# Introducción

En el presente TFM se analizará 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 revisara el autobús previo a sufrir una avería y tener que ir a taller de forma irremediable, 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.

Para lograrlo se pretende implementar 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 dataset con 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 a través de un en problema de regresión, intentando predecir el número 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 con lo que se decidió plantearlo como un problema de clasificación.

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 valiosa, 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.

# Carga de 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

# 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 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 en cualquier subárea y la actual. Y mediante la librería Numpy que esta 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.

Con lo que 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.

# Limpieza

Previo a la limpieza se comprueba que tipo de dato hemos imporatado y si corresponde con lo que queremos utilizar, en el conjunto de datos debemos tener datos de los diferentes tipos:

Al tipo object en pandas, string en Python pertenecen las columnas:

**ide\_ope**: int

**nomopera**: object  
**aliasemp**: object  
**dispositivo**: int  
**v\_codibud**: object  
**v\_edad**: int  
**ndias\_ant**: int  
**capacidad**: int  
**marca**: object  
**modelo**: object  
**potencia**: float

**longitud**: float

**tara**: float

**cilindrada**: float  
**tipo**: object  
**subtipo**: object

**subarea**: object

**fechaope**: datatime

**fecha\_ent**: datatime

**ndias\_ant**: int

**ndias\_ant\_tipo**: int

**ndias\_ant subarea**: int

**sum(c.`distancia`):** float

**sum(c.`litros\_totales`):** float

**sum(c.`frenazos`):** float

**sum(c.`exc\_rpm`):** float

**sum(c.`exc\_temp`):** float

**sum(c.`metros\_asc`):** float

**sum(c.`metros\_desc`):** float

**sum(c.`min\_ral`):** float

**sum(c.`acel\_bruscas`):** float

**sum(c.`decel\_bruscas`):** float

**AVG(c.`inercia`):** float

**sum(c.`kickdown`)**: float

# Fuentes.

Páginas web:

Orientado a Python:

https://www.python.org/

https://pandas.pydata.org/

https://numpy.org/