**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](#_Toc98793363)

[1. Introducción 8](#_Toc98793364)

[2. Objetivos 10](#_Toc98793365)

[3. Marco teórico 11](#_Toc98793366)

[3.1. Jupyter notebook 11](#_Toc98793367)

[3.2. Python, 12](#_Toc98793368)

[3.3. Pandas 13](#_Toc98793369)

[3.4. Numpy 13](#_Toc98793370)

[3.5. Matplotlib 14](#_Toc98793371)

[3.6. Seaborn 14](#_Toc98793372)

[3.7. Feature Selector 15](#_Toc98793373)

[3.8. Scikit learn 15](#_Toc98793374)

[3.8.1. Modelos 16](#_Toc98793375)

[3.9. Imbalanced Learn 17](#_Toc98793376)

[3.10. Power BI 18](#_Toc98793377)

[4. Desarrollo del proyecto y resultados 18](#_Toc98793378)

[4.1. Metodología 18](#_Toc98793379)

[4.1.1. Carga de los datos 18](#_Toc98793380)

[4.1.2. Descripción del conjunto de Datos 18](#_Toc98793381)

[4.1.3. Detalles del conjunto de datos: 19](#_Toc98793382)

[4.1.4. Creación la de variable objetivo 20](#_Toc98793383)

[4.1.5. Estudio mediante estadística descriptiva 21](#_Toc98793384)

[4.1.6. Creación de variables 21](#_Toc98793385)

[4.1.7. Limpieza de los datos 22](#_Toc98793386)

[4.1.8. Subconjunto de datos para subáreas motor 26](#_Toc98793387)

[4.1.9. Codificación y estandarización. 29](#_Toc98793388)

[4.1.10. Tablas de interpretabilidad 30](#_Toc98793389)

[4.1.11. Análisis de las correlaciones 31](#_Toc98793390)

[4.1.12. Herramientas de Feature Selector. 31](#_Toc98793391)

[4.1.13. PCA para características. 39](#_Toc98793392)

[4.1.14. PCA para reducir la dimensionalidad. 40](#_Toc98793393)

[4.1.15. Clusterización mediante Kmeans 42](#_Toc98793394)

[4.1.16. Selección del modelo y balanceo 43](#_Toc98793395)

[4.2. Resultados 47](#_Toc98793396)

[5. Cuadros de mando 49](#_Toc98793397)

[6. Conclusiones 52](#_Toc98793398)

[7. Referencias 53](#_Toc98793399)

Índice de figuras

[Figura 1. Logo Jupyter Notebook. Fuente: https://jupyter.org 12](#_Toc98793154)

[Figura 2. Logo Python. Fuente: https://www.python.org 13](#_Toc98793155)

[Figura 3. Logo Pandas. Fuente: https://pandas.pydata.org 14](#_Toc98793156)

[Figura 4, Logo Numpy. Fuente: https://numpy.org 14](#_Toc98793157)

[Figura 5. Logo Matplotlib. Fuente: https://matplotlib.org 15](#_Toc98793158)

[Figura 6. Logo Seaborn. Fuente: https://seaborn.pydata.org 15](#_Toc98793159)

[Figura 7. Logo Scikit learn. Fuente: https://scikit-learn.org 16](#_Toc98793160)

[Figura 8. Logo Imbalanced learn. Fuente: https://imbalanced-learn.org 18](#_Toc98793161)

[Figura 9. Logo Power BI. Fuente: https://powerbi.microsoft.com/es-es/ 19](#_Toc98793162)

[Figura 10. Observaciones en función la variable objetivo. Figura propia 22](#_Toc98793163)

[Figura 11. Boxplot caracteristica días\_ant 25](#_Toc98793164)

[Figura 12. Boxplot variables categóricas. Figura propia 25](#_Toc98793165)

[Figura 13. Boxplot variables numéricas estandarizadas. Figura propia 26](#_Toc98793166)

[Figura 14.Representación del número de observaciones por subárea. Figura Propia 28](#_Toc98793167)

[Figura 15. Heatmap Correlaciones. Figura propia 31](#_Toc98793168)

[Figura 16. Valores faltantes. Figura propia 32](#_Toc98793169)

[Figura 17. Frecuencia valores unicos. Figura propia 33](#_Toc98793170)

[Figura 18. Correlaciones por encima del humbral 34](#_Toc98793171)

[Figura 19. Importancia de las caractristicas e importancia acumulada. Figura propia 37](#_Toc98793172)

[Figura 20. Compenentes para explicar 99% de varianza 39](#_Toc98793173)

[Figura 21. Dispersiond e las caracteristicas en funcion de la varianza. Figura propia 40](#_Toc98793174)

[Figura 22. Numero variables sintéticas que representan el 95% de la varianza. Figura propia 41](#_Toc98793175)

[Figura 23. Distribución PCA. Figura propia 41](#_Toc98793176)

[Figura 24. Valores de coeficiente de silueta por número de clusters. Figura propia 42](#_Toc98793177)

[Figura 25. Visualización de clusters Kmeans. Figura propia 42](#_Toc98793178)

[Figura 26. Importancia de las características DT. Figura propia 44](#_Toc98793179)

[Figura 27. Importancia de las características AB. Figura propia 44](#_Toc98793180)

[Figura 28. Importancia de las características RF. Figura propia 45](#_Toc98793181)

[Figura 29. Informe de clasificación RF sin balancear 47](#_Toc98793182)

[Figura 30. Matriz de confusión RF sin balancear 48](#_Toc98793183)

[Figura 31. Porcentaje de vehículos reparados por edad y marca. Figura propia 49](#_Toc98793184)

[Figura 32. Cuadro de mando recuento de varias por marca. Figura propia 50](#_Toc98793185)

[Figura 33. Cuadro de mando capacidades 51](#_Toc98793186)

[Figura 34. Cuadro de mando filtrado de características. Figura Propia 52](#_Toc98793187)

Índice de tablas

[Tabla 1.Porcentaje de valores nulos por característica. Tabla propia 23](#_Toc98793491)

[Tabla 2. Número de observaciones por subáreas. Tabla propia 27](#_Toc98793492)

[Tabla 3. Codificación variables categoricas. Tabla propia 29](#_Toc98793493)

[Tabla 4. Estadarización valores numéricos. Tabla propia 30](#_Toc98793494)

[Tabla 5.Variable objetivo y código correspondiente. Tabla propia 30](#_Toc98793495)

[Tabla 6. Valores unicos. Tabla propia 33](#_Toc98793496)

[Tabla 7. Variables para eliminar por correlación, Tabla propia 35](#_Toc98793497)

[Tabla 8. Importancia e importancia acumulada 38](#_Toc98793498)

# 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 para 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**:
* **Modelización estadística**:
* **Creación y entrenamiento de modelos de aprendizaje automático**:
* **Visualización de datos**:

Podemos ver su logotipo en la Figura 1

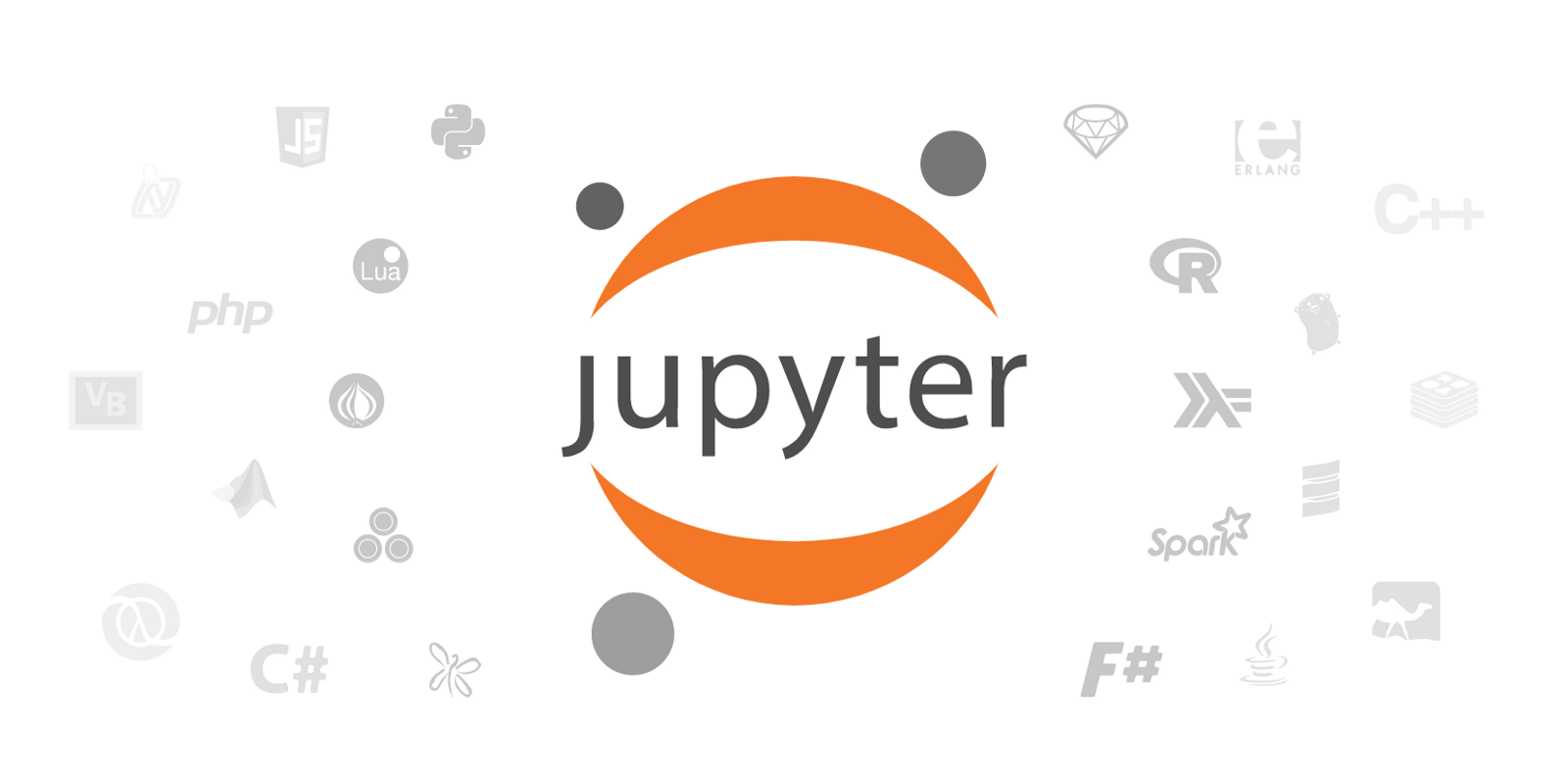


Figura 1. Logo Jupyter Notebook. Fuente: <https://jupyter.org>

## Python,

Python es un lenguaje de programación de alto nivel, simple pero rico en funciones.

Existen lenguajes de programación especializados como R, Scala, Julia, etc…, para solucionar los problemas generados por el análisis de datos o ciencia de datos,

Sin embargo, la facilidad de uso y de aprendizaje, hacen de Python un lenguaje muy poderoso, al no perder funcionalidades, y no se requerirse configuraciones complejas y ni compilaciones, además, permite diversos estilos de programación, como programación orientada a objetos o programación funcional.

Finalmente, el lenguaje tiene un código legible, que ayuda a la fácil comprensión de las bases del código lo que permite explorar los campos de big data y ciencia de datos, sin grandes conocimientos de programación.

Por otro lado, hemos comentado su riqueza en funcionalidades, podemos destacar las utilizadas para el propósito del presente TFM, enfocadas al campo de big data y ciencia de datos, tenemos paquetes estadísticos, como Pandas, Numpy, Scikit-learn, Imbalanced-learn y paquetes de visualización como Matplotlib y Seaborn, de los cuales se explicará su propósito a continuación.

Podemos ver su logotipo en la Figura 2.

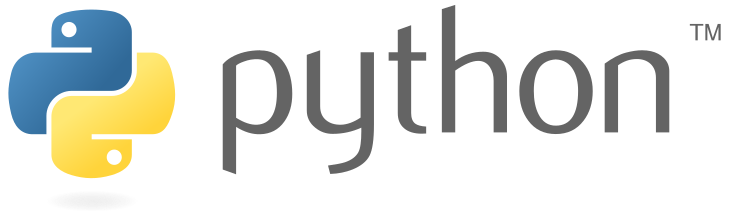


Figura 2. Logo Python. Fuente: <https://www.python.org>

## Pandas

En el mundo real, los datos están inevitablemente desordenados. Y Pandas es una biblioteca de Python para el análisis de datos, destinada a limpiar, transformar, manipular y analizar datos, de código abierto. Utiliza computación en memoria, lo que es bueno para conjuntos de datos de tamaño pequeño a mediano, lo que coincide con nuestro caso concreto. Para procesar grandes conjuntos de datos es una herramienta limitada debido a errores de falta de memoria.

Podemos ver su logotipo en la Figura 3



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

## Numpy

Se aplica a la programación científica en Python, especialmente para números, permite realizar funciones matemáticas básicas, funciones estadísticas básicas, y funciones avanzadas [en matrices multidimensionales](https://www.educba.com/multi-dimensional-array-in-javascript/), realizando cálculos pesados para extraer información útil. Las matrices de Numpy son mas rápidas que las listas de Python, aunque estas ultimas son más flexibles ya que las matrices de Numpy solo pueden almacenar un mismo tipo de dato en cada columna.

