimación es igual al promedio de la variable de salida en el nodo terminal, es decir:

$$\widehat{y} = h(x) = \frac{1}{|D_j|} \sum_{i \in D_j} y_i \tag{1.12}$$

siendo D_j el nodo terminal j con $j = 1, ..., T$, y T el número de nodos terminales del árbol.

1.3.3. Random Forest

Random Forest es un modelo predictivo formado por un conjunto de árboles de decisión (1.3.2). De igual forma que los árboles de decisión, Random Forest es un modelo no paramétrico y se puede aplicar tanto a problemas de clasificación como de regresión.

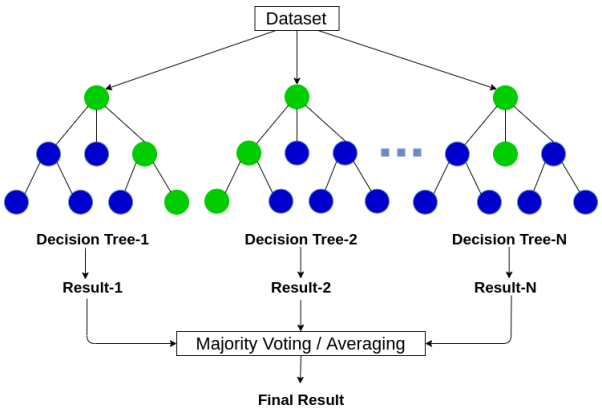


Figura 1.3. Algoritmo Random Forest.

Random Forest es una modificación de *Bagging*, que también es un método de aprendizaje estadístico [11] cuyo procedimiento tiene como propósito la reducción de la varianza del modelo.

Sean $X = (X_1, ..., X_p) \subseteq \mathbb{R}^p$ el espacio de características o variables predictoras ($p \in \mathbb{N}$), e $Y \subseteq \mathbb{R}$ el espacio de etiquetas o variable respuesta. El algoritmo que sigue un modelo Random Forest, representado en la Figura 1.3, se podría resumir en los siguientes pasos:

1. Obtener múltiples muestras de la población. Generalmente, no se tiene acceso a múltiples muestras. Por ello, se recurre a la técnica estadística no paramétrica *Bootstrapping* [12], basada en el muestreo repetido.
2. Ajustar un árbol distinto con cada una de ellas, donde, en cada división del árbol, se seleccionan aleatoriamente m ($m \leq p$) variables predictoras para ser las únicas a tener en cuenta en dicha división. Esto se hace con el fin de evitar problemas de correlación entre los árboles y para prevenir el overfitting.
3. Calcular el promedio (regresión) o la moda (clasificación) de las predicciones resultantes de cada árbol ajustado.

1.3.4. XGBoost

Al igual que Random Forest, XGBoost es un modelo predictivo basado en árboles de decisión (1.3.2), es un modelo no paramétrico y se puede aplicar tanto en regresión como en clasificación. La diferencia es que los árboles son entrenados de forma secuencial, de forma que cada nuevo árbol trata de mejorar los errores de los árboles anteriores. Es por esto, que las predicciones de cada árbol llevan asociado un peso que pondera su aportación al resultado final.

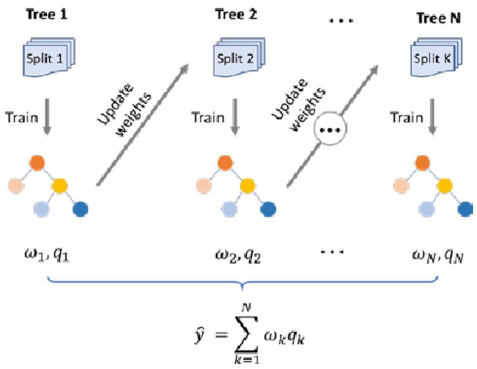


Figura 1.4. Algoritmo XGBoost.

Sea el dataset $D = \{(X_i, y_i)\}$, con $X_i \in \mathbb{R}^m$, $y_i \in \mathbb{R}$, $i = 1, ..., n$. Siendo m el número de variables predictoras, n el número de observaciones de la muestra y N el número de árboles que se van a entrenar. La predicción $\widehat{y}$ realizada por XGBoost [14] para cada árbol es :

$$\widehat{y}_i = \sum_{k=1}^{N} f_k(X_i) \quad , \quad f_k \in F \tag{1.13}$$

con

$$F = \left\{ f(X) = \omega_{q(X)} \right\} \quad , \quad (q : \mathbb{R}^m \to T, \omega \in \mathbb{R}^T) \tag{1.14}$$

donde

F : el espacio de los árboles de regresión.

$q(X)$: regla de decisión que mapea la observación X a su nodo terminal.

T : número de nodos terminales.

ω : el peso de cada nodo terminal.

f_k : k-ésimo árbol.

El objetivo es, en cada iteración t, encontrar el árbol f_t que minimice la siguiente función objetivo:

$$L^{(t)} = \sum_{i=1}^{n} \left\{ l\left(y_i, \widehat{y}_i^{(t-1)}\right) + f_t(X_i) \right\} + \Omega(f_t) \tag{1.15}$$

con

$$\Omega(f_t) = \gamma T + \frac{1}{2}\lambda\omega^2 \tag{1.16}$$

donde:

l : la función de pérdida, que ha de ser diferenciable y convexa.
Ω : término de regularización que restringe la complejidad.
γ : ganancia de información mínima requerida.
λ : coeficiente del término de regularización.

La estrategia seguida por XGBoost [15] para aproximar el óptimo de la ecuación 1.15 consiste en utilizar los gradientes de primer y segundo orden de la función de pérdida, tal que

$$L^{(t)} \approx \sum_{i=1}^{n} \left\{ l\left(y_i, \widehat{y}_i^{(t-1)}\right) + g_i f_t(X_i) + \frac{1}{2} h_i f_t^2(X_i) \right\} + \Omega(f_t) \tag{1.17}$$

con $g_i = \delta_{\widehat{y}_i^{(t-1)}} l\left(y_i, \widehat{y}_i^{(t-1)}\right)$ y $h_i = \delta^2_{\widehat{y}_i^{(t-1)}} l\left(y_i, \widehat{y}_i^{(t-1)}\right)$ los gradientes de primer y segundo orden de la función de pérdida. Eliminando los términos constantes, queda la expresión simplificada

$$\widehat{L}^{(t)} = \sum_{i=1}^{n} \left\{ g_i f_t(X_i) + \frac{1}{2} h_i f_t^2(X_i) \right\} + \Omega(f_t) \tag{1.18}$$

Definiendo $I_j = \{i \quad / \quad q(X_i) = j\}$ como el conjunto de observaciones del nodo j, la ecuación 1.18 se puede reescribir expandiendo $\Omega(f_t)$ como

$$\begin{aligned} \widehat{L}^{(t)} &= \sum_{i=1}^{n} \left\{ g_i f_t(X_i) + \frac{1}{2} h_i f_t^2(X_i) \right\} + \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} \omega_j^2 \\ &= \sum_{j=1}^{T} \left\{ (\sum_{i \in I_j} g_i)\omega_j + \frac{1}{2}(\sum_{i \in I_j} h_i + \lambda)\omega_j^2 \right\} + \gamma T \end{aligned} \tag{1.19}$$

Para cierta estructura fija $q(X)$, el peso óptimo ω_j^* para cada nodo j se puede computar de la siguiente forma

$$\omega_j^* = -\frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda} \tag{1.20}$$

y calcular el valor óptimo de

$$\widehat{L}^{(t)} = -\frac{1}{2}\sum_{j=1}^{T}\frac{(\sum_{i\in I_j} g_i)^2}{\sum_{i\in I_j} h_i + \lambda} + \gamma T \tag{1.21}$$

La ecuación 1.21 se puede entender como una función de puntuación que mide la calidad de un árbol con estructura $q(X)$. Entonces, en la iteración t, se escoge el árbol que maximice la puntuación 1.21. En la mayoría de los casos, es imposible tener en cuenta todas las estructuras $q(X)$ posibles, por lo que que se aplica un algoritmo *agresivo* que comienza con una rama e, iterativamente, va añadiendo divisiones.

El proceso de optimización continúa hasta que se alcanza el número máximo prefijado de iteraciones (árboles), N, regulado por γ, ganancia de información mínima requerida. Entonces, una vez seleccionados los árboles con sus correspondientes pesos para los nodos terminales, la predicción para un problema de regresión es la media ponderada de los promedios

$$\widehat{y}_i = \frac{1}{N}\sum_{k=1}^{N} w_{q_k(X_i)}\bar{y}_{q_k(X_i)} \tag{1.22}$$

siendo $q_k(X_i)$ el nodo terminal correspondiente a la observación X_i en el árbol k.

1.4. Algoritmos de Machine Learning no supervisado

Normalmente, la mayor parte de las definiciones, resultados teóricos y algoritmos clásicos más importantes de Machine Learning, se clasifican como algoritmos supervisados y, sobre todo en el pasado, muchos de los algoritmos no supervisados se reservaban para tareas de preprocesamiento.

Sin embargo, recientemente, han ido surgiendo nuevos algoritmos no supervisados relacionados con lo que se conoce como Aprendizaje de la Representación y donde líneas de trabajo como el Deep Learning están tomando gran importancia [17]. En el caso de este trabajo, se utilizarán modelos de ML no supervisado para la detección de *outliers*.

1.4.1. Conceptos previos

Definición 1.4.1.- Dentro de un conjunto de datos, un **outlier** es aquel dato que se comporta de una forma atípica respecto al resto de los datos.

El objetivo, es someter los outliers a estudio para conocer la razón de su peculiaridad. Muchas reglas han propuesto el eliminar estos puntos y continuar el estudio. Sin embargo, es altamente recomendable estudiar, en primer lugar, el origen y la estructura de los outliers para, posteriormente, tomar la decisión

sobre qué hacer con estos puntos. Esto se debe a que en algunos casos (por ejemplo, en la ciberseguridad), son precisamente las observaciones anómalas las que más información aportan al modelo y es fundamental tenerlas en cuenta.

Existen dos principales problemas de clasificación que surgen dentro de la detección de outliers:

- Clasificar una observación normal como outlier, lo que se conoce como **swamping**.
- Clasificar un outlier como observación normal, lo que se conoce como **masking**.

En el caso unidimensional, una de las técnicas más utilizadas para la detección de observaciones atípicas es el diagrama de cajas o boxplot. En la práctica, sin embargo, la mayoría de los problemas son multidimensionales y las técnicas llevadas a cabo se enfrentan a una mayor complejidad computacional a medida que aumenta la dimensión.

Según [19], los métodos utilizados para la detección de outliers se pueden clasificar en tres categorías:

- Métodos basados en los vecinos más cercanos (nearest neighbor).
- Métodos basados en análisis Cluster.
- Métodos basados en proyecciones.

En este trabajo, se aplicaron los siguientes métodos:

- Local Outlier Factor (LOF).
- Multidimensional Scaling (MDS).
- Isolation Forest.

