# 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 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 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 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 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 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.

# Resumen

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.

Tabla

Descripción generada automáticamente

Como se puede observar 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

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.

# 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 importado y si corresponde con lo que queremos utilizar, en el conjunto de datos deberíamos tener datos de los diferentes tipos los cuales se detallaran a continuación:

Tipo object que corresponden a una cadena de caracteres

Tipo int que corresponde a un número entero

Tipo float, que corresponde a un número con decimales

Tipo datetime, que corresponde a una fecha

**ide\_ope**: int

**nomopera**: object  
**aliasemp**: object  
**dispositivo**: object  
**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

Se observa que no todos los datos corresponden con el tipo de dato que debería ser. Las fechas en este caso las tenemos como tipo objeto y se van a transformar a tipo datetime. También hay varias características que deberían ser de tipo int, como se ha detallado previamente, y son de tipo float.

Texto

Descripción generada automáticamente con confianza media

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 lo que tendremos que eliminar dichas columnas, ya que el dispositivo 480 tiene el grueso de los datos, y utilizar el resto de 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.

# Eliminar columnas.

S eliminan las columnas que seconsidera 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.

# Creación nuevo atributo días\_ant:

Para poder crear una nueva variable que sea consistente con los datos del número de días que han pasado desde la anterior reparación por autobus 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’.

