**Máster en Big Data y Data Science**

*Análisis para el mantenimiento preventivo en autobuses*

Trabajo Fin de Máster

Alumno: Gómez Ramírez, Daniel

Dirección: Valencia

Tutor Trabajo Fin de Máster: Peralta Martín-Palomino, Arturo

Edición octubre 2021 a marzo 2022

Índice

[Resumen 6](#_Toc97810112)

[1. Introducción 8](#_Toc97810113)

[2. Objetivos 10](#_Toc97810114)

[3. Marco teórico 11](#_Toc97810115)

[3.1. Jupyter notebook 11](#_Toc97810116)

[3.2. Python 11](#_Toc97810117)

[3.3. Pandas 12](#_Toc97810118)

[3.4. Numpy 12](#_Toc97810119)

[3.5. Matplotlib 12](#_Toc97810120)

[3.6. Seaborn 13](#_Toc97810121)

[3.7. Feature Selector 13](#_Toc97810122)

[3.8. Scikit learn 13](#_Toc97810123)

[4. Desarrollo del proyecto y resultados 14](#_Toc97810124)

[4.1. Metodología 14](#_Toc97810125)

[4.1.1. Carga de los datos 14](#_Toc97810126)

[4.1.2. Descripción del conjunto de Datos 14](#_Toc97810127)

[4.1.3. Detalles del conjunto de datos: 14](#_Toc97810128)

[4.1.4. Creación la de variable objetivo 15](#_Toc97810129)

[4.1.5. Estudio mediante estadística descriptiva 16](#_Toc97810130)

[4.1.6. Creación de variables 17](#_Toc97810131)

[4.1.7. Limpieza de los datos 18](#_Toc97810132)

[4.1.8. Subconjunto de datos para subárea motor 23](#_Toc97810133)

[4.1.9. Codificación y estandarización. 25](#_Toc97810134)

[4.1.10. Tablas de interpretabilidad 26](#_Toc97810135)

[4.1.11. Análisis de las correlaciones 27](#_Toc97810136)

[4.1.12. Herramientas de Feature Selector. 27](#_Toc97810137)

[4.1.13. PCA para características. 34](#_Toc97810138)

[4.1.14. PCA para reducir la dimensionalidad. 35](#_Toc97810139)

[4.1.15. Clusterización mediante Kmeans 35](#_Toc97810140)

[4.1.16. Selección del modelo y balanceo 35](#_Toc97810141)

[4.2. Resultados 36](#_Toc97810142)

[5. Cuadros de mando 39](#_Toc97810143)

[6. Conclusiones 40](#_Toc97810144)

[7. Referencias 41](#_Toc97810145)

Índice de figuras

[Figura 1. Logo Jupyter Notebook. Fuente: https://jupyter.org 10](#_Toc97809924)

[Figura 2. Logo Python. Fuente: https://www.python.org 10](#_Toc97809925)

[Figura 3. Logo Pandas. Fuente: https://pandas.pydata.org 11](#_Toc97809926)

[Figura 4, Logo Numpy. Fuente: https://numpy.org 11](#_Toc97809927)

[Figura 5. Logo Matplotlib. Fuente: https://matplotlib.org 11](#_Toc97809928)

[Figura 6. Logo Seaborn. Fuente: https://seaborn.pydata.org 12](#_Toc97809929)

[Figura 7. Logo Scikit learn. Fuente: https://scikit-learn.org 12](#_Toc97809930)

[Figura 8. Observaciones en función la variable objetivo. Figura propia 15](#_Toc97809931)

[Figura 9. Boxplot caracteristica días\_ant 18](#_Toc97809932)

[Figura 10. Boxplot variables categóricas. Figura propia 19](#_Toc97809933)

[Figura 11. Boxplot variables numéricas estandarizadas. Figura propia 19](#_Toc97809934)

[Figura 12.Representación del número de observaciones por subárea. Figura Propia 22](#_Toc97809935)

[Figura 13 26](#_Toc97809936)

[Figura 14 27](#_Toc97809937)

[Figura 15 28](#_Toc97809938)

[Figura 16 30](#_Toc97809939)

[Figura 17 32](#_Toc97809940)

[Figura 18 33](#_Toc97809941)

[Figura 19. Informe de clasificación RF sin balancear 34](#_Toc97809942)

[Figura 20. Matriz de confusión RF sin balancear 35](#_Toc97809943)

Índice de tablas

[Tabla 1. variable objetivo y codigo correspondiente. Tabla propiaç 13](#_Toc97725084)

[Tabla 2.Porcentaje de valores nulos por característica. Tabla propia 16](#_Toc97725085)

[Tabla 3. Número de observaciones por subárea. Tabla propia 20](#_Toc97725086)

[Tabla 4 22](#_Toc97725087)

[Tabla 5 23](#_Toc97725088)

[Tabla 6 27](#_Toc97725089)

[Tabla 7 29](#_Toc97725090)

# Resumen

En el presente TFM, se plantea el análisis de los datos de la empresa Softour Sistems para obtener la predicción de cuando un autobús va a taller cuando el motivo sea una avería en el motor

Previo al análisis se agruparon las operaciones en subáreas, creando una variable nueva la cual contine entre otras la subárea denominada motor, la cual contiene todas las averías relacionadas con motor, y que usaremos para crear un subconjunto de datos que será el que analizaremos. Este paso se realizó por parte de la empresa, previo al inicio del TFM mediante MySQL, en la extracción de datos de su datawarehouse.

Se decidió plantear un problema categórico en vez un problema de regresión, creando 3 grupos con periodos de tiempo, estos periodos de tiempo fueron decididos con ayuda de criterio experto, siendo el primero de 0 a 2 meses, el segundo de 2 a 6 meses y el tercero mayor de 6 meses, debido a que los autobuses pasan la ITV cada 6 meses.

Se utilizó Jupyter Notebook con Python como lenguaje de programación para el procesado de los datos y obtención de modelos.

El proceso constó de los siguientes pasos:

Carga de los datos

Creación la de variable objetivo, que contine los grupos previamente mencionados.

Estudio de los datos mediante estadística descriptiva, comprobando la consistencia de los datos.