1.4.2. Local Outlier Factor (LOF)

Se puede considerar el Local Outlier Factor (LOF) como el grado de anomalía de un objeto. LOF fue introducido en [20] y supuso una nueva clasificación: outliers locales o globales. En pocas palabras, dicha clasificación considera la diferencia entre comparar una observación con el conjunto de datos completo o respecto a las observaciones más cercanas.

Previamente a la definición formal de LOF, se vuelven necesarias las siguientes definiciones.

Definición 1.4.2.1.- (k-distancia de un objeto p)

Sea el dataset $D = \{x_i \in \mathbb{R}^m, i = 1, ..., n\}$ $(n, m \in \mathbb{N})$. Para un entero positivo k, se define la k-distancia de un objeto $p \in D$ (denotada k-distancia(p)) como la distancia $d(p, o)$ entre p y un objeto $o \in D$ tal que:

1. como mínimo para k objetos $o' \in D - \{p\}$ se tiene que $d(p, o') \leq d(p, o)$
2. como máximo para $k - 1$ objetos $o' \in D - \{p\}$ se tiene que $d(p, o') < d(p, o)$

Definición 1.4.2.2.- (vecindario de k-distancia(p))

Dada la k-distancia(p), el vecindario de k-distancia(p) (denotado por $N_k(p)$) es el conjunto que contiene a cada objeto cuya distancia a p es menor o igual a la k-distancia(p), o sea,

$$N_k(p) = \{q \in D - \{p\} / d(p,q) \leq k\text{-distancia}(p)\}$$

Los objetos q son conocidos como los k-vecinos más cercanos de p. Nótese que puede darse que $|(N_k(p)| > k$, al poder existir puntos q que se encuentren a la misma distancia de p.

Definición 1.4.2.3.- (distancia de acceso de un objeto p con respecto a un objeto o)

Sea k un entero positivo. Se define la distancia de acceso (*reachability distance*) de un objeto p con respecto a un objeto o como

$$\text{reach-dist}_k(p,o) = \text{máx}\{k\text{-distancia}(o), d(p,o)\}$$

Si un objeto p se encuentra alejado de otro objeto o, entonces la distancia de acceso entre ellos es su correspondiente distancia. Sin embargo, si los objetos se encuentran lo suficientemente cerca, su distancia es sustituida por la k-distancia(o). Esto se hace con el fin de evitar las posibles fluctuaciones que se puedan dar si la cardinalidad de $N_k(o)$ fuera mayor que k.

Definición 1.4.2.4.- (densidad local de acceso de un objeto p)

Se define la densidad local de acceso (*local reachability density*) de un objeto p como

$$lrd_k(p) = 1 / \left(\frac{\sum_{o \in N_k(p)} \text{reach-dist}_k(p,o)}{|N_k(p)|} \right) \tag{1.23}$$

Intuitivamente, cuanto más cerca se encuentre p de sus k vecinos, mayor será la densidad local de acceso de p. Además, se supone que no existen observaciones duplicadas, ya que, en tal caso, puede darse que el sumatorio de las distancias de acceso (reach-dist) valga 0 y se tenga una densidad ∞. Finalmente,

Definición 1.4.2.5.- (Local Outlier Factor de un objeto p)

Se define el Local Outlier Factor de p como

$$LOF_k(p) = \frac{\sum_{o \in N_k(p)} \frac{lrd_k(o)}{lrd_k(p)}}{|N_k(p)|} \tag{1.24}$$

que es equivalente a

$$LOF_k(p) = \frac{\sum_{o \in N_k(p)} lrd_k(o)}{lrd_k(p)|N_k(p)|} \tag{1.25}$$

El LOF de un objeto p captura el grado de anomalía de p. Es el ratio de la media de la densidad lrd de p y sus k-vecinos más cercanos. Se puede observar

que, cuanto más baja sea la densidad de p y más altas sean las densidades de los k-vecinos, mayor será el valor de LOF de p (y, posiblemente, represente un outlier). La idea para poder detectar un outlier consiste en que un outlier albergará una densidad lrd menor comparada a la densidad de sus k-vecinos. Se puede probar que para las observaciones que se encuentran dentro de un cluster y, por tanto no se consideran anómalas, $LOF \sim 1$. Mientras que valores significativamente superiores a 1 indican valores atípicos. Sin embargo, no existe una regla definida para determinar cuándo un punto puede ser considerado un outlier y tal decisión dependerá del contexto del problema.

1.4.3. Multidimensional Scaling (MDS)

Se realizará a continuación una breve descripción sobre la técnica de reducción de dimensionalidad utilizada en este trabajo, Multidimensional Scaling (MDS), una de las técnicas más utilizadas en el análisis multivariante [21].

La reducción de la dimensionalidad consiste en, a grandes rasgos, representar datos multidimensionales en espacios de una menor dimensión, donde se conserve la mayor parte posible de la estructura original de los datos. Constituye una parte importante dentro del análisis de datos multivariados ya que representa una posibilidad para la interpretación gráfica de los datos si se reducen a un espacio de 2 o 3 dimensiones.

Sean n observaciones con p atributos, y sus respectivas similitudes o distancias d_{ij} en el espacio original con $i, j = 1, ..., n$.

MDS es un algoritmo que se encarga de construir n puntos $y_i \in \mathbb{R}^r (r < p)$ tal que:

$$\widehat{d_{ij}}^2 := ||y_i - y_j||^2 \approx d_{ij}^2, \quad i, j = 1, ..., n \tag{1.26}$$

siendo $||.||$ la norma euclídea (luego, $\widehat{d}$ es la distancia euclídea en $\mathbb{R}^r$) y r la dimensión del espacio reducido.

Para la construcción de los puntos, MDS se basa en la idea de preservar el orden de las distancias. O sea, teniendo n observaciones de un espacio p-dimensional, se tienen $\frac{n(n-1)}{2}$ distancias d_{ij} (con $i = 1, ..., n-1; j = 2, ..., n$) puesto que se considera que son simétricas ($d_{ij} = d_{ji}$) y que $d_{ii} = 0$. Entonces, dichas distancias se pueden ordenar tal que

$$d_{i_1 j_1} \leq d_{i_2 j_2} \leq d_{i_3 j_3} \leq ... \leq d_{i_M j_M}, \tag{1.27}$$

con $M = \frac{n(n-1)}{2}$.

El objetivo consiste en obtener n puntos de un espacio r-dimensional tal que se conserve el mismo orden con el mínimo error cometido posible

$$\widehat{d}_{i_1 j_1} \leq \widehat{d}_{i_2 j_2} \leq \widehat{d}_{i_3 j_3} \leq ... \leq \widehat{d}_{i_M j_M} \tag{1.28}$$

La función objetivo a minimizar propuesta por Kruskal [22], conocida como función de *stress*, es la siguiente:

$$S(x_1, ..., x_n) = \sqrt{\frac{\sum_{i<j}(d_{ij} - \widehat{d}_{ij})^2}{\sum_{i<j} d_{ij}^2}}, \quad (i = 1, ..., n-1; j = 2, ..., n) \tag{1.29}$$

La solución para este problema de optimización se puede obtener mediante el método del gradiente cuyo procedimiento se puede encontrar en [22] y fundamentos teóricos en [23].

Si bien MDS no constituye una técnica específica para detectar outliers, se considera de especial utilidad ya que permite visualizar gráficamente, de una forma aproximada y que puede llegar a ser muy precisa, la distribución de datos multidimensionales. Esto facilita la comprensión de la estructura de los datos y permite la elaboración de un diagnóstico gráfico para la clasificación de outliers. Sin embargo, tal diagnóstico por sí solo puede llegar a ser insuficiente y es altamente recomendable su combinación con alguna técnica analítica.

1.4.4. Isolation Forest

Una de las técnicas multidimensionales más utilizadas en la detección de observaciones anómalas [24] es **Isolation Forest**. Tal como se describe en [25], esto se debe a su eficiencia computacional y a su robustez frente a los problemas de masking y swamping.

Isolation Forest es un método no paramétrico y basado en proyecciones. A diferencia de los métodos basados en análisis Cluster y en los vecinos más cercanos, no necesita tener definida una distancia, y calcula el grado de anomalía o *anomaly score* a través de la estructura de un árbol de decisión (1.3.2).

Isolation Forest se basa en la idea de que las observaciones anómalas son "pocas y diferentes" y, por lo tanto, son susceptibles a estar aisladas. Entonces, organizados los datos con estructura de árbol, se espera que un outlier se encuentre en la parte alta del árbol y una observación normal atraviese las ramas hasta la parte más baja del árbol. Es por esto que, a la hora de medir su grado de anomalía, se utiliza la longitud de la rama para cada observación.

Dado un dataset $D = \{x_i \in \mathbb{R}^m, i = 1, ..., n\}$ $(n, m \in \mathbb{N})$, el algoritmo Isolation Forest se puede resumir en los siguientes pasos:

1. Escoger de forma aleatoria una submuestra de un tamaño (considerablemente menor al tamaño de la muestra) para construir un árbol de decisión binario, llamado *Isolation Tree* o *iTree*. Se demuestra que esta elección suaviza el masking y swamping.
2. Para tal submuestra, se comienza con todos los datos agrupados en el nodo principal del *iTree* y se comienzan a dividir los datos en distintos nodos. Para

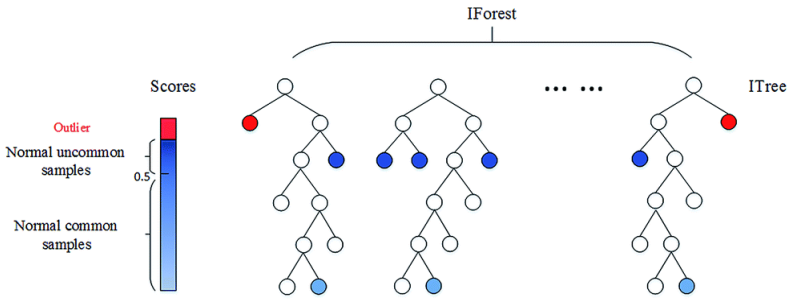


Figura 1.5. Algoritmo Isolation Forest.

cada nodo que se obtenga tras la división, el proceso continúa hasta que se tenga aislada una sola observación en el nodo o se alcance la profundidad máxima predefinida. Donde, en cada división, se selecciona aleatoriamente una variable j de entre las m variables que se tienen y se toma, también de forma aleatoria, un valor v dentro del rango de la variable j para dividir los datos en dos regiones, $D_L = \{X|X_{ij} < v\}$ y $D_R = \{X|X_{ij} \geq v\}$, $\forall i \in \{1, ..., n\}$.