Podemos ver su logotipo en la Figura 4

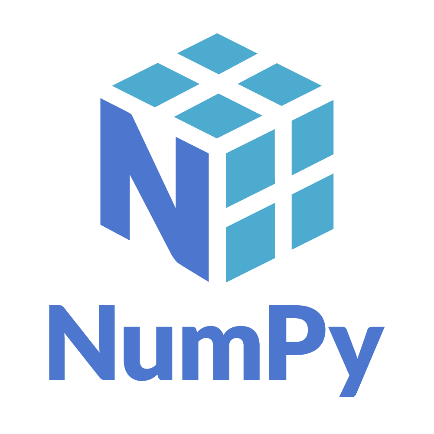


Figura 4, Logo Numpy. Fuente: https://numpy.org

## Matplotlib

Es una librería para crear visualizaciones en Python. Debido a que es más fácil obtener información de los datos representados visualmente en comparación con los datos en crudo en formato tabla, se ha utilizado esta librería para hacer más fácil la comprensión de los datos, ver anomalías en los mismos, y las tendencias de sus distribuciones,

Podemos ver su logotipo en la Figura 5

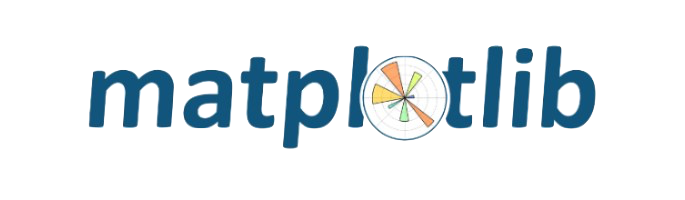


Figura 5. Logo Matplotlib. Fuente: https://matplotlib.org

## Seaborn

Es una librería basada en matplotlib, se utiliza para dibujar gráficos estadísticos atractivos e informativos, esta librería destaca por tener una buena calidad estética para poder ofrecer información significativa. Nos permite tener una interfaz de alto nivel en comparación con el bajo nivel de matplotlib. Las funciones de seeborn funcionan mejor con grandes conjuntos de datos, por lo que destaca en su uso en la ciencia de datos, generando visualizaciones atractivas.

Podemos ver su logotipo en la Figura 6



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, y de esta forman 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 se 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

Es una librería para el análisis predictivo de datos en Python. Proporciona herramientas para el aprendizaje automático, el modelado estadístico, de clasificación y regresión, clustering, reducción de dimensionalidad.

Podemos ver su logotipo en la Figura 7

Imagen que contiene Gráfico de burbujas

Descripción generada automáticamente

Figura 7. Logo Scikit learn. Fuente: <https://scikit-learn.org>

De esta libraría se han utilizado el análisis de componentes principales (PCA) con dos objetivos analizar las características en función de su varianza para mostrar que variables puedes darnos información similar y comprobar la posibilidad de reducción de características, lo cual se ha visto que no era útil debido a que se perdía la interpretabilidad a costa de reducir una solo variable.

Se ha utilizado Kmeans para crear un variable sintética en función de como se clusterizan con este método, obteniendo la mejor clusterización en función del coeficiente de silueta

### Modelos

En lo que a modelos se refiere se ha probado predicciones con los siguientes modelos

Árbol de decisión para clasificación, es una estructura de árbol similar a un diagrama de flujo, donde cada nodo interno aplica una prueba en un atributo, cada rama representa un resultado de la prueba y cada nodo hoja (nodo terminal) tiene una etiqueta de clase.

Random Forest para clasificación, es un tipo de modelo que combina diversos árboles de decisión y la salida de cada uno se contará como “un voto” y la opción más votada será́ la respuesta del Random Forest, al utilizar múltiples árboles se reduce considerablemente el riesgo de overfiting, normalmente da buenos resultados en problemas de clasificación, aunque no funciona tan bien con pocos datos.

Boosting (Adaboost para Clasificación). en este caso el algoritmo construye secuencialmente un conjunto de árboles de decisión que supone modelos débiles (weak learners). Un modelo débil es cualquier conjunto de aprendizaje que es al menos un poco mejor que la predicción aleatoria (> 50%). Posteriormente combina sus salidas de forma aditiva para obtener una predicción final. En cada iteración del algoritmo, cada modelo individual intenta corregir los errores cometidos en la iteración previa mediante la optimización de una función de pérdida (Freund, 1999)

Para encontrar los hiperparámetros se ha usado RandomizedSearchCV y GridSearchCV, que son algoritmos para muestrear de manera efectiva el espacio de búsqueda y encontrar una buena solución. Inicialmente y utilizando finalmente RandomizedSearchCV ya que con un número considerable de iteraciones se consigue valores muy similares a los de GridSearchCV que, a pesar de ser mucho más exhaustivo, el coste computacional es mucho mayor por lo que se ha utilizado en espacios de búsqueda menores, acotados mediante el RandomizedSearchCV. Además se ha utilizado cross validation que tiene dos pasos principales, dividir los datos en subconjuntos y rotar el entrenamiento y la validación entre los conjuntos, en los algoritmos anteriormente mencionas ya viene integrado, utilizando 5 particiones.

## Imbalanced Learn

Es un paquete para tratar desbalanceados en los datos. Estos desbalanceos se manifiestan generalmente el conjunto de datos tiene etiquetas de clases, y una o más de estas clases tienen muy pocas observaciones para poder aprender.

Existen tres categorías para abordar el desbalanceo sobremuestreo de la clase minoritaria, submuestreo de la clase mayoritaria y una combinación de las anteriores.

En este caso como no tenemos una cantidad muy grande datos se ha utilizado en el sobremuestreo y la combinación de sobremuestreo y submuestreo. En concreto con Randomoversampler para sobremuestrear las clases minoritaria y con Smotetomek para la combinar submuestreo y sobremuestreo.

Podemos ver su logotipo en la Figura 8



Figura 8. Logo Imbalanced learn. Fuente: https://imbalanced-learn.org

## Power BI

Es un conjunto de herramientas Business intelligence (BI) para ayudar a las empresas a la toma de decisiones basada en los datos y obtener una visión integral de los datos de las mismas, utilizando la estadística descriptiva.

para el presente TFM nos interesa el desarrollo de cuadros de mando para mostrar una serie de indicadores numéricos y gráficos objetivos que ayudaras a la toma de decisiones de los directivos

Podemos ver su logotipo en la Figura 9



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

## Estado del Arte

Se van a analizar distintas técnicas que se han utilizado para resolver problemas de clasificación similares en distintos artículos.

Los artículos que se han analizado son:

**Emma V. Barreno Vereau.(2012).** **Análisis Comparativo de modelos de clasificación en el estudio de la deserción universitaria. Lima**

https://dialnet.unirioja.es/servlet/articulo?codigo=6043123

El cual nos presenta la regresión logística que permite estimar la probabilidad de un suceso que depende de los valores de ciertas covariables como una de las opciones para clasificar, y lo va a comparar con árboles de clasificación, entre otras formas de evaluar los resultados se ha usado la matriz de confusión y la curva ROC.