# 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.

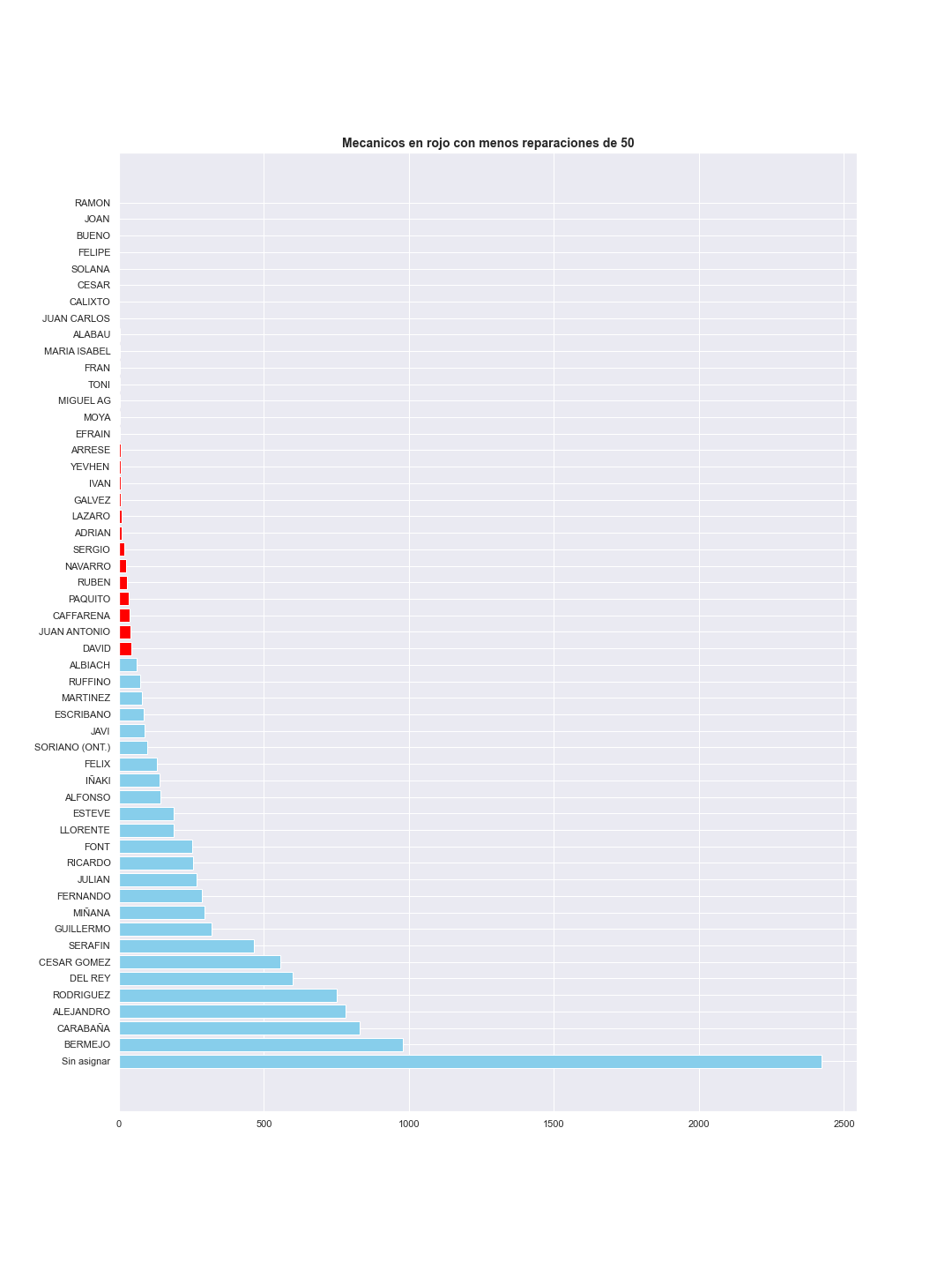
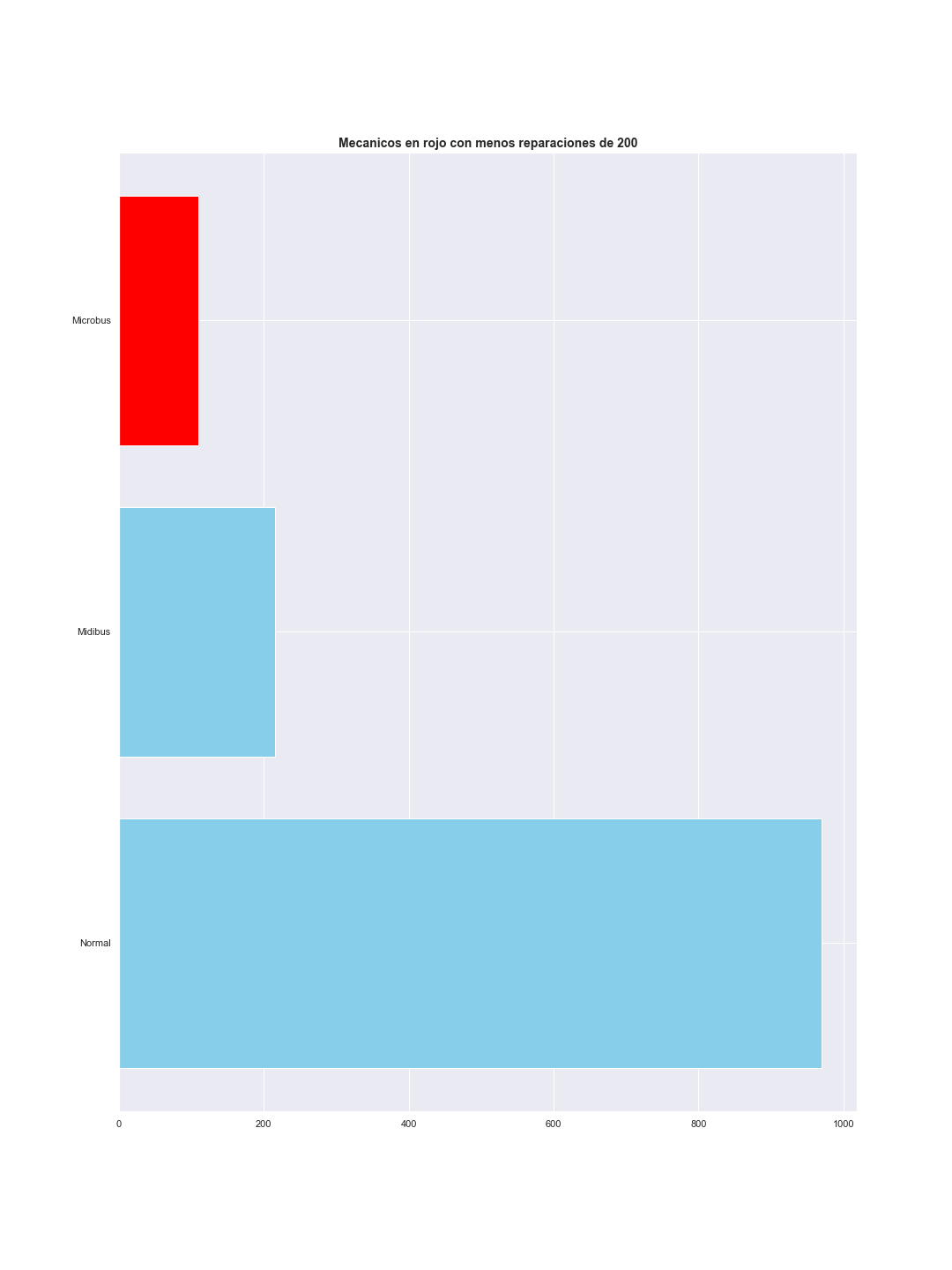
Para solucionarlo y no eliminar los valores nulos, se decide imputarlos y tratar los espacios en blanco como si de valores nulos se trataran, el valor de imputación para estos valores será una cadena de caracteres llamada ‘Sin asignar’, se corrigen los nombres mal escritos y se unifican en el mismo.