3. Una vez construido el *iTree*, para cada observación de la muestra completa, se mide la longitud del camino recorrido, tal que $\forall X \in D$, se define $h(X)$ como el número de ramas que atraviesa una observación X desde el nodo principal hasta su respectivo nodo terminal.
4. Se repiten los pasos 1, 2 y 3 un número prefijado N de veces. Se construye así, un bosque con N *iTrees*.
5. Finalmente, teniendo el Isolation Forest construido, se mide el grado de anomalía o el *anomaly score*, s, de cada observación $X \in D$ de la siguiente forma:

$$s(X, n) = 2^{-\frac{E[h(X)]}{c(n)}} \tag{1.30}$$

siendo

- $n :=$ número de observaciones de la muestra.
- $E[h(X)] :=$ promedio de ramas que recorre la observación X en los N *iTrees*.
- $c(n) :=$ esperanza de $h(X)$ dadas n observaciones, estimada como

$$c(n) = 2H(n-1) - \left(\frac{2(n-1)}{n}\right) \tag{1.31}$$

donde $H(i)$ es el i-ésimo número armónico y se puede estimar como $\ln(i) + 0.5772156649$ (constante de Euler).

Se tienen las siguientes indicaciones generales:

- (a) Si $s(X, n) \to 1 \Rightarrow X$ Outlier.

- (b) Si $s(X, n) \lll 0.5 \Rightarrow X$ observación normal.
- (c) Si todas las observaciones obtienen $s \approx 0.5$, entonces el conjunto de datos no contiene ninguna anomalía distinta.

Sin embargo, como en LOF (1.4.2), es imprescindible tener en cuenta el contexto específico del problema.

1.5. Optimización de Hiperparámetros

Muchos modelos de Machine Learning contienen parámetros que no se pueden aprender a partir de los datos de entrenamiento y, por lo tanto, deben ser establecidos por el analista. Por ejemplo,

- N, λ y γ en XGBoost (1.3.4).
- el número de árboles en Random o Isolation Forest (1.3.3 y 1.4.4).
- el número k de vecinos en kNN (1.3.1) y LOF (1.4.2).

A estos parámetros se les conoce como hiperparámetros y son uno de los factores clave en el Machine Learning, ya que los resultados de un modelo pueden depender, en gran medida, del valor que tomen sus hiperparámetros. Sin embargo, no se puede conocer de antemano cuál es el valor adecuado para un hiperparámetro. La forma más común de encontrar los valores óptimos es probando diferentes posibilidades.

Para los algoritmos de ML supervisado, tal como se procede en [14] y [16] en problemas similares, en este trabajo se aplicará el método exhaustivo *Grid Search* junto con el método de validación *k-Fold Cross-Validation*. En pocas palabras:

- Se define un conjunto de posibles valores para los hipérparametros.
- Para cada uno de dichos valores:
  1. Se separan las observaciones en k conjuntos.
  2. Se entrenan k modelos, excluyendo un conjunto distinto en cada modelo.
  3. Se mide el error de cada modelo, aplicando la función de pérdida en el conjunto excluido.
  4. Se agregan los k errores en un error final.
- Se eligen aquellos hipérparametros que resulten en un menor error final.

En la Sección 2.7 del capítulo 2 se describen los hipérparametros y los valores óptimos encontrados para los modelos de ML supervisado utilizados.

Por otro lado, para los algoritmos de ML no supervisado, se tienen datos no etiquetados y, por lo tanto, se desconoce el valor real de etiqueta de los datos. Es por esto que, estimar el error cometido al aplicar modelos no supervisados se vuelve una tarea más difícil. En este caso, en la sección 3.3 se explica el procedimiento seguido para la elección de los hiperparámetros. Dicho procedimiento se

puede resumir en la comparación entre los valores recomendados, es decir, valores que, en general, proporcionan resultados aceptables; y valores relacionados con los hipérparametros obtenidos en los modelos de ML supervisados que se aplicaron previamente.

2

Escenario 1

En este capítulo se presentará el problema real que se propone resolver en este trabajo, y se estimará una solución del mismo. En primer lugar, se describirán los antecedentes del estudio, junto con la población, muestra, variables de estudio y objetivos. A continuación, se realizará una estadística descriptiva de la muestra. Seguidamente, se aplicarán los modelos de Machine Learning supervisado, introducidos en la sección 1.3. Por último, se realizará una comparación de los resultados.

2.1. Antecedentes

En un puerto se producen una serie de operaciones sobre los barcos. Las más importantes y que llegan a ocupar el 65 % de la estadía en puerto se refieren a la descarga y carga de contenedores u otros productos en el barco.

Uno de los estados con los que trabajan las terminales es el ETC (*Estimated Time of Completed*), que corresponde a la fecha/hora estimada de finalización de trabajos.

El estudio de este problema pretende obtener un cálculo del ETC para el puerto de Valencia, con una confiabilidad suficiente para que sirva para Port Control para la estimación de la finalización de los trabajos de carga/descarga y, por tanto, que facilite la organización del tráfico portuario.

Los datos fueron suministrados por la empresa *Hiades Business Patterns S.L*, respecto a operaciones que tuvieron lugar en el puerto de Valencia en 2022. También se accedió a la base de datos pública de *Aemet* para obtener datos sobre el viento, con origen la estación metereológica con datos disponibles más cercana: Viveros, en Valencia.

Además, para analizar la muestra, preprocesar los datos y aplicar los modelos de Machine Learning se utilizó el lenguaje de programación Python, principalmente con las librerías *Numpy, Pandas* y *scikit-learn*. Todos los códigos utilizados se pueden encontar en: https://github.com/Roberto-Calvo/TFG

2.2. Variables de estudio

- *Hora_Inicio_Servicio*_Entrada ≡ Fecha y hora de inicio de servicio de practicaje durante la entrada (denotado por *Inicio*).
- *Hora_Fin_Servicio*_Salida ≡ Fecha y hora a la que termina el servicio de salida del buque del puerto (denotado por *Fin*).
- *Seconds* ≡ Tiempo que duró la operación medido en segundos, calculado con la librería *Datetime* en Python como:

$$Seconds = Fin - Inicio \tag{2.1}$$

- GT ≡ Arqueo bruto del buque, medido en Toneladas Moorson, que corresponde al volumen de los espacios cerrados que tiene el busque.
- *Eslora* ≡ Eslora del buque, expresada en metros.
- *Manga* ≡ Manga del buque expresada en metros.
- *Calado_Popa_Entrada* ≡ Calado de popa durante la entrada del buque, expresado en metros,.
- *Calado_Popa_Salida* ≡ Calado de popa durante la salida del buque, expresado en metros.
- *Gruas* ≡ Grúas disponibles para la operación. Calculada como el número de grúas que hubo disponibles el primer día de llegada a puerto.
- *DockCode* ≡ Valor alfanumérico asignado por las autoridades portuarias para identificar los muelles.
- *Mes* ≡ Mes en el que llega el barco al puerto, calculada a partir de *Hora_Inicio_Servicio*_Entrada.
- *Semana* ≡ Semana del año en la que llega el barco al puerto, calculada en formato ISO-8601 y a partir de *Hora_Inicio_Servicio*_Entrada.
- *DiaSemana* ≡ Día de la semana en el que llega el barco al puerto, calculado a partir de *Hora_Inicio_Servicio*_Entrada.
- *Viento* ≡ Velocidad media del viento para cada día de 2022, medida en m/s y tomada de la estación metereológica de Viveros en Valencia.

Se tiene pues, la clasificación 2.1 por tipo de variable:

Continuas	**Discretas**
GT	DockCode
Eslora	Semana
Manga	Mes
Calado_Popa_Entrada	DiaSemana
Calado_Popa_Salida	Gruas
Viento	
Seconds	

Tabla 2.1.

2.3. Población y muestra

La muestra inicial contenía más de 6000 observaciones, con respecto a datos de barcos que atracaron en el puerto de Valencia durante el año 2022. Sin embargo, no todas esas observaciones correspondían a operaciones de carga/descarga de barcos. Se consideraron como observaciones válidas aquellas en las que, durante toda su estancia en el puerto, existiera como mínimo 1 día en el que se tuviera al menos 1 grúa operativa en el lugar donde atracó el barco. Para obtener dichas observaciones, se procedió de la siguiente forma:

1. Para cada observación (barco), se calculó el intervalo de días transcurridos en el muelle en el que atracó, ya que se disponía de fecha y hora, tanto de entrada como de salida al puerto.
2. Para cada día transcurrido, se comprobaron las grúas disponibles por muelle y localización de atraque. Esto se pudo hacer debido a que se disponía de las coordenadas de los amarres del barco al muelle, así como un fichero con los estados y localización de las grúas para cada día del año. Para filtrar por localización se procedió de la siguiente manera.
   - Sea la grúa i para un determinado día del año, con x_i e y_i su longitud y latitud, respectivamente.
   - Por otro lado, sean *Longitud_Amarre*, *Latitud_Amarre*, *Longitud_Amarre_2* y *Latitud_Amarre_2* las localizaciones de los amarres del barco al puerto.
   - Entonces, en función de las coordenadas de los respectivos amarres, se trazó una región rectangular de aceptación con un margen de, aproximadamente, 30 metros que permitiese a las grúas alejarse de los amarres como máximo 30 metros. De tal forma que, para que la grúa i fuera aceptada como disponible en la operación se debía cumplir que:

$$min(longitud) - 0.00025 \leq x_i \leq max(longitud) + 0.00025 \quad (2.2)$$
$$min(latitud) - 0.00025 \leq y_i \leq max(latitud) + 0.00025 \quad (2.3)$$

   siendo las coordenadas bajo el sistema de referencia EPSG:4326

$$min(longitud) = \text{mín}\{Longitud\_Amarre, Longitud\_Amarre\_2\}$$
$$max(longitud) = \text{máx}\{Longitud\_Amarre, Longitud\_Amarre\_2\}$$
$$min(latitud) = \text{mín}\{Latitud\_Amarre, Latitud\_Amarre\_2\}$$
$$max(latitud) = \text{máx}\{Latitud\_Amarre, Latitud\_Amarre\_2\}$$
$$margen = 0.00025 \approx 30 \text{ metros}$$

3. Se calculó la media aritmética para cada operación, es decir, el promedio del número total de grúas disponibles para cada día por el número de días distintos transcurridos en el puerto.

4. Por último, se consideraron como operaciones de carga/descarga solo aquellas observaciones dónde como mínimo participase 1 grúa en la totalidad de su estancia en el puerto. Es decir, aquellas cuya media aritmética fuese estrictamente mayor que 0.

Por simplicidad, se eliminaron las observaciones con valores desconocidos, resultando una muestra con 1037 operaciones realizadas durante 2022 en 4 muelles del puerto de Valencia. En concreto, los correspondientes a los códigos 32, 36, 75 y 76 (variable *DockCode*). Por tanto, la población de estudio se supondrá como todos los barcos que realicen una operación de carga/descarga en dichos muelles.

Además, la división aleatoria en datos de entrenamiento y test fue con una proporción 80/20 respectivamente, es decir, el 80 % de la muestra (830 operaciones) se guardó para entrenar a los modelos y el 20 % (207 operaciones) se reservó para validar dichos modelos.

2.4. Objetivo

Se pretende estimar la variable *Seconds* (denotada por Y, variable de salida) a partir del resto de variables anteriormente descritas (denotadas por X, variables de entrada), con la confiabilidad suficiente.