Creación de variable temporal consistente con los datos, y eliminando variable inconsistente

Creación de variables que aportan valor, añadiendo a cada observación un parámetro de una observación previa, lo que nos da información de una acción previa, al suceso de la variable objetivo

Limpieza de los datos, cambio tipos de datos, eliminando columnas que no aportan valor y eliminación de valores nulos-

Generar el subconjunto de datos que contenga solo la subárea motor, objeto del estudio.

Codificación de los datos categóricos y estandarización de los datos numéricos, para poder aplicar modelos. Eliminando el hecho de que el modelo de más importancia a valores altos.

Estudio de si outlayers pueden ser eliminados o aportan información.

Creación de tablas para mantener la interpretabilidad entre el conjunto de datos codificado y el conjunto de datos no codificado.

Análisis de las correlaciones, búsqueda de variables que aporten información similar por estar fuertemente relacionadas entre sí, y búsqueda de características que nos aporten más información par predecir por estar correlacionadas con la variable objetivo.

Aplicación de las herramientas de la librería Feature Selector, nos ayudan a seleccionar las variables más relevantes, eliminando las que no son tanto.

Análisis de componentes principales (PCA) sobre el conjunto de datos traspuesto, para el estudio de la distribución de cada característica por la varianza que explican.

Análisis de componentes principales (PCA) sobre el conjunto de datos, para valorar si conviene reducir la dimensionalidad.

Clusterización mediante Kmeans, evaluado mediante el valor de silueta.

Selección del modelo de clasificación entre diversos modelos, Arboles de decisión, Random Forest, Adaboost, GradienBoosting, con gran interpretabilidad

Testeo de los modelos con diferentes métodos de balanceo, penalización para compensar, implementados en el propio modelo, oversampling de las clases minoritarias y mediante una combinación de oversample y subsample de los datos.

Selección del Modelo Random Forest, y método de balanceo con mejor resultado para la empresa, priorizando la seguridad de los autobuses.

Creación de cuadros de mando para la ayuda en la toma de decisiones mediante Business Intelligence con Power BI

# Introducción

En el presente TFM se ha analizado un conjunto de datos, suministrados por la empresa Softour Sistemas. Su problema actual es que se necesita reducir gastos, y una línea en la que están trabajando y en la cual podrían ahorrar costes es en la de las reparaciones de los autobuses. Actualmente disponen de mantenimiento correctivo y de mantenimiento preventivo, y están interesados en incluir mantenimiento predictivo, utilizando los datos históricos de los que disponen.

Lo que supondría que en vez de arreglarlos cuando se estropean, se revisará el autobús previo a sufrir una avería y no tendrían que ir, con tanta frecuencia, a taller de forma irremediable por una avería, lo que podría afectar a otras piezas, las cuales no habrían sufrido deterioro si se hubiera hecho un mantenimiento antes de averiarse, con el consecuente ahorro en coste de piezas y tiempo de reparación y reducción de tiempo de inmovilización del vehículo, lo que aumenta las horas de producción.

Para lograrlo se ha implementado un modelo que prediga en función del conjunto de datos facilitado, cuanto tiempo tardará un autobús en averiarse, en concreto cuando el motor sea el motivo.

Actualmente disponen de un conjunto de datos sobre diferentes reparaciones en piezas de autobuses. Dependiendo de la pieza del autobús, las reparaciones se efectúan al cabo periodo de tiempo. En un principio se contempló buscar la solución planteando de un problema de regresión, intentando predecir el número de días en el cual un autobús iba a volver a taller. Pero debido a que los datos de los que se dispone tienen pocas observaciones y uno de los dispositivos no había registrado correctamente las características medibles, no se obtuvo unos buenos resultados, ya que el R2 de los modelos que se probaron, rondaban en 0.2, con lo que se decidió plantearlo como un problema de clasificación.

Se puede consultar el código de las distintas pruebas en el anexo el problema de regresión: [*enlace*](https://github.com/SoftDani/Notebooks/blob/main/TFM/TFM_560_480_v5.ipynb)

*https://github.com/SoftDani/Notebooks/blob/main/TFM/TFM\_560\_480\_v5.ipynb*

Para poder predecir, si la avería es por el motor, para poder tener una cantidad de datos razonables, por encima de 1000 observaciones, se han de reducir las características, en concreto, las cuales, uno de los dispositivos no ha registrado datos.

Con ayuda de criterio experto se han creado unos rangos de tiempos con los cuales se puede obtener información que puede aportar valor, creando 3 grupos en los cuales determinan el tiempo desde que se produjo una avería en cualquier otra subárea, ‘de 0 a 2 meses’, de 2 a 6 meses’, ‘más de 6 meses’ ya que, a los 6 meses un autobús ha de pasar la ITV.

Para el procesado de datos y creación de los modelos se ha utilizado Python, que es un lenguaje de programación de alto nivel, que no es necesario compilar para ejecutar.

# Objetivos

Los objetivos principales del presente trabajo de prácticas en empresa es el predecir en cuanto tiempo un autobús necesitará ir al taller por una avería de motor, que analizaremos usando los datos extraídos de la base de datos, provista por parte de la empresa.

Los objetivos marcados para alcanzar dicho objetivo se pueden resumir en los siguientes:

1. Realizar un análisis descriptivo de los datos de los que disponemos para ganar una mayor interpretabilidad y conocimiento de ellos, para posteriormente sacar conclusiones significativas.
2. Realizar la limpieza de datos, para que podamos realizar predicciones y usar esos datos con mayor fiabilidad.
3. Encontrar las variables más importantes, que nos ayudarán a tener una mejor predicción sobre la causa de la avería.
4. Consensuar con la empresa dichas variables, para enfocarnos en las variables que más valor aportan a la empresa.
5. Analizar e investigar diferentes modelos de Machine Learning y ver con cuál de ellos obtenemos unos mejores resultados y por qué. Además, considerar la interpretabilidad de los resultados y si los resultados son entendibles a la hora de explicarlos al cliente.
6. Utilizar métodos de visualización vistas a lo largo del Máster para una mejor interpretabilidad para el lector.
7. Previsión de las piezas necesarias en los talleres, para pedirlas con el tiempo suficiente al llevar a cabo de si próximamente un autobús va a ir a taller por una avería de motor.
8. Poder ahorrar costes en piezas que no habrían sufrido deterioro al haberse hecho un mantenimiento previo a la avería.
9. Poder ahorrar costes por un menor tiempo de reparación e inmovilización de los vehículos, lo que aumenta las horas de producción

# Marco teórico

## Jupyter notebook

Jupyter notebook es una aplicación cliente-servidor que permite crear y compartir documentos. Estos documentos están compuestos por celdas, las que pueden contener entre más cosas, código, texto y fórmulas matemáticas, lo que nos permiten dichas celdas es probar bloques concretos de código de forma individual.