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

# Creación de variables sintéticas:

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’, agrupamos por código de autobús y rellenamos la columna ‘aliasemp\_ant’ con el valor de ‘aliasemp’ de la observación anterior.



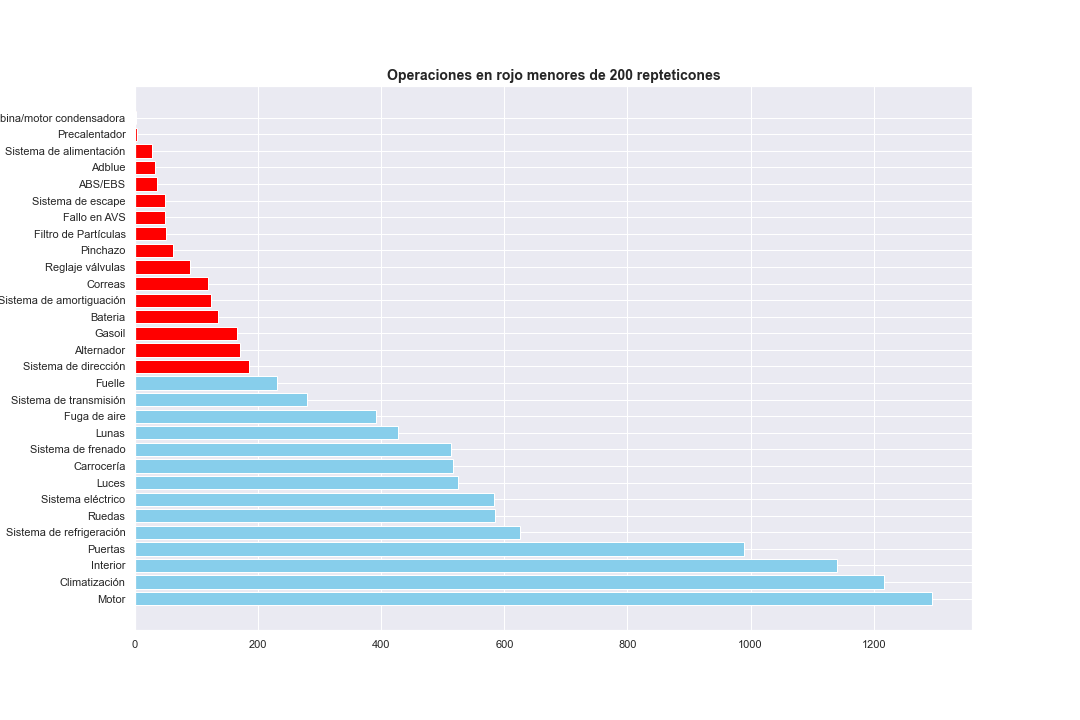
# Observaciones por avería de motor

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



# Selección de variables después del análisis del conjunto de datos

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’

# Eliminamos valores nulos

Obtenemos el porcentaje de los valores nulos,

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

# Conjunto de datos Motor.

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 nuimericas. Para ello vamos a utilizar libreria **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:

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

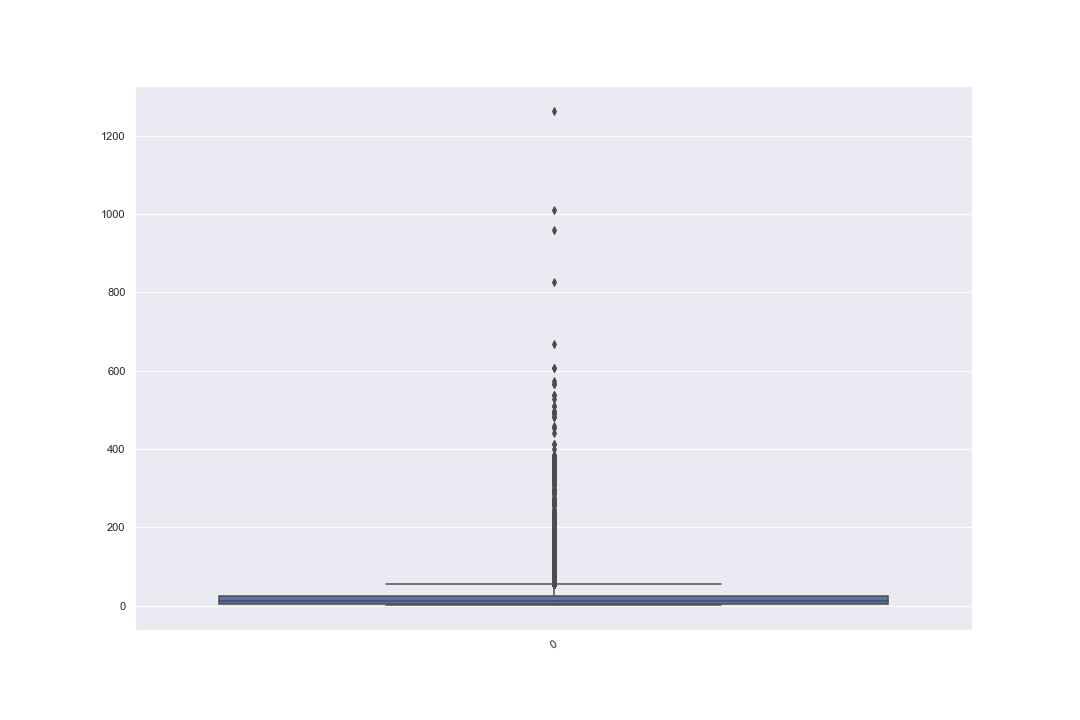
Tabla

Descripción generada automáticamente con confianza media

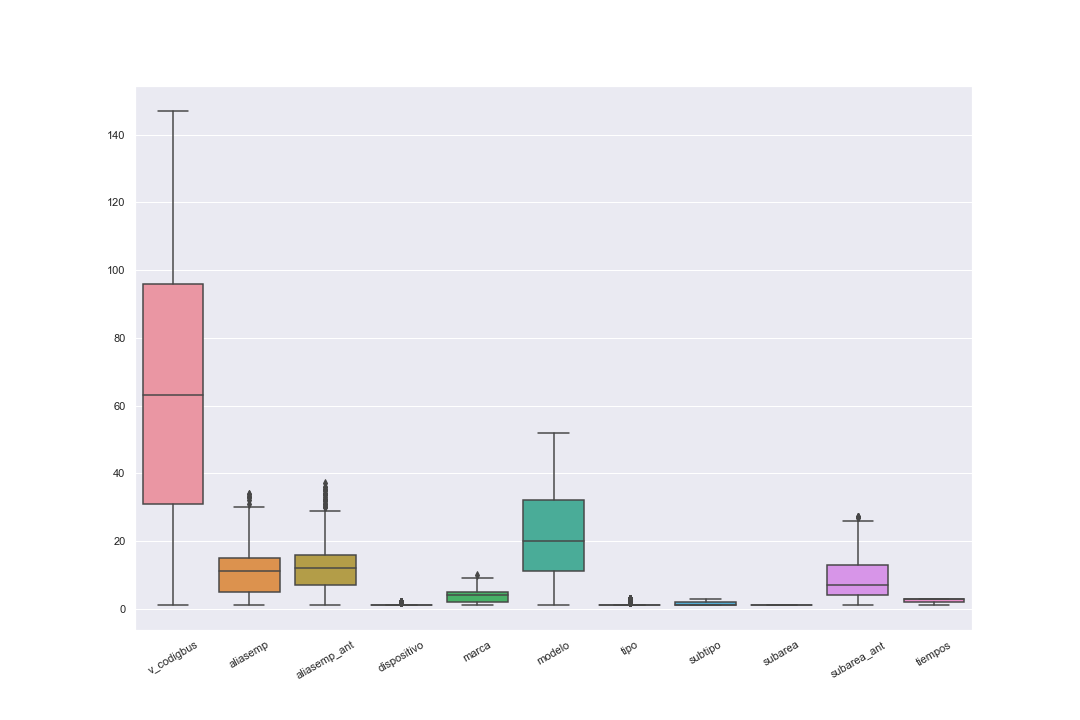
Unimos el conjunto de datos codificado con el conjunto de datos estandarizado a través del índice.