En primer lugar, se realizará un análisis exploratorio mediante una estadística descriptiva sobre cada una de las variables. Luego, se aplicarán 3 modelos de Machine Learning: kNN, Random Forest y XGBoost para, finalmente, comparar los resultados.

Nota: Con fines orientativos, la variable *Seconds* irá acompañada de su equivalente en horas y en días.

2.5. Estadística descriptiva

2.5.1. Variable de salida

Se tiene el siguiente histograma 2.1 y tabla de medidas 2.2 de la variable *Seconds*.

Además, se tiene el box-plot 2.2, donde, a partir del límite de 209547 segundos, calculado como $3 \cdot (Q_3 - Q_1)$, con Q_1 y Q_3 el primer y tercer cuartil, se observa que, entre el mencionado límite y 400000 segundos, existe un conjuntos de datos considerablemente agrupados. Y, de 400000 segundos en adelante, en comparación, las observaciones se encuentran significativamente dispersas.

2.5.2. Variables predictoras

Se distinguirá entre las variables predictoras continuas y las discretas.

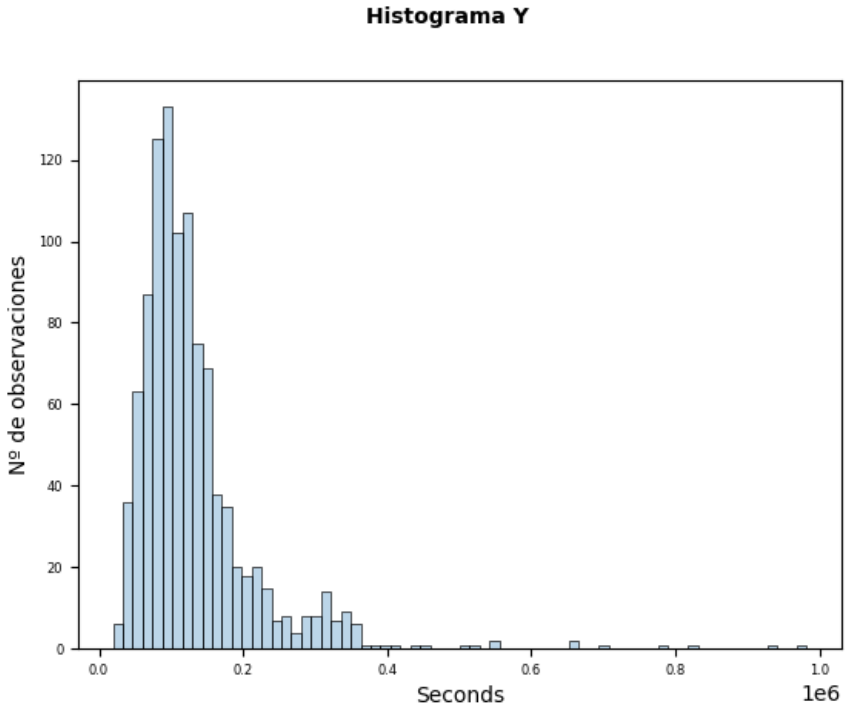


Figura 2.1. Histograma de Y.

	Seconds	Horas	Días
Media aritmética	132978.1871	36.9384	1.5391
Mediana	111349.0000	30.9302	1.2888
Desviación estándar	91860.5221	25.5168	1.0632
Mínimo	19206.0000	5.3350	0.2223
Máximo	982773.0000	272.9925	11.3747

Tabla 2.2. Medidas descriptivas de Y.

Variables predictoras continuas

Se tienen los siguientes histogramas 2.3 y box-plots 2.4 de las variables *GT, Eslora, Manga, Calado_Popa_Entrada (CPE), Calado_Popa_Salida (CPS) y Viento*.

Se observan distribuciones asimétricas de los datos, así como la posible presencia de observaciones anómalas para las variables *GT y Viento*. Además, se tomaron las medidas mostradas en 2.3:

Variables predictoras discretas

Se tienen los siguiente gráficos de barras 2.5, 2.6 de las variables discretas *DockCode, Mes, DiaSemana, Gruas y Semana*.

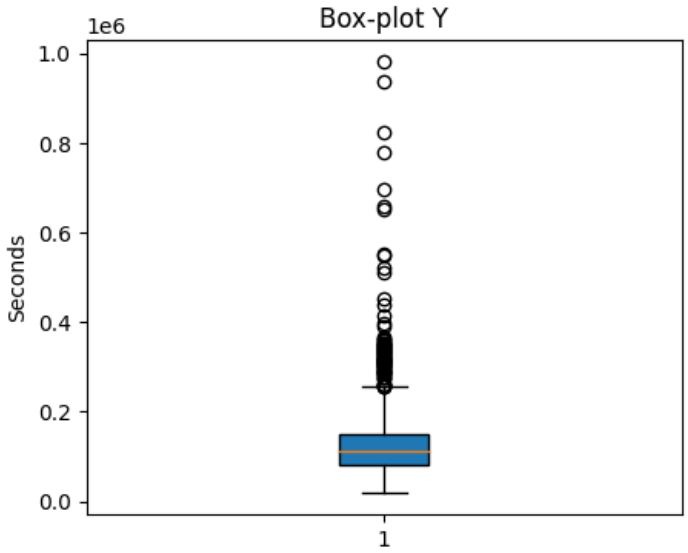


Figura 2.2. Box-plot de Y.

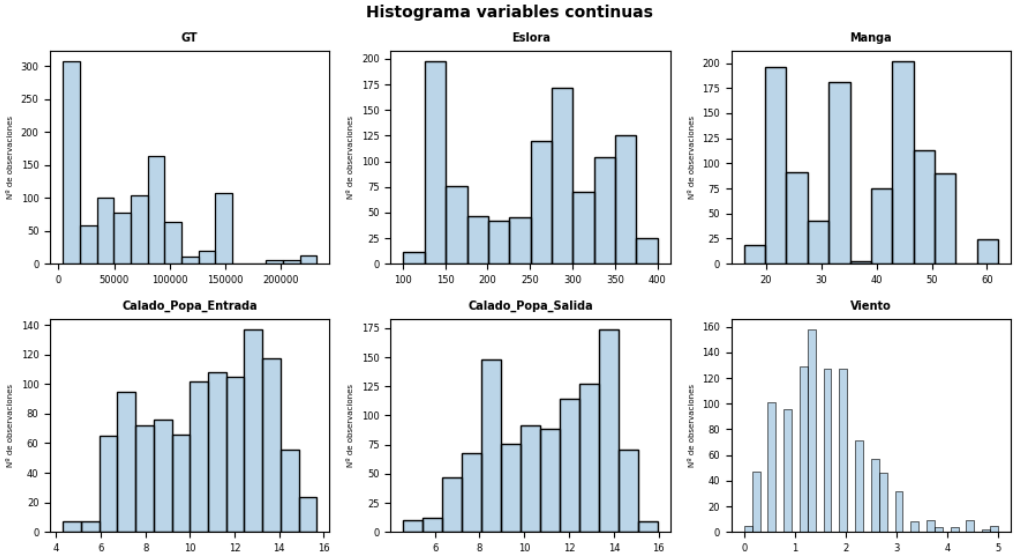


Figura 2.3. Histogramas de las variables predictoras continuas.

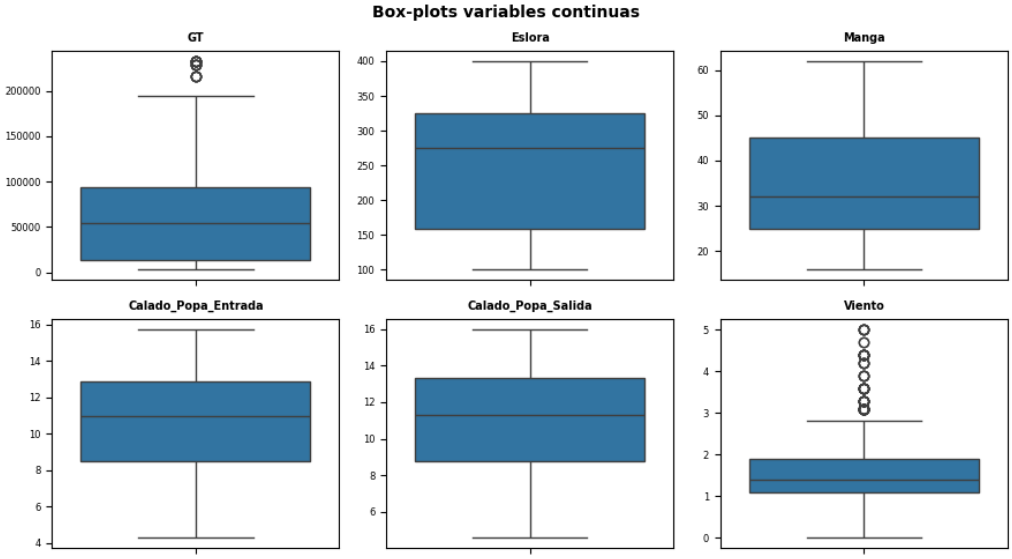


Figura 2.4. Boxplots de GT, Eslora, Manga, CPE, CPS y Viento.

	GT (tM)	Eslora (m)	Manga (m)	CPE (m)	CPS (m)	Viento (m/s)
Media arit.	64928.2305	256.7840	36.2240	10.7000	11.0493	1.6043
Mediana	54193.0000	275.0000	32.0000	11.0000	11.3000	1.4000
Desviación est.	51536.1368	83.0671	10.7547	2.5950	2.4785	0.8626
Mínimo	3850.0000	101.0000	16.0000	4.3000	4.6000	0.0000
Máximo	232618.0000	400.0000	62.0000	15.7000	15.9500	5.0000

Tabla 2.3. Medidas descriptivas de las variables predictoras continuas.

En la variable *DockCode*, destaca la categoría 36, con un porcentaje del 62.58 % (649 operaciones), es decir, el 62.58 % de las observaciones (649 operaciones) están vinculadas al muelle con código 36.

Además, para la variable *Gruas* se decidió unir los valores de categoría de 3 en adelante como > 2, con el fin de evitar posibles problemas a la hora de dividir los datos durante la validación cruzada, tal como se recomienda en [27], de forma que resultó el gráfico de barras 2.7.