El programa es una aplicación web que funciona en cualquier navegador.

Para poder definir el lenguaje de programación se utilizan procesos específicos denominados kernels. que se ejecutan de forma independiente. El kernel por defecto es IPython, que permite trabajar con Python, y es el que utilizaremos para el presente estudio.

Al ser de código abierto hay nuevos kernel disponibles con frecuencia Los distintos kernels se pueden consultar en *https://github.com/jupyter/jupyter/wiki/Jupyter-kernels*

Se ha elegido esta herramienta ya que está pensado para trabajar con simulaciones numéricas y ciencia de datos, permitiendo ejecutar código, visualizar datos, realizar cálculos, visualizar resultados y documentar en un mismo entorno.

Algunos de los principales usos que se da a Jupyter Notebook:

* **Depuración de datos**: distinguir entre los datos que son importantes y los que no lo son al ejecutar un análisis de big data.
* **Modelización estadística**: método matemático para estimar la probabilidad de distribución de una característica concreta.
* **Creación y entrenamiento de modelos de aprendizaje automático**: diseño, programación y entrenamiento de modelos basados en aprendizaje automático
* **Visualización de datos**: representación gráfica de datos para visualizar con claridad patrones, tendencias, interdependencias, etc.

Podemos ver su logotipo en la Figura 1

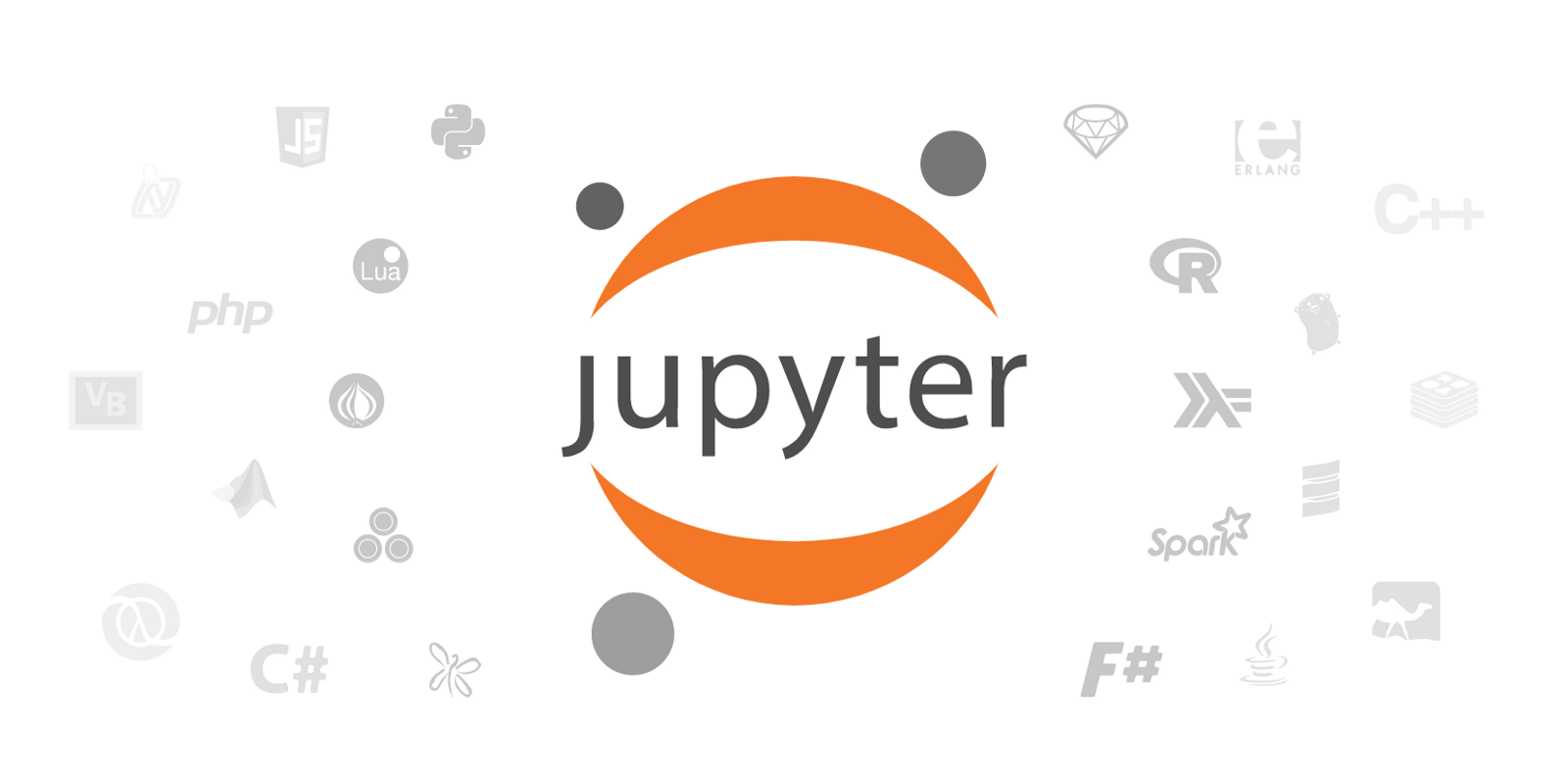


Figura 1. Logo Jupyter Notebook. Fuente: <https://jupyter.org>

## Python

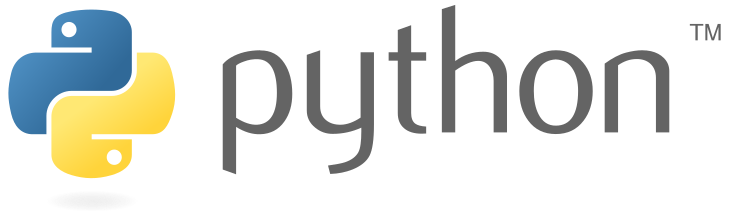


Figura 2. Logo Python. Fuente: https://www.python.org

## Pandas



Figura 3. Logo Pandas. Fuente: https://pandas.pydata.org

## Numpy

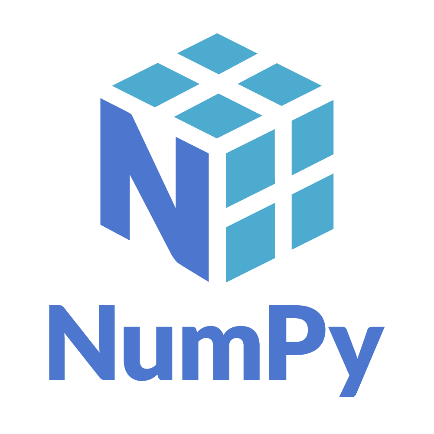


Figura 4, Logo Numpy. Fuente: https://numpy.org

## Matplotlib

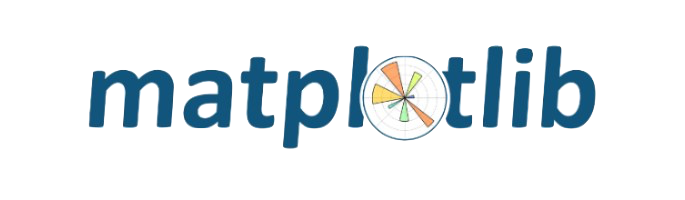


Figura 5. Logo Matplotlib. Fuente: https://matplotlib.org

## Seaborn



Figura 6. Logo Seaborn. Fuente: <https://seaborn.pydata.org>

## Feature Selector

Fuente: <https://github.com/WillKoehrsen/feature-selector>

Feature Selector es una librería de Will Koehrsen para reducir la dimensionalidad en los conjuntos de datos para machine learning.

Se ha utilizado las distintas herramientas, para seleccionar las variables más relevantes de nuestro conjunto de datos, se ha utilizado la librería feature selector, y sus distintas herramientas para la reducción de la dimensionalidad.

Dispone de 5 herramientas, de los cuales usaremos los que nos convenga.

1. Valores faltantes, que sirve para encontrar cualquier columna con una fracción faltante mayor que un umbral especificado.
2. Valores únicos con una sola observación, que sirve para encontrar cualquier característica que contenga valores únicos.
3. Características colineales, que se usa para encontrar pares de características colineales basadas en el coeficiente de correlación de Pearson.