Ambos métodos brindaron similares resultados en este caso, siendo el modelo de regresión logista el más fácil de implementar

**Bach. Cárdenas Garro José Antonio. (2019)** **Clasificación de aceptación de campañas para una entidad financiera, usando random forest con datos balanceados y datos no balanceados. Lima, Perú**

<http://repositorio.urp.edu.pe/handle/URP/2307>

En el cual se planteó abordar un enfoque de modelamiento de aprendizaje supervisado de clasificación mediante el modelo de Random Forest, que permitió tener una comparación de los modelos planteados dando un balanceo de la variable objetivo y sin balancear.

Los 4 modelos planteados obtuvieron similar indicador, en el indicador de especificidad, los mejores modelos fueron los que tenían datos desbalanceados en el indicador de sensibilidad fueron los que tenían los datos balanceados. Para este problema se eligió priorizar la sensibilidad usando datos balanceados.

**Barón García, Alejandro. (2020). Detección y clasificación de fallos en motores mediante procedimientos Boosting. Valladolid, España**

<https://uvadoc.uva.es/handle/10324/43777>

El cual presenta una metodología estadística basada en técnicas de Boosting para clasificar los motores en un estado de deterioro, se haga determinado que en función d ellos inversores se requieren modelos diferentes, pero que el mejor algoritmo de boosting para este problema es XGBoost, siendo más rápido LightGBM pero con peores resultado y dando los peores resultados AdaBoost en cualquier situación.

**Jose Luis Contreras Alvarez. (2020). Diseño de un modelo para mantenimiento predictivo en motores de inducción utilizando técnicas de la Industria 4.0. Lima Perú**

https://repositorio.utp.edu.pe/handle/20.500.12867/4275

El cual tiene como objetivo utilizar las técnicas predictivas para el diseño y desarrollo de un modelo de mantenimiento predictivo, para motores de inducción, se obtuvo los mejores resultados con tres algoritmos de clasificación, K vecinos más cercanos, Maquina de Vector de Soporte y Random Forest siendo este último el que obtuvo el mayor grado de predicción, para evaluar fallos cuando se realiza mantenimiento preventivo.

**Jorge Luis Borges Jiménez. (2015).** **Métodos avanzados de preprocesamiento de datos para maximizar el problema de clases desbalanceadas en la clasificación multinstancia. Santa Clara**

https://dspace.uclv.edu.cu/handle/123456789/7624

En el cual se implementa un método de preprocesamiento de datos combina sobremuestreo de la clase minoritaria y submuestreo de la clase mayoritaria en la clasificación multinstancia. La evidencia experimental mostró que hay mejoras en la calidad de la clasificación. Se modificó un algoritmo de preprocesamiento en concreto MISMOTE y se obtuvieron mejoras en el proceso de clasificación en algunos conjuntos de datos.

# Desarrollo del proyecto y resultados

## Metodología

### Carga de los datos

Para la carga de datos se ha utilizado pandas, que es una librería de software para el lenguaje de programación Python, 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 caracteristicas importantes:

- 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

- Dispositivo, hay dos tipos dispositivos de adquisición de datos de telemetría, de los cuales uno el 480 no ha registrado datos de telemetría, pero si tenemos características de los autobuses autobuses con dicho dispositivo

- Disponemos de registros de numero de días que han pasado desde una anterior reparación, ndias\_ant desde cualcier tipo de reparación, ndias\_ant\_tip desde reparaciones del mismo tipo, ndias\_ant\_subarea desde reparaciones dentro de la misma subárea, esta ultima esla que mas nos interesa ya que una de estas subáreas es Motor

- Datos de telemetría que registran el acumulado los cuales no se han podido usar por que al eliminar los missing perdíamos muchos registros ya que el grueso de los datos son del dispositivo 480

### 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 10 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.

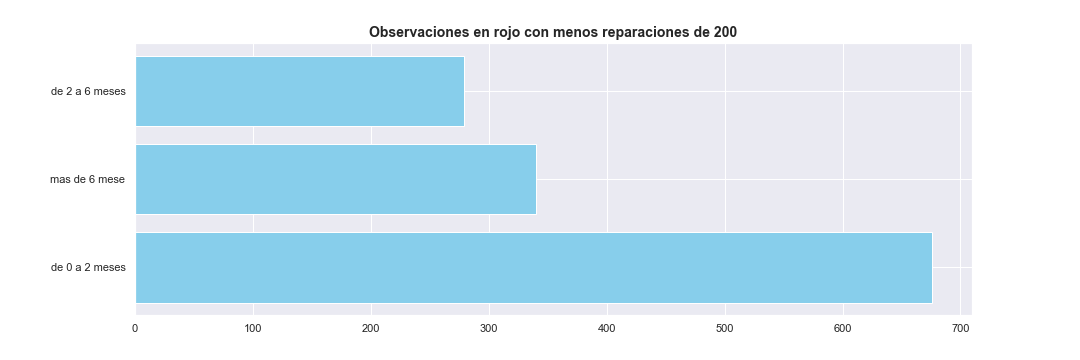


Figura 10. Observaciones en función la variable objetivo. Figura propia

### 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 usará el atributo 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 código de autobús y le asignamos 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 ‘días’ sea igual a 0 y que ‘match’ es igual a True, lo que nos indica que son el mismo autobús y 0 días 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 ‘días’.

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 ‘subáreas’ 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, mostrados en la Tabla 1

Tabla 1.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, y no tiene problemas de credibilidad ni de integridad los valores.

Podemos ver ‘días\_ant’ en la Figura 11. Boxplot caracteristica días\_ant

Tabla

Descripción generada automáticamente con confianza media

Figura 11. Boxplot caracteristica días\_ant

Variables categóricas se muestran en la Figura 12

Gráfico, Gráfico de cajas y bigotes

Descripción generada automáticamente

Figura 12. Boxplot variables categóricas. Figura propia

Variables numéricas mostradas en la Figura 13Gráfico, Gráfico de cajas y bigotes

Descripción generada automáticamente

Figura 13. Boxplot variables numéricas estandarizadas. Figura propia

### Subconjunto de datos para subáreas motor

A continuación, podemos ver,

Tabla 2. Número de observaciones por subáreas. 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 en la Figura 14

Gráfico, Gráfico de barras

Descripción generada automáticamente

Figura 14.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 como se muestra en la Tabla 3

Tabla 3. Codificación variables categoricas. Tabla propia

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. Quedando como se muestra en la Tabla 4

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

Tabla 4. Estadarización valores numéricos. Tabla propia

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 5.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

Se determina por criterio experto, que un 0,8 es una correlación alta, y que si hay una correlación entre dos variables (excluyendo la variable objetivo), de 0.8 o más, debe ser estudiadas por si hubiera que eliminar una de ellas.

Podemos determinar al ver la Figura 15 que hay correlaciones con valores altos entre características, por lo que se deduce que van a generar información similar, y además se observa que no hay una alta correlación con la variable objetivo ‘tiempos’, que sea determinante, excepto ‘dias\_ant’ que tiene un 22%, lo que indica que aportara más información que el resto a la hora de predecir la variable objetivo en función de la correlación.

Como tenemos correlaciones altas, para la selección de las variables a eliminar por su correlación, se usará la librería Feature Selector.