2.6. Preprocesado de los datos

- Con el fin de evitar problemas de escala, se estandarizaron las variables continuas; de tal forma que a cada variable se le restó su media muestral y se dividió por su desviación estándar. $\left(Z = \frac{X-\mu}{\sigma}\right)$
- En la variable *DockCode* se aplicó **One-hot encoding**, tal como se recomienda en [28] y [29]. Esto se hizo para evitar que valores de categoría mayo-

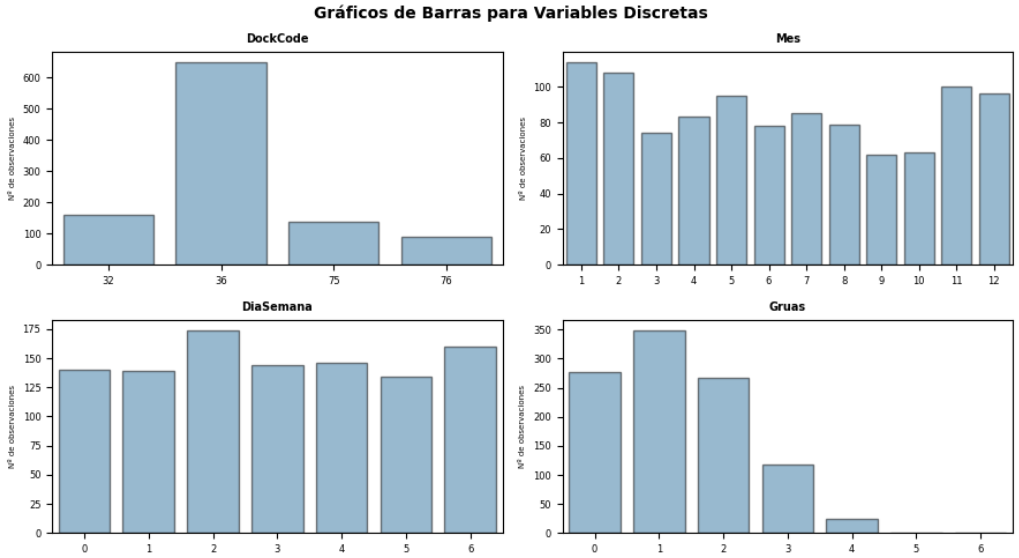


Figura 2.5. Gráficos de barras de DockCode, Mes, DiaSemana y Gruas.

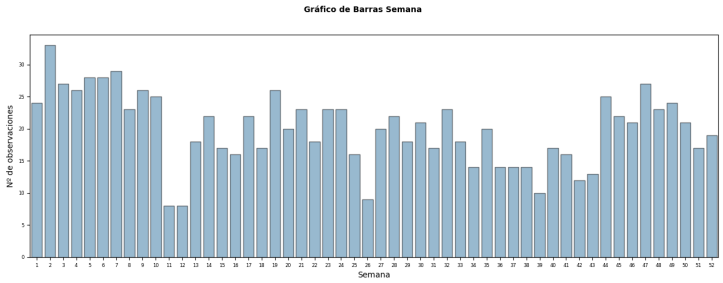


Figura 2.6. Gráfico de barras de Semana.

res, como por ejemplo 76, tuvieran una ponderación mayor en el modelo en comparación con valores de categoría menores, como por ejemplo 32, ya que *DockCode* no se considera una variable ordinal.

Nota: One-hot encoding es una técnica comúnmente utilizada en el preprocesamiento de datos. En general, se utiliza con variables discretas no ordinales y consiste en construir tantas variables binarias como posibles valores tenga la variable original. Cada variable binaria representa la ausencia o presencia de cada posible valor. Por ejemplo, para un conjunto de datos, sea la variable *Color* que tiene 3 posibles valores: Rojo, Verde o Azul. Entonces, al aplicar One-hot encoding se construirán 3 nuevas variables (*Color_Rojo*, *Color_Verde* y *Color_Azul*) que valdrán 0, si no se contiene el respectivo color, y 1, si se contiene.

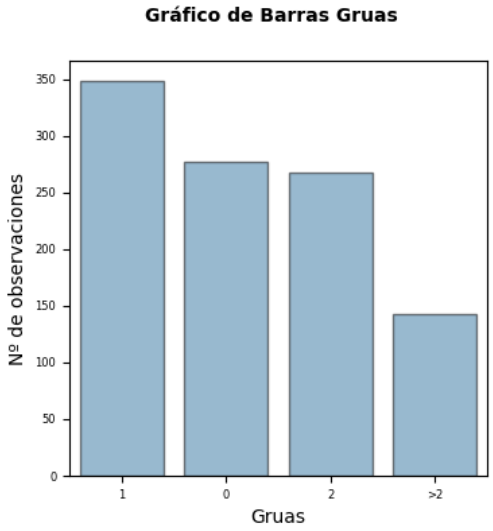


Figura 2.7. Gráfico de barras de Gruas (reagrupada).

2.7. Modelos

2.7.1. k-Nearest Neighbour (kNN)

A continuación, se aplicará el modelo de ML supervisado kNN con los datos preprocesados de la muestra. Tal como se describe en la sección 1.3.1 del capítulo 1, primero se definirá la forma de medir la distancia entre las observaciones. Luego, se mostrará el valor óptimo encontrado para los hiperparámetros y los errores cometidos mediante las métricas RMSE y MAE (1.1.3).

Similaridad de Gower

Al disponer de variables continuas y discretas, ha de definirse una forma de medir la distancia entre observaciones acorde a la presencia de variables mixtas. Es por esto que, tal y como se aplica en [30], las distancias se estimarán a través del coeficiente de similaridad propuesto por Gower en [31].

Sea $X_i = (X_{i1}, ..., X_{ip})$ la observación i. Se tienen p_1 variables continuas, p_2 variables dicotómicas y p_3 variables discretas ($p_1 + p_2 + p_3 = p$). Entonces, la distancia entre la observación i y la observación i' se define como:

$$d(i,i') = 1 - \sqrt{\frac{\sum_{k=1}^{p_1}\left(1 - \frac{|x_{ik} - x_{i'k}|}{r_k}\right) + c_{1ii'} + m_{ii'}}{p_1 + (p_2 - c_{0ii'}) + p_3}} \tag{2.4}$$

donde $c_{1ii'}$ es el número de coincidencias de la forma (1,1) y $c_{0ii'}$ el número de coincidencias (0,0), para las p_2 variables dicotómicas, $m_{ii'}$ es el número de coincidencias para las p_3 variables discretas y r_k es el rango (o distancia) para la k-ésima variable continua.

Descripción

Se utilizaron los siguientes módulos en Python:

- *KNeighborsRegressor*, *mean_absolute_error* y *mean_squared_error*. Módulos de la librería *Sci-kit Learn*, para entrenar y validar el modelo.
- *RepeatedKFold* y *RandomizedSearchCV*. Módulos también de *Sci-kit Learn*, para la optimización de hiperparámetros.

Además, se definió una función en Python para poder calcular la distancia entre observaciones mediante el coeficiente de similaridad de Gower, ya que dicha métrica no venía incluida en el paquete *KNeighborsRegressor*.

Hipérparametros

Como se comentó en la sección 1.5 del capítulo 1, el procedimiento seguido para obtener los valores de los hipérparametros fue aplicar el método exhaustivo *Grid Search* junto con el método de validación *k-Fold Cross-Validation*. Se tiene la tabla 2.4 de hipérparametros para kNN:

Hiperparámetro	**Descripción**	**Rango de valores**	**Óptimo**
n_neighbours	Número de observaciones más cercanas a tener en cuenta	Enteros entre 1 y 100	27

Tabla 2.4. Optimización de hiperparámetros para kNN.

Resultados

Teniendo en cuenta el valor óptimo para los hiperparámetros de la Tabla 2.4, se entrenó un modelo con dicho valor, obteniendo la tabla de errores 2.5. Se puede observar un RMSE considerablemente mayor al MAE. Esto parece razonable debido a la mayor robustez del MAE frente a observaciones alejadas.

	Seconds	**Horas**	**Días**
RMSE	90074.4690	25.0207	1.0425
MAE	52985.5937	14.7182	0.6133

Tabla 2.5. RMSE y MAE kNN.

2.7.2. Random Forest

A continuación, se aplicará el modelo Random Forest, cuya descripción se puede encontrar en la sección 1.3.3 del capítulo 1, sobre la muestra. En primer

lugar, se procederá a enumerar las librerías utilizadas para aplicar el modelo. Luego, se mostrará el valor óptimo encontrado para los hiperparámetros y los errores cometidos mediante las métricas RMSE y MAE (1.1.3).

Descripción

Se utilizaron los siguientes módulos en Python:

- *RandomForestRegressor*, *mean_absolute_error* y *mean_squared_error*. Módulos de la librería *Sci-kit Learn*, para entrenar y validar el modelo.
- *RepeatedKFold* y *RandomizedSearchCV*, también de *Sci-kit Learn*, para el proceso de optimización de hiperparámetros.

Hiperparámetros

Como se comentó en la sección 1.5 del capítulo 1, el procedimiento seguido para obtener los valores de los hipérparametros fue aplicar el método exhaustivo *Grid Search* junto con el método de validación *k-Fold Cross-Validation*. Se tiene la Tabla 2.6 de hipérparametros para Random Forest.

Resultados

Teniendo en cuenta el valor óptimo para los hiperparámetros de la Tabla 2.6. Se entrenó un modelo con dichos valores, obteniendo la Tabla de errores 2.7. Tal como para kNN, se da una notable diferencia entre RMSE y MAE. Cabe destacar también un mejor resultado (menor RMSE y MAE) que kNN.

2.7.3. XGBoost

El último modelo de ML supervisado que se aplicará es XGBoost, cuya introducción se puede encontrar en la sección 1.3.4 del capítulo 1. Primero, se procederá a enumerar las librerías utilizadas. Entonces, se mostrará el valor óptimo encontrado para los hiperparámetros y los errores cometidos mediante las métricas RMSE y MAE (1.1.3).

Descripción

Se utilizaron los siguientes módulos en Python:

- *XGBRegressor*. Módulo de la librería *xgboost*, para entrenar el modelo.
- *mean_absolute_error* y *mean_squared_error*. Módulos de la librería *Sci-kit Learn*, para entrenar y validar el modelo.
- *RepeatedKFold* y *RandomizedSearchCV*. Módulos también de *Sci-kit Learn*, para la optimización de hiperparámetros.

Hiperparámetro	Descripción	Rango de valores	Óptimo
n_estimators	Número de árboles incluidos en el modelo	[50, 100, 1000, 2000]	2000
max_features	Número de predictores considerado para realizar la división	[1, 3, 5, 7]	1
max_depth	Profundidad máxima que pueden alcanzar los árboles	[None, 3, 5, 10, 20]	10
min_samples_split	Número mínimo de observaciones que debe tener un nodo para que pueda dividirse	2	2
min_samples_leaf	Número mínimo de observaciones que deben tener los nodos terminales	1	1
max_leaf_nodes	Número máximo de nodos terminales que pueden tener los árboles	None (ilimitado)	None
min_impurity_decrease	Ganancia de información mínima requerida para realizar una división	0.0	0.0
criterion	Medida de impureza	[squared_error, absolute_error]	absolute_error

Tabla 2.6. Optimización de hiperparámetros para Random Forest.

	Seconds	Horas	Días
RMSE	89247.4750	24.7909	1.0329
MAE	48054.8972	13.3486	0.5562

Tabla 2.7. RMSE y MAE Random Forest.

Hiperparámetros

Como se comentó en la sección 1.5, el procedimiento seguido para obtener los valores de los hipérparametros fue aplicar el método exhaustivo *Grid Search* junto con el método de validación *k-Fold Cross-Validation*. Se tiene la Tabla 2.8 de hipérparametros para XGBoost.