4. Características de importancia cero, el método basa en encontrar la importancia de las características utilizando un Gradientboosting
5. Características de baja importancia, se basa en las importancias de las características del Gradientbossting,al encontrando las características de menor importancia que no se necesitan para alcanzar una importancia acumulada total especificada, normalmente un 99%

## Scikit learn

Imagen que contiene Gráfico de burbujas

Descripción generada automáticamente

Figura . Logo Scikit learn. Fuente: <https://scikit-learn.org>

## Imbalanced Learn



Figura . Logo Imbalanced learn. Fuente: https://imbalanced-learn.org

## Power BI



Figura 9. Logo Power BI. Fuente: https://powerbi.microsoft.com/es-es/

# Desarrollo del proyecto y resultados

## Metodología

### Carga de los datos

Para la carga de datos se ha utilizado pandas, que es una librería de sofware para el lenguaje de programación Phython, para la manipulación y análisis de datos, en particular tiene estructuras que permiten manipular tablas numéricas. al disponer de un archivo .csv, se ha utilizado ‘pandas.read\_csv’ que nos ha permitido cargar los datos desde una ruta interna del ordenador, aunque permite cargarlos de diferentes URLs, además para conseguir un formato adecuado se ha indicado que el separador entre características es un ‘ ; ’, ya que read\_csv también nos permite pasarle este parámetro

### Descripción del conjunto de Datos

La empresa nos ha facilitado los datos, extraídos desde una base de datos SQL, en formato .csv. Es con este .csv con el que hemos llevado a cabo nuestro análisis.

El .csv analizado consta de las siguientes variables:

- Código de autobús: se dispone de 147 autobuses diferentes.

- Las 26 subáreas que contienen las Operaciones realizadas a cada uno de los autobuses.

- Edad del autobús

- Número de días desde la anterior reparación

- Número de días desde la anterior reparación por la misma subárea

- Capacidad de cada autobús

- Marca de cada autobús: se dispone de 10 marcas diferentes

- Modelo de cada autobús

- Potencia de cada autobús

- Tipo de autobús: Interurbano, discrecional, urbano

- Subtipo de cada uno de los autobuses: normal, microbús, midibus

### Detalles del conjunto de datos:

**ide\_ope**: id operación

**nomopera**: nombre de la operación que se realiza en la reparación  
**aliasemp**: nombre del empleado que hizo la reparación  
**v\_codibud**: Código del autobús  
**v\_edad**: Edad del autobús  
**ndias\_ant**: día desde la anterior reparación  
**capacidad**: capacidad del vehículo  
**marca**: Marca del vehículo  
**modelo**: Modelo del vehículo  
**potencia**: potencia del vehículo

**longitud**: longitud del vehículo

**tara**: tara del vehículo

**cilindrada**: cilindrada del vehículo  
**tipo**: depende del uso y el modelo del vehículo  
**subtipo**: depende de la longitud y la capacidad del vehículo  
**dispositivo**: dispositivo recopilación datos telemetría