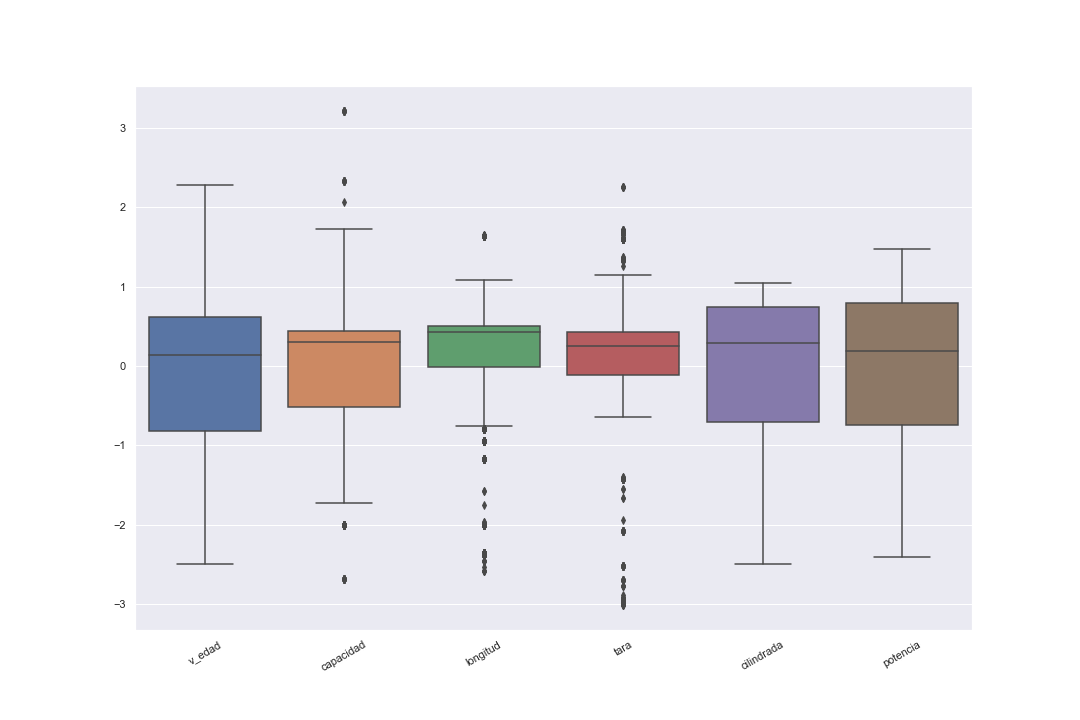
Comprobar si hay outlayers d eforma visual. Pero vemos que no podemos eliminar ninguno ya que aportan información

‘días\_ant’



Variables categóricas



Variables numericas

# Histogramas

Generamos los histogramas en las que se representan gráficamente con barras la distribución de frecuencias de cada una de las variables, de esta forma obtenemos una vista general de la distribución de la población

Imagen que contiene Diagrama

Descripción generada automáticamente

Subtipos

Imagen que contiene Texto

Descripción generada automáticamente

# 

Tiempos

No balanceado

# 

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.

Pudiendo consultarse en los anexos:

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

# Fuentes.

Páginas web:

Orientado a Python:

https://www.python.org/

https://pandas.pydata.org/

https://numpy.org/