Resultados

Teniendo en cuenta el valor óptimo para los hiperparámetros de la Tabla 2.8, se entrenó un modelo con dichos valores, obteniendo la Tabla de errores 2.9. Como en los anteriores modelos, surge también la considerable diferencia entre

Hiperparámetro	Descripción	Rango de valores	Óptimo
n_estimators	Número de árboles incluidos en el modelo	[1000, 2000, 3000, 4000, 5000]	5000
max_depth	Profundidad máxima que pueden alcanzar los árboles	[None, 2, 3, 4, 5, 6]	6
learning_rate	Parámetro utilizado para prevenir el overfitting	[0.01, 0.05, 0.1]	0.1
min_child_weight	Número mínimo de observaciones que deben tener los nodos terminales	[1,3,5,7,9]	9
lambda	Término de regularización L2 para los pesos	1.0	1.0
alpha	Término de regularización L1 para los pesos	0.0	0.0
gamma	Ganancia de información mínima requerida para realizar una división	0.0	0.0

Tabla 2.8. Optimización de hiperparámetros para XGBoost.

RMSE y MAE. Cabe destacar un mejor resultado que kNN (menor RMSE y MAE) y peor resultado (mayor RMSE y MAE) que Random Forest.

	Seconds	Horas	Días
RMSE	90697.6026	25.1938	1.0497
MAE	50339.2971	13.9831	0.5826

Tabla 2.9. Errores para XGBoost.

2.8. Comparación

Una vez entrenados los tres modelos, los resultados totales se resumen en la Tabla 2.10.

Como ya se comenta a lo largo de la sección 2.7, la diferencia existente entre el **RMSE** y el **MAE** se debe a que esta última métrica es más robusta frente a observaciones alejadas de los datos. De ahí, que el error medido mediante RMSE se reduzca a aproximadamente la mitad si se toma mediante MAE. Por otro lado, destaca el **Random Forest** como el modelo con menor error cometido (tanto RMSE como MAE) de entre los tres modelos elegidos.

	kNN	**Random Forest**	**XGBoost**
RMSE, segundos	90074.4690	89247.4750	90697.6026
MAE, segundos	52985.5937	48054.8972	50339.2971
RMSE, horas	25.0207	24.7909	25.1938
MAE, horas	14.7182	13.3486	13.9831
RMSE, días	1.0425	1.0329	1.0497
MAE, días	0.6133	0.5562	0.5826

Tabla 2.10. Errores para los 3 modelos.

Sin embargo, el resultado se considera insuficiente debido a que estimar la finalización de las operaciones de carga y descarga con un error esperado de, aproximadamente, 13 horas, no parece ser información suficientemente útil para las terminales del puerto.

Por lo tanto, y aún en el mejor de los casos, el error es considerablemente grande. Ello puede ser debido a varias causas como por ejemplo, la insuficiencia del número de observaciones o la necesidad de más variables explicativas, entre otras. Otra causa, y que es la dirección que va a tomar el trabajo en el siguiente capítulo, puede ser la presencia de observaciones anómalas que agraven los resultados. Se tratará de clasificar las observaciones en normales o anómalas y, si existen observaciones anómalas, entender cuál es su origen.

3

Escenario 2

En este capítulo se tratará de identificar observaciones anómalas en el conjunto de datos del problema práctico, teniendo por objetivo mejorar los resultados obtenidos en el capítulo 2. Para ello, se pretende profundizar en las variables predictoras, o sea, en el espacio de características dado en la sección 2.2 del capítulo 2. Se utilizarán técnicas de Machine Learning no supervisado. Concretamente, *Isolation Forest*, *Multidimensional Scaling* y *Local Outlier Factor*. Finalmente, se mostrarán los resultados obtenidos y se llegará a una conclusión final.

3.1. Antecedentes

Los apartados de antecedentes, población y muestra, variables de estudio y estadística descriptiva son los mismos que se tienen en el capítulo 2, descritos en las secciones 2.1, 2.3, 2.2 y 2.5, respectivamente.

3.2. Objetivo

El principal objetivo consiste en conseguir una mayor precisión en los modelos de ML supervisados que se aplicaron en el capítulo 2, es decir, tratar de disminuir los errores obtenidos. Para ello, se estudiará la estructura de la muestra con el fin de comprobar si existe alguna relación entre posibles observaciones anómalas en el espacio de características X y las observaciones anómalas dadas en el espacio de etiquetas Y.

Además, se tratará de conocer la posible causa de las observaciones anómalas en el espacio de etiquetas Y. Para ello, se pondrá especial interés en aquellas operaciones cuya estancia en puerto haya sido mayor a 400000 segundos.

Nota: el estudio de las observaciones anómalas u outliers se realizará de forma independiente en los espacios de características y etiquetas. O sea, se estudiarán ambos espacios por separado, para luego compararlos y comprobar si pudiera existir alguna relación entre las anomalías dadas en ambos.

3.3. Modelos

A continuación se detallarán los resultados de los modelos de ML no supervisados aplicados.

3.3.1. Multidimensional Scaling (MDS)

Para poder aplicar MDS (1.4.3) se necesita tener definida una forma de medir la distancia o similaridad entre los datos. En este caso, se utilizará el coeficiente de similaridad de Gower, definido en 2.7.1.

Se aplicaron las siguientes librerías en Python:

- *gower*, librería que contiene el módulo *gower_matrix* que calcula la matriz de distancias mediante el coeficiente de similaridad de Gower.
- *sklearn.manifold*, de donde se importó el módulo MDS.
- *matplotlib.pyplot*, para dibujar los gráficos.

Se redujo el conjunto de datos a 2 dimensiones, y se obtuvo el Gráfico 3.1.

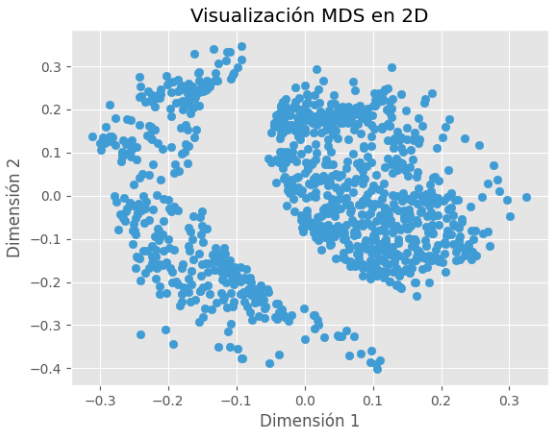


Figura 3.1. Gráfico MDS (r=2).

Además, en la Figura 3.2 se oscurecieron las observaciones consideradas como anómalas para la variable *Seconds*, Y, tales que $Y > 400000$ segundos, y se agruparon las observaciones por código de muelle (*DockCode*).

Gráficamente, las observaciones se agrupan por código de muelle con una mayor dispersión para los muelles 75 y 32. Sin embargo, las observaciones consideradas como outliers en Y no parecen tener un comportamiento diferente al resto de observaciones en X.

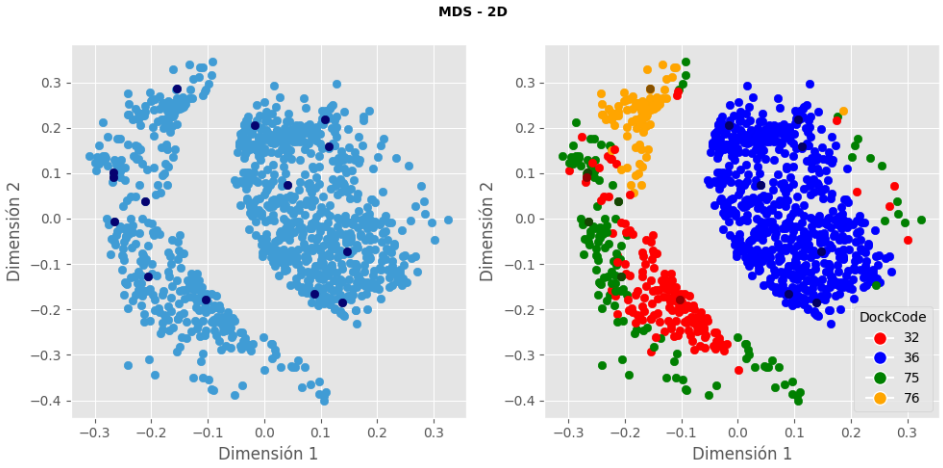


Figura 3.2. Gráfico MDS (r=2) remarcado y por DockCode.

3.3.2. Local Outlier Factor

De igual forma que en MDS, para poder calcular el Local Outlier Factor o LOF (1.4.2) de las observaciones, es indispensable tener definida una distancia o similitud. Como en los demás casos de este trabajo donde se necesita medir distancias, se utilizará el coeficiente de similaridad de Gower, definido en 2.7.1.

Por otro lado, el número k de vecinos más cercanos a considerar es un hiperparámetro y, por lo tanto, ha de ser fijado previamente a la aplicación del algoritmo. Existen métodos para encontrar el valor óptimo de k en conjuntos de datos no etiquetados, por ejemplo, Natural Neighbor [33]. Sin embargo, se tomó $k = 27$, debido a ser el valor óptimo obtenido en el capítulo 2 para kNN en 2.4 y por estar dentro de los límites de valores recomendados en la documentación de la librería sci-kit learn [34].

Se aplicaron las siguientes librerías en Python:

- *gower*, librería que contiene el módulo *gower_matrix* que calcula la matriz de distancias mediante el coeficiente de similaridad de Gower.
- *sklearn.neighbors*, de donde se importó el módulo *LocalOutlierFactor*.
- *seaborn*, para dibujar los gráficos.

Se calculó el respectivo LOF para cada observación de la muestra y se obtuvo el Histograma 3.3 donde se marcó el LOF para las observaciones consideradas como anómalas en la variable *Seconds* y se guardaron en la Tabla 3.1.

En [32] se estudian distintos umbrales para determinar a partir de qué valores considerar las observaciones como outliers. El umbral más conservativo que se obtuvo fue 1.4.

Considerando dicho umbral y que las puntuaciones obtenidas no se consideran significativamente lejanas a 1, las operaciones cuya duración fue mayor a

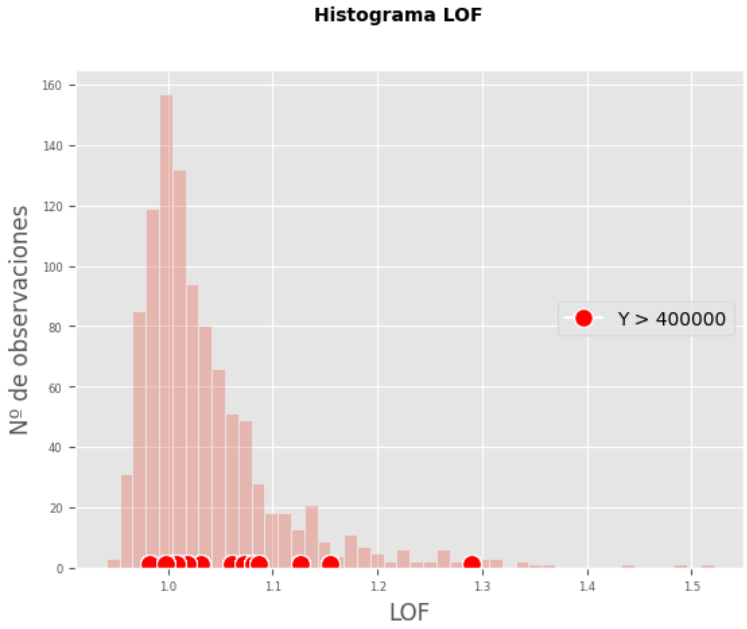


Figura 3.3. Histograma LOF de la muestra.