**subarea**: agrupación de operaciones relacionadas por el área de reparación

**fechaope**: fecha de la operación

**fecha\_ent**: fecha entrada en taller

**ndias\_ant**: número de días desde la anterior averiade cualquier tipo

**ndias\_ant\_tipo**: Número de días desde la anterior avería del mismo tipo

**ndias\_ant subarea**: número de días desde la anterior avería de la misma subárea

**sum(c.`distancia`):** distancia acumulada hasta la fecha de la observación

**sum(c.`litros\_totales`):** litros acumulados hasta la fecha de la observación

**sum(c.`frenazos`):** frenazos acumulados hasta la fecha de la observación

**sum(c.`exc\_rpm`):** exceso de rpm acumulados hasta la fecha de la observación

**sum(c.`exc\_temp`):** exceso de temperatura acumulados hasta la fecha de la observación

**sum(c.`metros\_asc`):** metros ascendidos acumulados hasta la fecha de la observación

**sum(c.`metros\_desc`):** metros descendidos acumulados hasta la fecha de la observación

**sum(c.`min\_ral`):** minutos a ralentí acumulados hasta la fecha de la observación

**sum(c.`acel\_bruscas`):** aceleraciones bruscas acumuladas hasta la fecha de la observación

**sum(c.`decel\_bruscas`):** deceleraciones bruscas acumuladas hasta la fecha de la observación

**AVG(c.`inercia`):** media de inercias

**sum(c.`kickdown`)**: aceleraciones a fondo acumuladas hasta la fecha de la observación

### Creación la de variable objetivo

Para crear la variable objetivo se ha tomado la columna ‘ndias\_ant\_subarea’, que indica el número de días que han pasado desde la anterior avería con la misma subárea. Y mediante la librería Numpy que está orientada a operar con grandes conjuntos de vectores y matrices, en concreto con numpy.where que nos devuelve los elementos elegidos dependiendo de una condición y nos permite realizar una acción en función de si se cumple dicha condición o no. Se ha creado una nueva columna llamado ‘tiempos’ en la cual se ha definido que si el valor de ‘ndias\_ant\_subarea’, por cada observación, es menor o igual a 60 días se devolverá en la columna ‘tiempo’ una cadena de caracteres con el valor ’de 0 a 2 meses’ en caso contrario devolverá un 0.

Si el valor de ‘ndias\_ant\_subarea’, por cada observación, es mayor de 60 y menor o igual a 180 días se devolverá en la columna ‘tiempo’ una cadena de caracteres con el valor ’de 2 a 6 meses’ en caso contrario mantendrá el valor de la columna ‘tiempos’.

Si el valor de ‘ndias\_ant\_subarea’, por cada observación, es mayor 180 días se devolverá en la columna ‘tiempo’ una cadena de caracteres con el valor ’más de 6 meses’ en caso contrario mantendrá el valor de la columna ‘tiempos’.

De esta forma se crean la variable objetivo con 3 categorías a predecir.

Observando en Figura 2 cómo se distribuyen la variable objetivo, y vemos que el conjunto de datos está desbalanceado, lo que nos indica que debemos intentar balancear los datos, para comprobar si se obtiene mejores resultados.

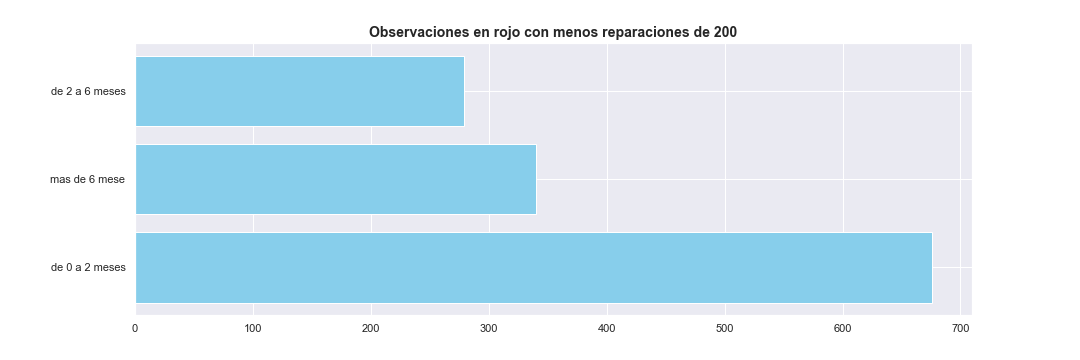


Figura 10. Observaciones en función la variable objetivo. Figura propia

### Estudio mediante estadística descriptiva

Mediante el atributo describe de pandas. Podemos obtener de nuestro conjunto de datos los datos estadísticos que resumen las tendencias, mostrando la cantidad de datos que tenemos, su media, su desviación estándar, el valor mínimo, el valor máximo, y el primer segundo y tercer cuartil.

Analizando dichos datos se observa que hay valores que no son consistentes como ndias\_ant ndias\_ant\_tipo y que generan un problema de credibilidad ya que nuestro valor mínimo es negativo -100.

Por lo que estas columnas se vas a desechar calculando mediante las fechas de entrada días\_ant, que corresponderá a la columna ndias\_ant, peor sin valores que comprometan la consistencia de los datos.

Se observa que hay muchos ceros en las variables medibles, tanto en el valor mínimo como en percentiles 25 y 50, lo que nos indica que se han registrado una gran cantidad de ceros grade, algo que no tiene lógica ya que los valores son acumulativos, por lo que, al analizar los datos, vemos que el dispositivo 480, no ha registrado correctamente datos medibles.

Por este hecho tendremos que eliminar dichas columnas, ya que el dispositivo 480 tiene el grueso de los datos, y utilizar el resto de los atributos que si tengamos bien registrados.

Vemos que del dispositivo 480 tenemos 10376 observaciones mientras que del 560 tenemos 2140 observaciones.

Se comprueban los outlayers, en el número de días anterior por subárea, ya que hay valores muy altos, pero no se pueden descartar, ya que los valores coinciden con la edad del autobús, por lo que se decide con ayuda de criterio experto, que es información importante, que debe mantenerse.

### Creación de variables

Para poder crear una nueva variable que sea consistente con los datos del número de días que han pasado desde la anterior reparación por autobús se ordenan los datos por fecha de entrada a taller y por el Código de autobús y se resetea el índice.

Para ello se usarán los atributos sort para ordenar los valores pasandole los valores de las columnas correspondientes a fecha de entrada y codigo de autobus.

Para reiniciar el índice se ha usado rest\_index .

Para poder crear la variable, necesitamos de variables de apoyo,

Creamos una nueva columna que contendrá la fecha anterior a la reparación, para ello, agrupamos el conjunto de datos por codigo de autobus y le asiganamos a la columna fecha anterior la fecha de la observación anterior mediante el atributo shift, que nos permite seleccionar valores de observaciones previas o posteriores a la observación con la que estamos trabajando para añadir un valor nuevo.