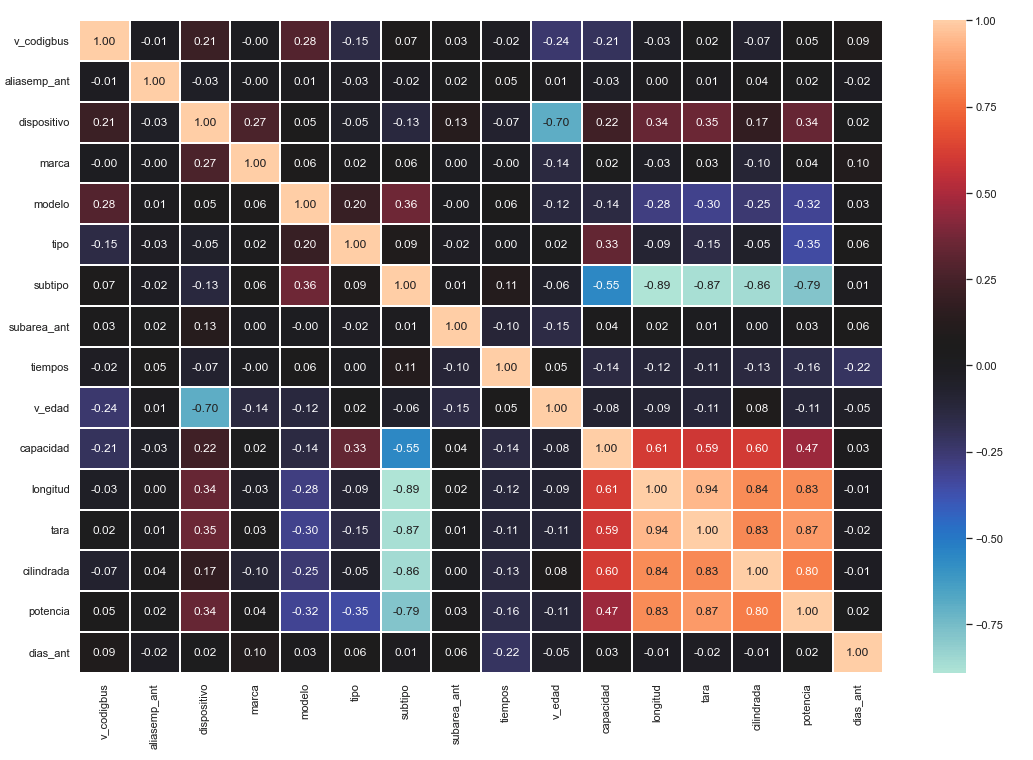


Figura 15. Heatmap Correlaciones. Figura propia

### 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 de los valores totales faltantes, mayor que un umbral especificado, en nuestro caso no es necesaria ya que hemos hecho previamente este proceso. Como podemos ver en la Figura 16 el valor de los missigns es de 0.0

Gráfico

Descripción generada automáticamente

Figura 16. Valores faltantes. Figura propia

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 Figura 17 se muestra la gráfica en la podemos ver la frecuencia de los valores únicos que tenemos

Gráfico, Histograma

Descripción generada automáticamente

Figura 17. Frecuencia valores unicos. Figura propia

En la Tabla 6 podemos ver el número de valores únicos que tenemos por variable.

Tabla 6. Valores unicos. Tabla propia

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 características fuertemente correlacionadas.

Subtipo, longitud, tara, cilindrada, potencia que coinciden con las correlaciones que vimos anteriormente la Figura 15

Podemos ver un mapa de calor de las correlaciones por encima del umbral Figura 18. Correlaciones por encima del humbral. Las características que se eliminarán están en el eje x.

Gráfico, Gráfico de rectángulos

Descripción generada automáticamente

Figura 18. Correlaciones por encima del humbral

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. Como se puede ver en la Tabla 7

Tabla 7. Variables para eliminar por correlación, Tabla propia

*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). Figura 19

Dando como características más importantes el número de días anterior, el empleado que reparó la anterior avería la subárea que 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 19. Importancia de las caractristicas e importancia acumulada. Figura propia

Paralelamente nos muestra la importancia acumulada Figura 19 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. Que podemos ver en detalle en la Tabla 8

Tabla 8. Importancia e importancia acumulada

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 características menos 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.

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 variables 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.

Vemosen la Figura 20 que con 3 variables podemos representar el 99% de la distribución en varianza de todas las componentes.

Gráfico

Descripción generada automáticamente

Figura 20. Compenentes para explicar 99% de varianza

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. Figura 21

Imagen que contiene Gráfico

Descripción generada automáticamente

Figura 21. Dispersiond e las caracteristicas en funcion de la varianza. Figura propia

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.

Como se ha explicado previamente el objetivo de la PCA 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.

Este apartado tiene como objeto comprobar si vale la pena reducir la dimensionalidad a razón de sacrificar la interpretabilidad del conjunto de datos, aplicando PCA sobre el conjunto de datos. Como podemos ver en la Figura 22, para explicar el 95% de la varianza necesitamos 6 por lo que no compensa perder la interpretabilidad de las variables para ahorrarnos dos variables.

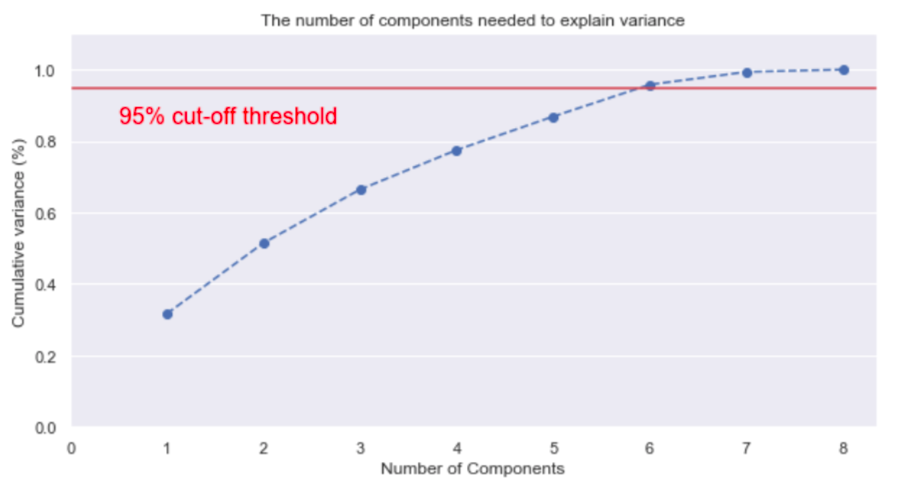


Figura 22. Numero variables sintéticas que representan el 95% de la varianza. Figura propia

Podemos ver en la Figura 23 que, del conjunto de datos, se agrupan unas partes separadas de otras, lo que indica que al menos hay varios clusters diferenciados, se a probado a nombrar cada punto con distintas columnas de características, obteniendo como resultado que la mayoría de los autobuses del subtipo 1 'Normal’ están agrupados y al igual que el subtipo 2 y 3.