LOF	Seconds
1.0728	415053
1.0010	439492
1.0605	453753
0.9979	511288
1.0181	520740
1.0867	549120
1.1547	552884
0.9987	653986
1.0815	658348
1.0310	696150
0.9821	779492
1.2893	822910
1.1262	939600
1.0078	982773

Tabla 3.1. LOF de las operaciones tales que $Y > 400000$.

400000 segundos no serán clasificadas como outliers en el espacio de características estudiado. Además, se obtuvieron 3 observaciones tales que $LOF > 1.4$, identificadas en la Figura 3.4 y, con su respectivo valor de etiqueta en la Tabla 3.2. Por lo comentado en 2.5, no se considerarán como anomalías en el espacio Y.

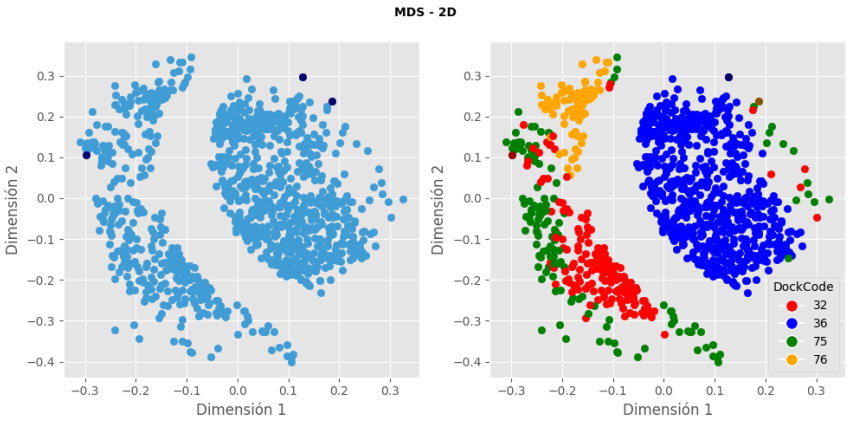


Figura 3.4. MDS. Marcadas las observacions con $LOF > 1.4$.

LOF	Seconds
1.4936	232551
1.4334	189120
1.5209	57097

Tabla 3.2. Operaciones tales que $LOF > 1.4$.

3.3.3. Isolation Forest

Por último, se entrenó un Isolation Forest (1.4.4) con 5000 *iTrees* y 32 observaciones por submuestra. El primer hiperparámetro se eligió por ser el número óptimo de árboles obtenido en el modelo XGBoost (descrito en 2.8) sumado al hecho de que se probó con distintos valores dentro del rango $[500, 5000]$ y la diferencia en ninguno de los resultados se consideró significativa. Se tomó el tamaño de submuestra 32 debido a que el valor que se suele recomendar para este parámetro es $\sqrt{n}$ con n el tamaño de la muestra ($\sqrt{1037} = 32.2025...$). Además, se tomaron valores próximos a 32 y tampoco hubo diferencia significativa.

Se calcularon y se agruparon los anomaly scores en el Histograma 3.5.

En [25], para el umbral s_0 del anomaly score, a partir del cual clasificar a las observaciones como potenciales outliers, se propone $s_0 = 0.6$. Sin embargo, también se debe tener en cuenta la indicación general que dice que si el anomaly score de todas las observaciones se aproxima a 0.5, entonces no se recomienda extraer conclusiones sobre posibles anomalías respecto a tal puntuación.

Como se puede observar en el Histograma 3.5, las puntuaciones se agrupan dentro del rango $[0.4428, 0.6133]$. Además, aquellas operaciones tales que $Y > 400000$ no obtuvieron una puntuación significativamente distinta al resto de las observaciones.

En este trabajo, se considerará que todas las puntuaciones se encuentran próximas a 0.5 y, por tanto, no se tendrá en cuenta el modelo Isolation Forest

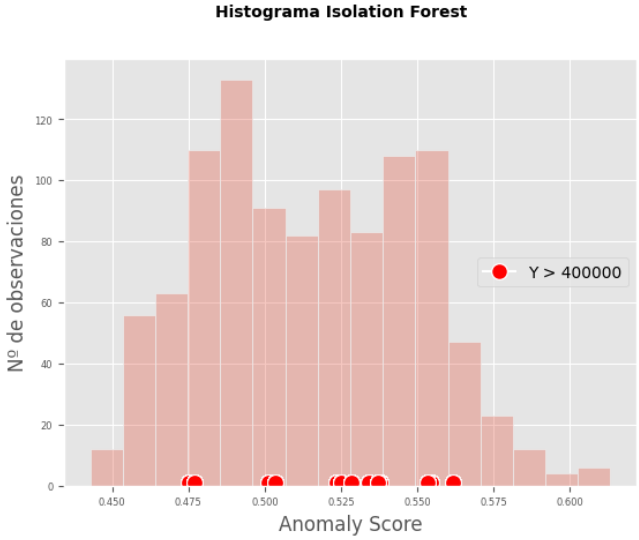


Figura 3.5. Anomaly Score de la muestra.

para la detección de observaciones anómalas. Sin embargo, sí se valorará el hecho de que las observaciones alejadas en el espacio de etiquetas Y obtuvieran un anomaly score 'normal'.

3.4. Resultados

Como conclusión a lo visto en la anterior sección 3.3, las operaciones que duraron más de 400000 segundos no serán consideradas como outliers en el espacio de características que se tiene en cuenta en este trabajo, descrito en 2.2. Se desconoce cuál puede ser el origen de su larga estancia en el puerto, pero se concluye que no existe correlación entre el alejamiento en el espacio de etiquetas y el espacio de características dado.

Por esta razón, se eliminaron las 14 operaciones de la muestra tales que $Seconds > 400000$, obteniendo así una muestra de 1023 observaciones. Luego, se reentrenaron los modelos de ML supervisados aplicados anteriormente en la sección 2.7 del capítulo 2 con los mismos hiperparámetros y se agruparon los resultados en la Tabla 3.3.

En la Tabla 3.4 se comparan los errores para ambos escenarios, por simplicidad medidos en días.

En la Tabla 3.5 se recoge la mejora conseguida tras pasar del escenario 1 al 2, en cada uno de los casos. Tomada como porcentaje de error disminuido, tal que

$$\%mejora = \frac{(Error_1 - Error_2)}{Error_1} \cdot 100 \tag{3.1}$$

	kNN	**Random Forest**	**XGBoost**
RMSE, segundos	60274.4437	56274.3618	57199.0726
MAE, segundos	43675.9409	38278.8157	39133.9243
RMSE, horas	16.7429	15.6318	15.8886
MAE, horas	12.1322	10.6330	10.8705
RMSE, días	0.6976	0.6513	0.6620
MAE, días	0.5055	0.4430	0.4529

Tabla 3.3. Errores para los 3 modelos ajustados.

	kNN	**Random Forest**	**XGBoost**
RMSE, Escenario 1	1.0425	1.0329	1.0497
RMSE, Escenario 2	0.6976	0.6513	0.6620
MAE, Escenario 1	0.6133	0.5562	0.5826
MAE, Escenario 2	0.5055	0.4430	0.4529

Tabla 3.4. Escenarios 1 y 2.

siendo $Error_1$ el RMSE o MAE medido para un modelo en el escenario 1 y $Error_2$ el respectivo RMSE o MAE medido para ese mismo modelo en el escenario 2.

	kNN	**Random Forest**	**XGBoost**
RMSE	33.0839 %	36.9445 %	36.2675 %
MAE	17.5770 %	20.3524 %	22.26 %

Tabla 3.5. Porcentajes de mejora.

Conclusiones

En el capítulo 1 se introduce la teoría que fundamenta las técnicas aplicadas en los capítulos 2 y 3. Al comienzo del capítulo 2 se presenta el problema que se pretende resolver, junto con los tres modelos de Machine Learning supervisado utilizados para estimar una solución. De entre los tres modelos aplicados, destaca Random Forest con el menor error cometido. Sin embargo, se concluye la necesidad de un estudio estructural de los datos con el fin de mejorar los resultados.

En el capítulo 3, se analiza la estructura de la muestra de estudio. Para ello se aplica ML no supervisado con el fin de detectar observaciones anómalas y se concluye que no existe correlación entre las anomalías dadas para la variable de salida Y, *Seconds*, y las variables predictoras. Entonces, se determina que las observaciones más alejadas en el espacio Y no aportan información en el espacio de características X dado. Por tanto, son interpretadas como errores y se eliminan de la muestra. Con ello se consigue reducir el error para todos los modelos presentados en el capítulo 2, volviendo a destacar Random Forest como modelo más preciso.

En cuanto a la aplicación de Machine Learning para abordar un problema, en general, y como se puede observar en este trabajo, es recomendable aplicar conjuntamente modelos supervisados y no supervisados para así poder atacar el problema desde distintas perspectivas y llegar a comprenderlo de una forma más completa. Por otro lado, cabe destacar la eficiencia y rapidez computacional de los algoritmos empleados. Sin duda alguna, la Ciencia de Datos y el Machine Learning auguran un futuro prometedor.

Respecto al problema y a la solución planteada, en el mejor de los casos (pudiendo eliminar las operaciones que duraron más de 400000 segundos de la muestra y entrenando un modelo Random Forest) se espera cometer un error de, aproximadamente, 10 horas en la estimación. Además, se proponen las siguientes acciones de mejora.

Acciones de mejora

Como continuación a los escenarios planteados para este problema, se recomienda obtener más datos, por ejemplo, las operaciones que tuvieron lugar en 2023. Además, se propone la investigación para obtener nuevas variables de estudio e implementarlas a los modelos, como por ejemplo el puerto de origen de cada barco, la temperatura media del día o alguna característica relacionada con las grúas. Por otro lado, se recomienda la aplicación de otros modelos de Machine Learning, tanto supervisados como no supervisados, como por ejemplo las redes neuronales, y comparar los resultados con los modelos aplicados en este estudio.

A

Apéndice

Distancia y similaridad.