Creamos otra variable auxiliar llamada días, en la que registraremos el número de días que hay entre fecha de entrada y fecha anterior.

Creamos otra variable auxiliar de tipo bool que llamaremos ‘match’ , el objetivo de esta variables es de que si en el caso de que el autobús anterior sea el mismo que se está comprobando registraremos ‘True’ en caso contrario, si es otro autobús registraremos ‘False’, de este modo con esta variable podemos asignar los días anteriores si que afecte la fecha de otro autobús.

Para crear la variable días\_ant se crea una condición para no cometer errores si dos observaciones tienen la misma fecha de entrada, poder registrar correctamente el número de días de la entrada anterior, se comprueba que el ‘dias’ sea igual a 0 y que ‘match’ es igual a True, lo que nos indica que son el mismo autobús y 0 dias por lo que se toma el registro anterior que corresponde con el número de días correctos desde la anterior reparación, en caso contrario se deja el valor de la variable ‘dias’.

Para crear la variable de la subárea anterior afectada, para cada registro, antes de tener una reparación por motor, agrupamos por código de autobús y rellenamos la columna ‘subarea\_ant’ con el valor de la ‘subarea’ de la observación anterior.

Realizamos el mismo proceso para asignar el empleado que reparo la subárea anterior, creando de este modo, ‘aliasemp\_ant’, que nos indica el empleado que reparo la avería anterior, agrupamos por código de autobús y rellenamos la columna ‘aliasemp\_ant’ con el valor de ‘aliasemp’ de la observación anterior.

### Limpieza de los datos

#### Eliminar columnas

Se eliminan las columnas que se considera que no aportan valor en este caso:

id\_ope: ya que es el valor de identificación de cada operación

fechaope: ya que es el valor de la fecha de operación y a nosotros nos interesa el valor de la fecha de entrada a taller, ya que un autobús podría haber entrado a taller en una fecha y ser reparado más tarde por haber cola a la hora de reparar

ndias\_ant y ndias\_ant\_tipo: se eliminan por no ser consistentes y generar un problema de credibilidad.

#### Imputación y modificación de erratas en variables

En la variable ‘aliasemp’ tenemos varios problemas ya que tenemos valores vacíos, espacios en blanco y nombres mal escritos.

Para solucionarlo y no eliminar los valores nulos, se decide imputarlos y tratar los espacios en blanco como si de valores nulos se trataran, el valor de imputación para estos valores será una cadena de caracteres llamada ‘Sin asignar’, se corrigen los nombres mal escritos y se unifican en el mismo.

En la variable ‘dispositivo’, tenemos dos valores ‘560’ y ‘480 top’, por tanto, el tipo de esta variable es object, modificamos ‘480 top’ por ‘480’, ahora podemos asignarle el tipo int

#### Valores nulos

Obtenemos el porcentaje de los valores nulos,

Tabla .Porcentaje de valores nulos por característica. Tabla propia

Tabla

Descripción generada automáticamente

Como vemos que no son muchos se decide eliminar en lugar de imputarlos se utilizado el atributo ‘dropna’ de pandas para eliminar todas las observaciones que contengan valores nulos.

Quedando 10621 registros y 19 filas

Definimos el tipo de las variables modicicando:

‘dias\_ant’ como int

Capacidad como int

Potencia como int

ndias\_ant\_subarea como int

#### Outlayers

Comprobar si hay outlayers de forma visual. Pero vemos que no podemos eliminar ninguno ya que aportan información

‘días\_ant’

Tabla

Descripción generada automáticamente con confianza media

Figura . Boxplot caracteristica días\_ant

Variables categóricas se muestran en la Figura 4

Gráfico, Gráfico de cajas y bigotes

Descripción generada automáticamente

Figura . Boxplot variables categóricas. Figura propia

Variables numéricas mostradas en la Figura 5Gráfico, Gráfico de cajas y bigotes

Descripción generada automáticamente

Figura . Boxplot variables numéricas estandarizadas. Figura propia

### Subconjunto de datos para subárea motor

Tabla . Número de observaciones por subárea. Tabla propia

Tabla

Descripción generada automáticamente con confianza media

Sacamos el número de repeticiones por subárea, vemos que por subárea motor tenemos 1446 observaciones

Se puede observar también gráficamente, que no hay muchas observaciones

Gráfico, Gráfico de barras

Descripción generada automáticamente

Figura .Representación del número de observaciones por subárea. Figura Propia

Una vez creadas las variables sintéticas vamos a crear el conjunto de datos en el cual la subárea afectada sea el motor,

Para ello, se puede hacer de diversas formas, en este caso, se ha creado una variable auxiliar booleana llamada ‘target’ y se le ha asignado un 1 si la subárea es motor y un 0 si no lo es, posteriormente se ha creado un conjunto de datos con las observaciones que contenían un 1 en la variable auxiliar ‘target’

Quedando 1295 observaciones las cuales la subárea de la avería es motor.

### Codificación y estandarización.

Para poder aplicar modelos en concreto de la librería de scikit learn, diseñada para el aprendizaje automático en Python que contiene herramientas para el análisis predictivo de datos y que es de código abierto, necesitamos codificar las variables categóricas y transformarlas en numéricas. Para ello vamos a utilizar librería **category\_encoders** que contiene diversas herramientas, en este caso se ha elegido **ce.OrdinalEncoder**  que categoriza como ordinales y utilizando números enteros, pudiendo pasarle un diccionario para indicar el orden de la categoría, en este caso las clases no tiene un orden concreto por lo que los números enteros se seleccionan al azar.

https://contrib.scikit-learn.org/category\_encoders/

https://contrib.scikit-learn.org/category\_encoders/ordinal.html

Quedando:

Tabla

Pantalla de computadora

Descripción generada automáticamente con confianza baja

Para que las variables con valores más altos no tomen más importancia se decide estandarizar las variables numéricas para igualar la escala utilizando de la librería slearn.prepocessing la hermienta StandaScales eliminando ‘días\_ant’, ‘ndias\_ant\_subarea’ y ‘target’ las cuales son se estandarizaran.

https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

Tabla

Tabla

Descripción generada automáticamente con confianza media

Unimos el conjunto de datos codificado con el conjunto de datos estandarizado a través del índice.