Como podemos ver en la Figura 23. los autobuses se distribuyen en función de las dos dimensiones con más varianza explicada, en torno a un 56%, en conjuntos dividiéndose en subtipos, según nos muestra la primera dimensión y cada subtipo a su vez en dos grupos según nos muestra la segunda dimensión.

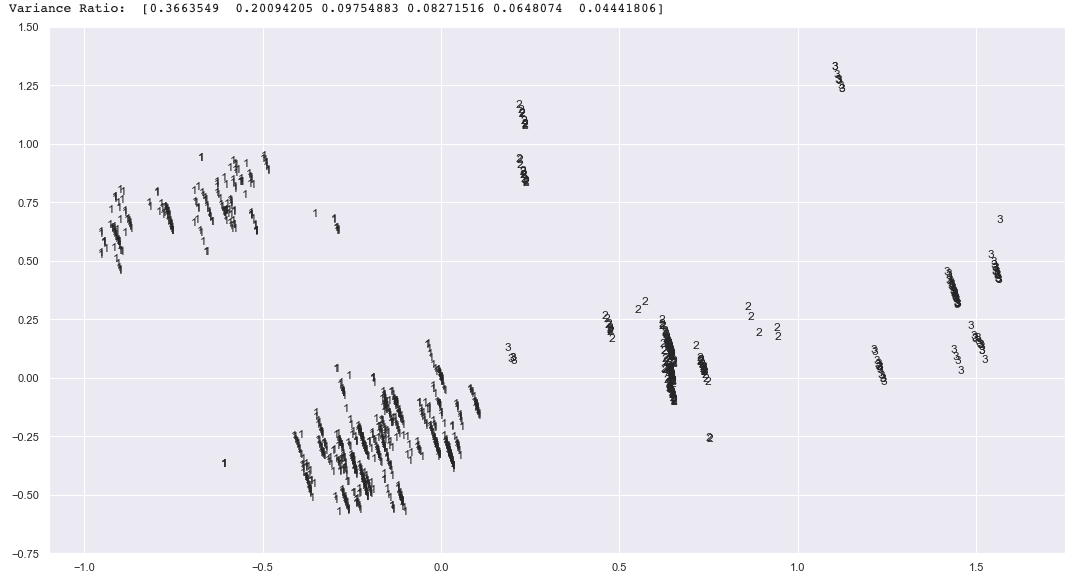


Figura 23. Distribución PCA. Figura propia

### Clusterización mediante Kmeans

La agrupación en clusters de Kmeans sirve para dividir un conjunto de datos en K clusters distintos que no se superponen. Para realizar el agrupamiento de Kmeans, se debe especificar el número deseado de agrupamientos K, luego, el algoritmo de Kmeans asignará cada observación exactamente a uno de los K conjuntos de datos.

Se prueba a implementar Kmeans con distinta cantidad de clusters, observando que con 4 clusters se obtiene el mejor valor de silueta, como se muestra en la Figura 24 y Figura 25 que es una medida para cuantificar cuan similar es un objeto a su propio grupo en comparación con otros grupos, por lo tanto, nos indica cual es la mejor opción para clusterizar, algunas clusterizaciones se pueden ver claramente de forma grafica, en las dos dimensiones representadas pero no otras, como parecía, en la distribución de la PCA, por lo que el valor de silueta es el que tomamos de referencia para determinar el numero de clusters que usáremos para crear una variable sintética que indique a que clúster pertenece cada observación, evitando de este modo crear nuevos subconjuntos de datos, ya que no tenemos una gran cantidad de valores.

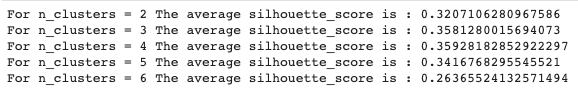


Figura 24. Valores de coeficiente de silueta por número de clusters. Figura propia

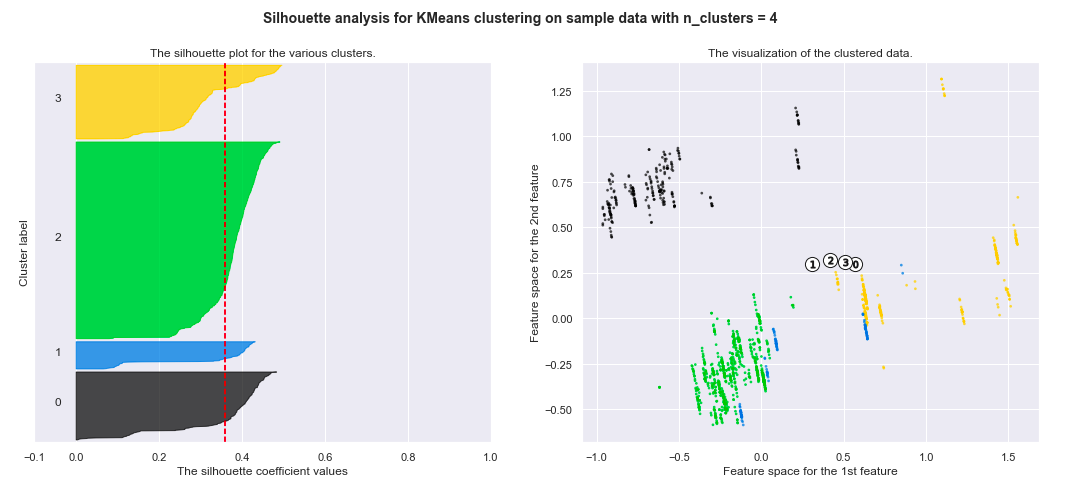


Figura 25. Visualización de clusters Kmeans. Figura propia

### Selección del modelo y balanceo

Cuando entrenamos un modelo, dividimos el conjunto de datos en dos conjuntos principales: entrenamiento y prueba. El conjunto de entrenamiento representa todos los ejemplos de los que un modelo está aprendiendo, mientras que el conjunto de prueba simula los ejemplos de prueba

#### 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 hay que eliminar las observaciones con valores nulos.

Por este motivo las características seleccionadas serán:

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

#### Modelos

Para la búsqueda del modelo que más se adapte a nuestras necesidades se ha utilizado RandomizedSearch de la librería sklearn.model\_selection, de skcikit learn con un valor para la validación cruzada de 5, se han probado con los datos sin balancear, con los datos balanceados con el parámetro class\_weight en balanceado, que penaliza para compensar, con sobremuestreo y con el balanceado combinado explicados en el apartado de imbalanced learn del marco teórico

Después de varias pruebas se ha visto que el mejor modelo para realizar este tipo de predicción es el Random Forest sin balancear los datos, obteniendo mejores resultados que con un Árbol de decisión, explicados en el marco teórico, ya que este, aun teniendo valores similares de f1-score, perdía la información de la importancia de las características, como se muestra en la Figura 26 que solo asignaba importancia a días\_any y al modelo, con los tres tipos de balanceado de datos se han obtenidos valores muy similares en la importancia de las características y f-1 score.