Definición 1.3.1.- Sea U un conjunto finito o infinito de elementos. Una función $d : U \times U \to \mathbb{R}$ se llama una *distancia métrica* si $\forall x, y \in U$ se tiene:

1. $d(x, y) \geq 0$
2. $d(x, y) = 0 \iff x = y$
3. $d(x, y) = d(y, x)$
4. $d(x, z) \leq d(x, y) + d(y, z), \forall z \in U$

Definición 1.3.2.- Sea U un conjunto finito o infinito de elementos. Una función $s : U \times U \to \mathbb{R}$ se llama *similaridad* si $\forall x, y \in U$ se tiene:

1. $s(x, y) \leq s_0$
2. $s(x, x) = s_0$
3. $s(x, y) = s(y, x)$

donde s_0 es un número real finito arbitrario.

Bibliografía

[1] Sancho Caparrini, F. *Fundamentos del ML.* https://www.cs.us.es/~fsancho/Blog/posts/Fundamentos_de_ML/

[2] https://www.ibm.com/es-es/topics/data-labeling

[3] Menoyo Ros, D. García López, E. y García Cabot, A. (2021). *Fundamentos de la ciencia de datos*: (ed.) Universidad de Alcalá. https://elibro-net.accedys2.bbtk.ull.es/es/ereader/bull/177631?page=1.

[4] Underfitting y overfitting. Disponible en: https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/

[5] https://www.cs.us.es/~fsancho/Blog/posts/Aprendizaje_Supervisado_No_Supervisado.md

[6] Cunningham, Pádraig and Delany, Sarah Jane(2021). k-Nearest Neighbour Classifiers - A Tutorial. *ACM Comput. Surv.*, 54(6), 1-25

[7] https://www.ugr.es/~gallardo/pdf/cluster-2.pdf

[8] Díaz Sepúlveda, J. F., y Correa Morales, J. C. (2013).Comparación entre árboles de regresión CART y regresión lineal. *Comunicaciones En Estadística*, 6(2), 175-95

[9] Reyes Dorta, N. (2023). *Técnicas analíticas para detectar problemas de ciberseguridad* [Trabajo Fin de Grado, Universidad de La Laguna]. RIULL. https://riull.ull.es/xmlui/bitstream/handle/915/33778/Tecnicas%20analiticas%20para%20detectar%20los%20problemas%20de%20ciberseguridad.pdf?sequence=1&isAllowed=y

[10] Amat Rodrigo, J. *Árboles de decisión con Python: regresión y clasificación* https://cienciadedatos.net/documentos/py07_arboles_decision_python

[11] Medina-Merino, R. F., y Ñique-Chacón, C. I. (2017). Bosques aleatorios como extensión de los árboles de clasificación con los programas R y Python. *Interfases*, (10), 165. https://revistas.ulima.edu.pe/index.php/Interfases/article/view/1775/1828

[12] Hesterberg, T. (2011). Bootstrap. *Wiley Interdisciplinary Reviews. Computational Statistics*, 3(6), 497-526. https://

wires-onlinelibrary-wiley-com.accedys2.bbtk.ull.es/doi/full/10.1002/wics.182

[13] Amat Rodrigo, J. *Random Forest con Python* https://cienciadedatos.net/documentos/py08_random_forest_python

[14] Zhang, J., Sun, Y., Shang, L., Feng, Q., Gong, L., y Wu, K. (2020). A unified intelligent model for estimating the (gas + n-alkane) interfacial tension based on the eXtreme gradient boosting (XGBoost) trees. *Fuel (Guildford)*, 282, 118783. https://www-sciencedirect-com.accedys2.bbtk.ull.es/science/article/pii/S0016236120317798?via%3Dihub

[15] Chen, T., y Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 13-17, 785-794. https://dl-acm-org.accedys2.bbtk.ull.es/doi/abs/10.1145/2939672.2939785

[16] Zhang, J., Feng, Q., Zhang, X., Shu, C., Wang, S., y Wu, K. (2020). A Supervised Learning Approach for Accurate Modeling of CO2-Brine Interfacial Tension with Application in Identifying the Optimum Sequestration Depth in Saline Aquifers. *Energy and Fuels*, 34(6), 7353-7362. https://pubs-acs-org.accedys2.bbtk.ull.es/doi/full/10.1021/acs.energyfuels.0c00846

[17] https://www.nature.com/articles/nature14539#Sec8

[18] Suárez Rancel, M.M. (2023). Apuntes de la asignatura Análisis Multivariante.

[19] Boukerche, A., Zheng, L., y Alfandi, O. (2020). Outlier Detection: Methods, Models, and Classification. *ACM Computing Surveys*, 53(3), 1-37. https://dl-acm-org.accedys2.bbtk.ull.es/doi/abs/10.1145/3381028

[20] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, y Jörg Sander. (2000). LOF: Identifying density-based local outliers. *ACM Sigmod Record*, 29, 93-104. https://www.dbs.ifi.lmu.de/Publikationen/Papers/LOF.pdf

[21] Kong, L., Qi, C., y Qi, H.-D. (2019). Classical Multidimensional Scaling: A Subspace Perspective, Over-Denoising, and Outlier Detection. *IEEE Transactions on Signal Processing*, 67(14), 3842-3857. https://ieeexplore.ieee.org/document/8734699

[22] J. B. Kruskal. (1964). Nonmetric multidimensional scaling: A numerical method. *Psychometrika*, 29, 115-129. http://cda.psych.uiuc.edu/psychometrika_highly_cited_articles/kruskal_1964b.pdf

[23] Palenzuela Rodríguez, J.A. (2023). *Problemas de Programación Convexa* [Trabajo Fin de Grado, Universidad de La Laguna].

[24] Xu, H., Pang, G., Wang, Y., y Wang, Y. (2023). Deep Isolation Forest for Anomaly Detection. *IEEE Transactions on Knowledge and Data Enginee-*

ring, 35(12), 1-14. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10108034

[25] Liu, F. T., Kai Ming Ting, y Zhi-Hua Zhou. (2008). Isolation Forest. *2008 Eighth IEEE International Conference on Data Mining*, 413-422. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4781136

[26] https://donghwa-kim.github.io/iforest.html

[27] Amat Rodrigo, J. *Machine learning con Python y Scikit-learn* https://cienciadedatos.net/documentos/py06_machine_learning_python_scikitlearn

[28] Yu, L., Zhou, R., Chen, R., y Lai, K. K. (2022). Missing Data Preprocessing in Credit Classification: One-Hot Encoding or Imputation? *Emerging Markets Finance and Trade*, 58(2), 472-482. https://web-p-ebscohost-com.accedys2.bbtk.ull.es/ehost/pdfviewer/pdfviewer?vid=0&sid=cdf4d57b-7fc7-4b98-9398-c12a21c1a6cd%40redis

[29] Cook, A. https://www.kaggle.com/code/alexisbcook/categorical-variables

[30] Guerrero, S. C. (2017). Una metodología para el tratamiento de la multicolinealidad a través del escalamiento multidimensional. *Ciencia En Desarrollo*, 8(2), 9-24. https://web-p-ebscohost-com.accedys2.bbtk.ull.es/ehost/pdfviewer/pdfviewer?vid=0&sid=c84fa9a7-ca86-4d77-88ec-0c1e662a5697%40redis

[31] J. C. Gower. (1966). Some distance properties of latent root and vector methods used in multivariate analysis. *Biometrika*, 53, 325-338

[32] Kumaravel, V. P., Buiatti, M., Parise, E., y Farella, E. (2022). Adaptable and Robust EEG Bad Channel Detection Using Local Outlier Factor (LOF). *Sensors (Basel, Switzerland)*, 22(19), 7314. https://www.mdpi.com/1424-8220/22/19/7314

[33] Zhu, Q., Feng, J., y Huang, J. (2016). Natural neighbor: A self-adaptive neighborhood method without parameter K. *Pattern Recognition Letters*, 80, 30-36. https://www-sciencedirect-com.accedys2.bbtk.ull.es/science/article/pii/S016786551630085X?via%3Dihub#sec0017

[34] https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.LocalOutlierFactor.html

[35] Python. https://www.python.org/.

Machine Learning in a Regression Problem

Roberto Calvo Cubas
Facultad de Ciencias • Sección de Matemáticas
Universidad de La Laguna
alu0101031989@ull.edu.es

Abstract

Nowadays, an inmense amount of data is generated. This data can be structured in such a way that it can be extracted valuable information and facilitate real-world problem solving.
In this work, the aim is to introduce the reader to the field of Machine Learning. To this end, it is proposed to estimate a solution for a real-world problem in a given port of Spain.

1. Introduction

THE aim is to obtain a calculation of ETC (Estimated Time of Completed) for Valencia's Port with sufficient reliability to make easier port traffic organization.
To meet the goal, supervised and unsupervised Machine Learning will be applied in a sample with 1037 observations. In addition, let the dataset be $D = \{(x_i, y_i) \in (\mathbb{R}^m \times \mathbb{R}) / i = 1, ..., n\}$ $(n, m \in \mathbb{N})$, validation metrics used were

- **RMSE** (*Root Mean Squared Error*):

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - h(x_i))^2}$$

- **MAE** (*Mean Absolute Error*):

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - h(x_i)|$$

where $h(x_i)$ is the estimated value for y_i.

2. Supervised Machine Learning

THE following supervised Machine Learning models were applied:

- k Nearest Neighbor (kNN)
- Random Forest
- XGBoost

Once the models were trained with the sample, taking into account hyperparameters tuning, the following errors were measured

	kNN	Random Forest	XGBoost
RMSE, days	1.0425	1.0329	1.0497
MAE, days	0.6133	0.5562	0.5826

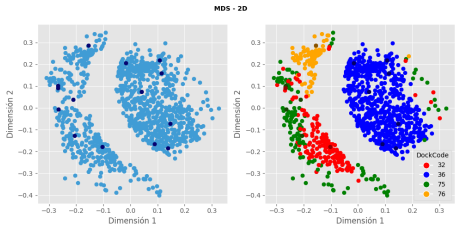
Measured Errors

Where Random Forest stands out as the most accurate model.

3. Unsupervised Machine Learning

THE following unsupervised Machine Learning models were applied:

- Multidimensional Scaling (MDS)
- Local Outlier Factor (LOF)
- Isolation Forest



By analysing the structure of the sample, it was possible to reduce the comitted error in the previous supervised ML models. Achieving the following improvements

	kNN	Random Forest	XGBoost
RMSE	+33.0839%	+36.9445%	+36.2675%
MAE	+17.5770%	+20.3524%	+22.26%

Improvement Percentages.

References

[1] Menoyo Ros, D. García López, E. y García Cabot, A. (2021). *Fundamentos de la ciencia de datos*: (ed.) Universidad de Alcalá.

[2] Kong, L., Qi, C., y Qi, H.-D. (2019). Classical Multidimensional Scaling: A Subspace Perspective, Over-Denoising, and Outlier Detection. *IEEE Transactions on Signal Processing*, 67(14), 3842-3857.

[3] Boukerche, A., Zheng, L., y Alfandi, O. (2020). Outlier Detection: Methods, Models, and Classification. *ACM Computing Surveys*, 53(3), 1-37.

[4] Amat Rodrigo, J. *Machine learning con Python y Scikit-learn*.