### Tablas de interpretabilidad

Se crean las tablas para mantener la interpretabilidad, al codificar las variables categóricas, con lo que encada una de las tablas tendremos el valor original en una columna y su valor numérico codificado, correspondiente, asignado de forma aleatoria.

Tabla 6.Variable objetivo y código correspondiente. Tabla propia

Texto

Descripción generada automáticamente con confianza media

Pudiendo consultarse en los anexos:

https://github.com/SoftDani/Notebooks/blob/main/TFM/df\_codificacion.xlsx

### Análisis de las correlaciones

### Herramientas de Feature Selector.

Fuente: <https://github.com/WillKoehrsen/feature-selector>

Para seleccionar las variables más relevantes de nuestro conjunto de datos nos vamos a ayudar de una librería llamada feature selector, que es una herramienta para la reducción de la dimensionalidad.

Utilizando 5 métodos, de los cuales usaremos los que nos convenga, ya que previamente ya hemos utilizado otros métodos para obtener el mismo resultado.

1. Valores faltantes
2. Valores únicos con una sola observación
3. Características colineales
4. Características de importancia cero
5. Características de baja importancia

Para hacer uso de esta herramienta lo primero que debemos hacer es especificar que parte del conjunto de datos es la variable objetivo y cual no.

La primera herramienta, valores faltantes, encuentra cualquier columna con una fracción faltante mayor que un umbral especificado, en nuestro caso no es necesaria ya que hemos hecho previamente este proceso.

Gráfico

Descripción generada automáticamente

Figura

La segunda herramienta, Valores únicos con una sola observación, encuentra cualquier característica que contenga valores únicos, con esta herramienta vemos que no tenemos ningún valor único que solo se repita una vez, aunque sí que tenemos valores únicos,

En la siguiente grafica podemos ver la frecuencia de los valores únicos que tenemos

Gráfico, Histograma

Descripción generada automáticamente

Figura

Y el número de valores únicos que tenemos por variable.

Tabla

Descripción generada automáticamente

La siguiente herramienta, colinealidad de características, se usa para encontrar pares de características colineales basadas en el coeficiente de correlación de Pearson que mide la dependencia lineal entre dos variables, independientemente de la escala de medida de cada variable, por encima de un umbral especificado. En este caso se ha decidido que 0.8 es lo suficiente mente alto como para determinar que la información que aportan es similar.

Encontramos cuatro características fuertemente correlacionadas.

Longitud, tara, cilindrada, potencia

Podemos ver un mapa de calor de las correlaciones por encima del umbral. Las características que se eliminarán están en el eje x.

Gráfico, Gráfico de rectángulos

Descripción generada automáticamente

Figura

Podemos acceder a los detalles de las correlaciones por encima del umbral. Donde se muestra que variables la herramienta recomienda eliminar, en función de la correlación.

Tabla

*Tabla

Descripción generada automáticamente*La siguiente herramienta, características de importancia cero, se basa en un modelo de aprendizaje automático para identificar las funciones que se recomienda eliminar. Por lo tanto, requiere un problema de aprendizaje supervisado con etiquetas. El método funciona al encontrar la importancia de las características utilizando un Gradientboosting implementado en la biblioteca LightGBM. Para reducir la variación en las importancias de las características calculadas, el modelo se entrena por defecto 10 veces. El modelo también se entrena de forma predeterminada con parada anticipada utilizando un conjunto de validación (15 % de los datos de entrenamiento) para identificar el número óptimo de estimadores para entrenar.

A diferencia de los otros métodos, las características importantes de un modelo no son deterministas (tienen un poco de aleatoriedad). Los resultados de ejecutar este método pueden cambiar cada vez que se ejecuta.

En nuestro caso se ha ejecutado varias veces devolviendo que no hay ninguna característica con importancia 0.

Esta herramienta además devuelve una gráfica con importancia de las características (en una escala normalizada donde las características suman 1).

Dando como características mas importantes el número de días anterior, el empleado que reparó la anterior avería la subárea de se reparó previamente, la edad del autobús y el código de este, obteniendo con estas características un 72% de la importancia acumulada.

Gráfico

Descripción generada automáticamente

Figura

Paralelamente nos muestra la importancia acumulada de las características frente al número de características, donde nos indica que se necesita 12 características para obtener el 99% de la importancia acumulada de 15 características

Tabla

Tabla

Descripción generada automáticamente

La herramienta de baja importancia de características, se basa en las importancias de las características del Gradientbossting, al encontrar las características de menor importancia que no se necesitan para alcanzar una importancia de característica total acumulada específica en nuestro caso un 99%.

Devolviendo que las caracteristicas menso importantes son:

‘marca’,’ tipo’, ‘subtipo’ y ‘dispositivo’.

Se decide no eliminar subtipo ya que en pruebas previas se ha visto que los datos se distribuyen en clusters por subtipo, aunque los cluster tengan muy pocas observaciones para poderlos separar, y generar modelos independientes, se considera que de esta forma el modelo tendrá una escalabilidad mejor al aumentar los datos en un futuro, aunque se conoce que la importancia actual según hemos visto con feature selector, de dicha variable no tiene una alta relevancia a la hora en la que el modelo toma las decisiones actualmente.

Quedando el conjunto de datos con las siguientes características.

'v\_codigbus', 'aliasemp\_ant', 'modelo', 'subarea\_ant', 'v\_edad', 'capacidad', 'dias\_ant’,’subtipo’

### PCA para características.

Se su objetivo es reducir la dimensionalidad, a costa de la interpretabilidad, utilizando nuevas variables sintéticas no correlacionadas, para describir el conjunto de datos, determinando el orden de las nuevas variables en función de la varianza original que describen. Al emplearlo sobre la matriz transpuesta y graficando estas en función de las dos primeras variable sintéticas en el caso de que están expliquen un alto porcentaje de la varianza, podemos ver como se distribuyen las características, y si aportan información similar en función de su varianza.

El primer paso del análisis es comprobar cómo se distribuyen las variables utilizando PCA sobre la traspuesta del conjunto de datos.

Vemos que con 3 variables podemos representar el 99% de la distribución en varianza de todas las componentes.