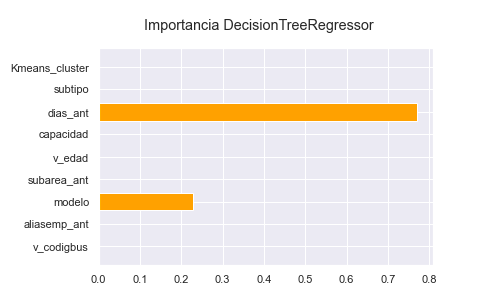


Figura 26. Importancia de las características DT. Figura propia

En comparación con el Adaboost si obtenemos la información de la importancia de las características como se muestra en la figura Figura 27 pero, con resultados similar de f1-score a los del Randomforest y con un coste computacional mucho mayor. obteniéndose valores muy similares con los tres tipos de balanceado de datos.

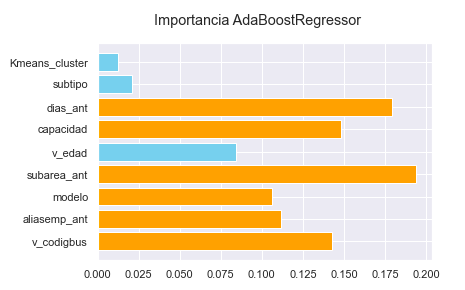


Figura 27. Importancia de las características AB. Figura propia

Por tanto el modelo seleccionado es el Random Forest, como se ha explicado en el marco teórico es un tipo de modelo que combina diversos árboles de decisión cada árbol depende de los características almacenadas en un vector aleatorio, probados independientemente y con la misma distribución, en este caso como se trata de un problema de clasificación, la salida es la clase seleccionada por la mayoría de árboles. al utilizar múltiples árboles se reduce considerablemente el riesgo de overfiting, normalmente da buenos resultados en problemas de clasificación, aunque no funciona tan bien con pocos datos.

El motivo por el que se opta por este tipo de modelo es, que es más simple y no difiere en exceso en resultados, mejorando así́ el coste computacional, por otro lado, este modelo sin balancear los datos presenta unos resultados que priorizan la seguridad de los vehículos, además de mostrarnos la importancia de cada variable para este modelo como se muestra en la Figura 28.

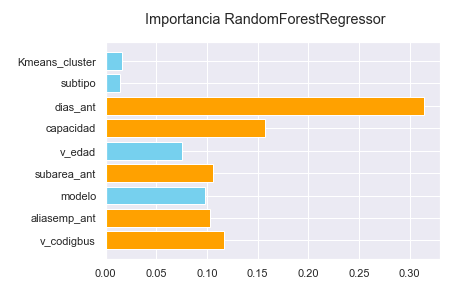


Figura 28. Importancia de las características RF. Figura propia

Para encontrar los hiperparámetros se ha usado RandomizedSearchCV y GridSearchCV, que son algoritmos para muestrear de manera efectiva el espacio de búsqueda y encontrar una buena solución inicialmente. Utilizando finalmente RandomizedSearchCV ya que con un número considerable de iteraciones se consigue valores muy similares a los de GridSearchCV que, a pesar de ser mucho más exhaustivo, el coste computacional es mucho mayor por lo que se ha utilizado en espacios de búsqueda menores acotados mediante el RandomizedSearchCV. Además de que se ha implementado cross validation que tiene dos pasos principales, dividir los datos en subconjuntos y rotar el entrenamiento y la validación entre ellos, en los algoritmos anteriormente mencionas ya viene integrado, utilizando 5 particiones.

## 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 más 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 más frecuencia el rango de tiempo entre ‘entre 0 y 2 meses’ aunque sea erróneamente, ya que es más 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 la Figura 29

Tabla

Descripción generada automáticamente

Figura 29. Informe de clasificación RF sin balancear

Como se puede observar en la Figura 29 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, Figura 30, 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 30. 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 más 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 la clase 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.

# Cuadros de mando

En la Figura 31 se muestra el porcentaje por marca y edad de las reparaciones por motor, en el ejemplo se puede ver que los autobuses de la marca IVECO, el 32% han entrado a taller por motor ha los 11 años, en un periodo de dos meses después de su última avería por motor, de la marca MAN un 11,98%, de la marca Mercedes un 9.35% y de la marca Scania un 18,29%, estas marcas, son las marcas más relevantes para la empresa por ser las que más autobuses tienen.

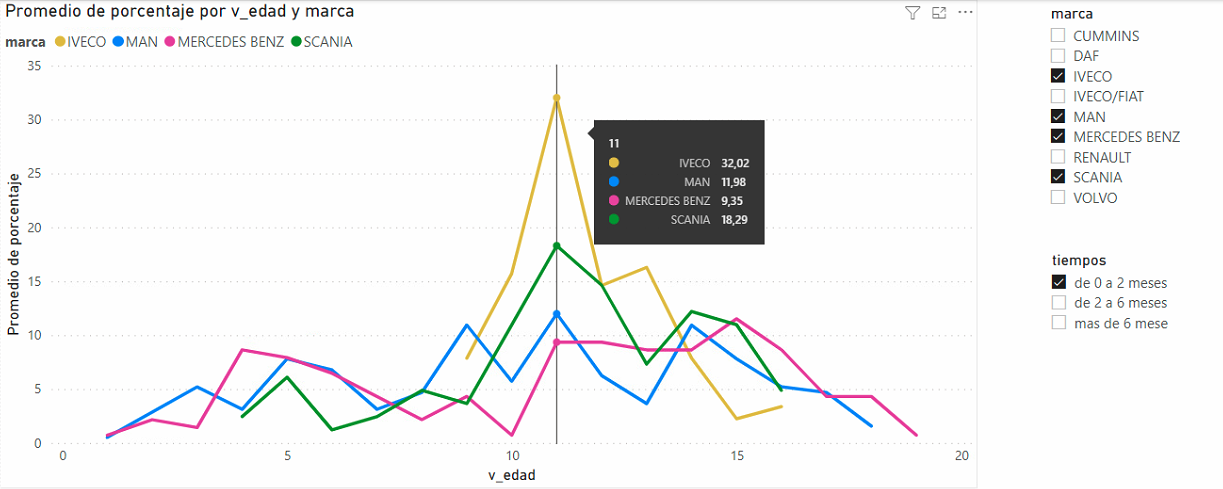


Figura 31. Porcentaje de vehículos reparados por edad y marca. Figura propia

En el siguiente cuadro de mando de la Figura 32 se pueden ver el recuento de averías por la subárea motor por marca en porcentaje y en valores absolutos pudiendo ver el porcentaje de reparaciones de cada empleado para cada grupo de tiempo, pudiendo seleccionar también que subtipo de autobús que se quiera mostrar