Gráfico

Descripción generada automáticamente

Figura

Obtenemos que la varianza explicada con 3 variable es del 99%

Con una ratio de varianza [0.74216193 0.12000071 0.07364817], esto nos indica que, en la siguiente representación, tenemos representada con dos dimensiones el 86% de la varianza explicada

Imagen que contiene Gráfico

Descripción generada automáticamente

Figura

Y que las variables subtipo, v\_edad y capacidad al estar muy próximas entre sí aportarán información similar en función de su varianza.

### PCA para reducir la dimensionalidad.

### Clusterización mediante Kmeans

### Selección del modelo y balanceo

#### Selección de caracteristicas

Seleccionamos las variables que nos hacen perder menos cantidad de observaciones ya que nuestro conjunto de datos por causa motor no superará las 1400 observaciones debido a que uan hay que eliminar las observaciones con valores nulos.

Por este motivo las características seleccionadas serán:

'v\_codigbus', 'v\_edad', 'aliasemp', 'aliasemp\_ant', 'dispositivo', 'capacidad', 'marca', 'modelo', 'longitud', 'tara', 'cilindrada', 'potencia', 'tipo', 'subtipo', 'subarea', 'subarea\_ant', 'dias\_ant', 'ndias\_ant\_subarea', 'tiempos’

## Resultados

Los resultados en un principio se pretendían evaluar, con el parámetro de f1-score que combina la precisión que es la proporción de verdaderos positivos entre todos los resultados positivos, ‘VP/(VP+FP)’, con la sensibilidad, que es la proporción de casos positivos entre el total de positivos reales ‘VP/(VP+FN) ‘, en una sola métrica, ya que lo ideal es tener un valor alto de precisión y de sensibilidad, obteniendo de este modo un buen valor de f1-score.

De este modo se puede ver que la sensibilidad nos indica numéricamente cuando el modelo se decanta por una de las clases y la precisión nos indica numéricamente como acierta dentro de esa clase.

En nuestro caso particular, esto no se daba, con lo que en vez de elegir el modelo con el mejor parámetro de f1-score, se ha observado que para anteponer la seguridad no se debía seleccionar el modelo con mejor f-1 score, se ha escogido el modelo en el cual la precisión en los casos en los que mas tardaba el autobús en ir a taller ‘entre 2 y 6 meses’ y ‘ más de 6 meses’ fuera alta, aunque la sensibilidad fuera baja, y por el contrario en el caso en los que menos tardaba el autobús en ir a taller ‘entre 0 y 2 meses’ se ha priorizado una sensibilidad alta, aunque la precisión fuera baja.

Por tanto al haber realizado varios modelos, y comprobar que la precisión de los modelos con mejor f1-score, no es muy alta en las clases que más tiempo tarda el autobús en ir a taller, se ha buscado un modelo que anteponga la seguridad de los autobuses a la precisión, esto quiere decir, que se ha priorizado que prediga con mas frecuencia el rango de tiempo entre ‘entre 0 y 2 meses’ aunque sea erróneamente, ya que es mas seguro para el autobús al predecir que habrá una avería antes de lo que en realidad sucedería, ya que al contrario sería catastrófico, al averiarse antes de haber sido revisado.

El mejor resultado que hemos obtenidos con esta premisa ha sido con un modelo Random Forest, sin balancear los datos. Como se puede ver en Figura 2. Informe de clasificación RF sin balancear

Tabla

Descripción generada automáticamente

Figura . Informe de clasificación RF sin balancear

Como se puede observar en la Figura 3. Matriz de confusión RF sin balancear tenemos una sensibilidad baja para las dos primeras clases, pero una precisión relativamente alta, actualmente los mejores resultados de precisión, y para la tercera clase tenemos una sensibilidad alta, aunque una precisión algo más baja, priorizando de esta forma la seguridad, quedando la matriz de confusión de este modo, teniendo en cuenta que los valores predichos se representan en el eje de las x (abscisas) y los reales en el eje y (ordenadas):

Imagen que contiene Patrón de fondo

Descripción generada automáticamente

Figura . Matriz de confusión RF sin balancear

Se puede ver como el modelo se decanta más por la clase 3, ‘de 0 a 2 meses’ aunque falla mas ya que un 45% de los valores predichos se reparten entre la clase ‘de 2 a 6 meses’ y ‘más de 6 meses’, siendo el error mayor en la clase 2 entorno a un 28% y en la clase 1 entorno a un 17%, lo que favorece al modelo ya que la primera y la segunda clase son contiguas en lo que a tiempo se refiere.

Por otro lado, el modelo selecciona menos las clases 2,’ de 2 a 6 meses’, pero cuando los selecciona acierta más, en un 80%, en el caso de fallar lo hace a nuestro favor ya que predice más la clase 1, siendo esto positivo desde el punto de vista de la seguridad.

Como contrapunto en el caso de que el modelo prediga la clase 1, ‘más de 6 meses’ algo poco frecuente como muestra la sensibilidad, acierta en un 60% decantándose también más por la clase 2 contigua en el tiempo con ella entorno a un 26% y entorno a un 13% a la clase 3, que es la que menos antepone la seguridad.

No obtenemos los valores deseados, pero si podemos decir que el modelo antepone la seguridad de los autobuses en el caso de errar.

# Cuadros de mando

# Conclusiones

Conclusiones que se derivan de forma lógica del trabajo realizado.

En el caso de incluirse reflexiones de carácter menos objetivo, indirectas o colaterales al trabajo, deberán justificarse.

Pueden incluirse sugerencias y trabajos futuros sobre la aplicación del trabajo y sus resultados.

*¡¡¡INFORMATIVO!!! (quitar)*

# Referencias

Sí se han utilizado referencias bibliográficas externas en alguna de las etapas de desarrollo de trabajo. Por un lado, las referencias deberán ser incluidas en el texto, en un formato corto mostrando año y apellido de autor, por otro lado, se debe incluir esta misma cita en el apartado de referencias.

La normativa a seguir para hacer referencias será APA en formato de paréntesis.

https://normasapa.com/como-hacer-referencias-bibliografia-en-normas-apa/

*¡¡¡INFORMATIVO!!! (quitar)*