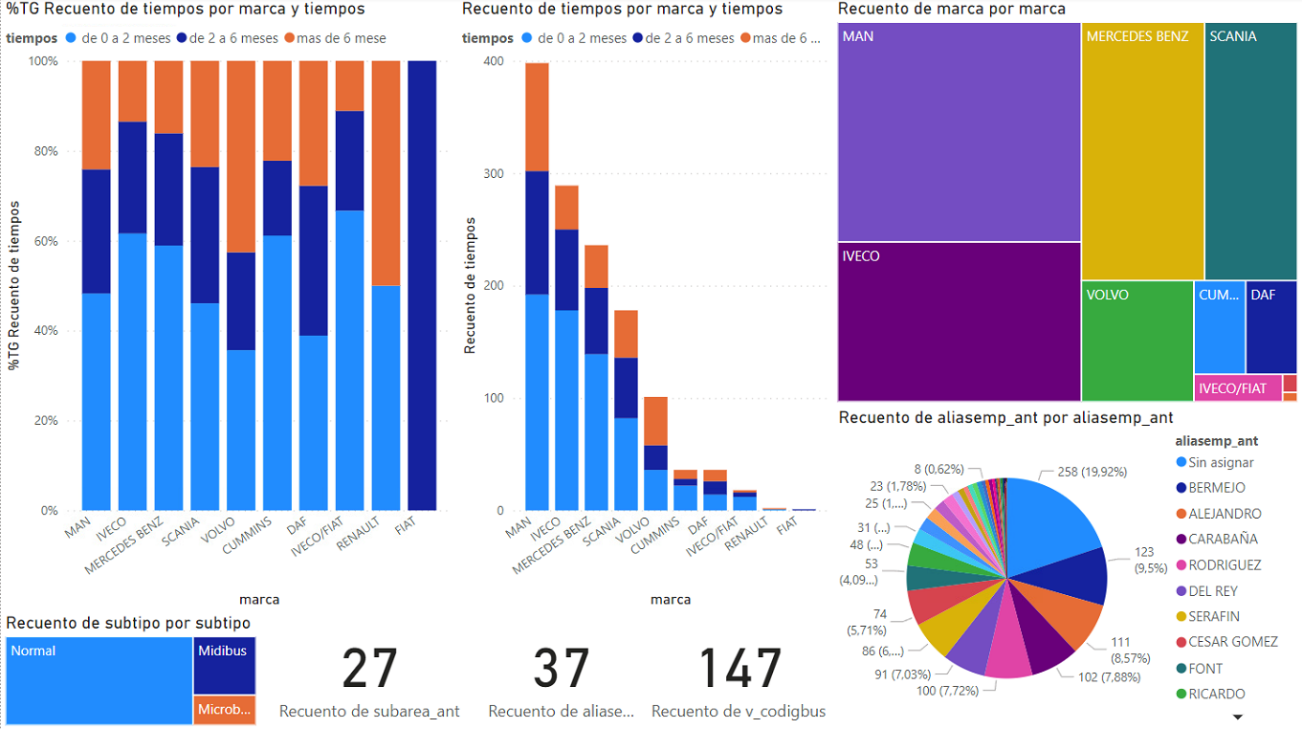


Figura 32. Cuadro de mando recuento de varias por marca. Figura propia

Se ha creado un cuadro de mando para poder analizar en porcentaje las subáreas anteriores que se han reparado antes de que hubiera un fallo de motor y a su vez ver que empleado las ha reparado mostrando los tres rangos de tiempo, permitiendo seleccionar el subtipo. Figura 33

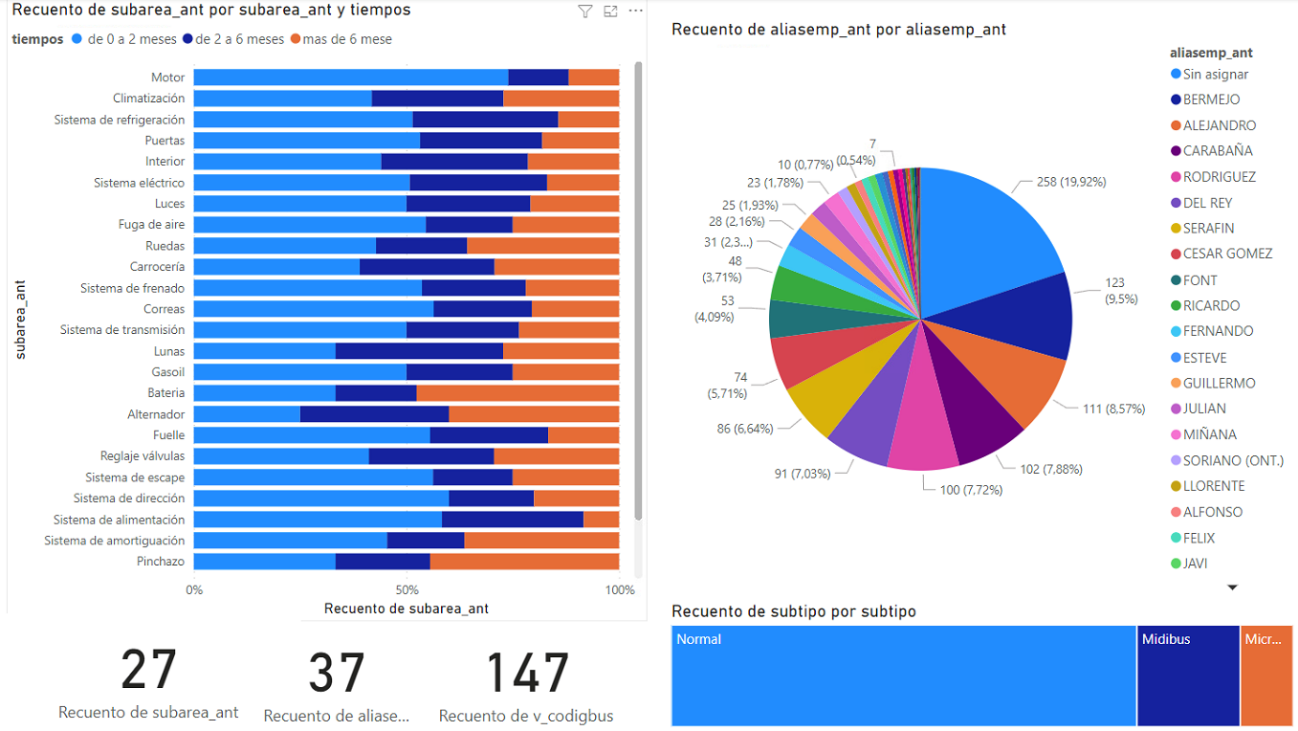


Figura 33. Cuadro de mandos subáreas anteriores. Figura propia

Se ha implementado un cuadro de mando Figura 34 para poder ver las características en función de las capacidades por marca ya que es una característica impórtate a la hora de que el modelo tome decisiones de predicción, tampien podemos ver la cantidad de averías por capacidad en el diagrama circular, y en el diagrfama de barras la cantidad de veces que se ha reparado un autobús en función del rango de tiempos, pudiendo filtrar por el subtipo

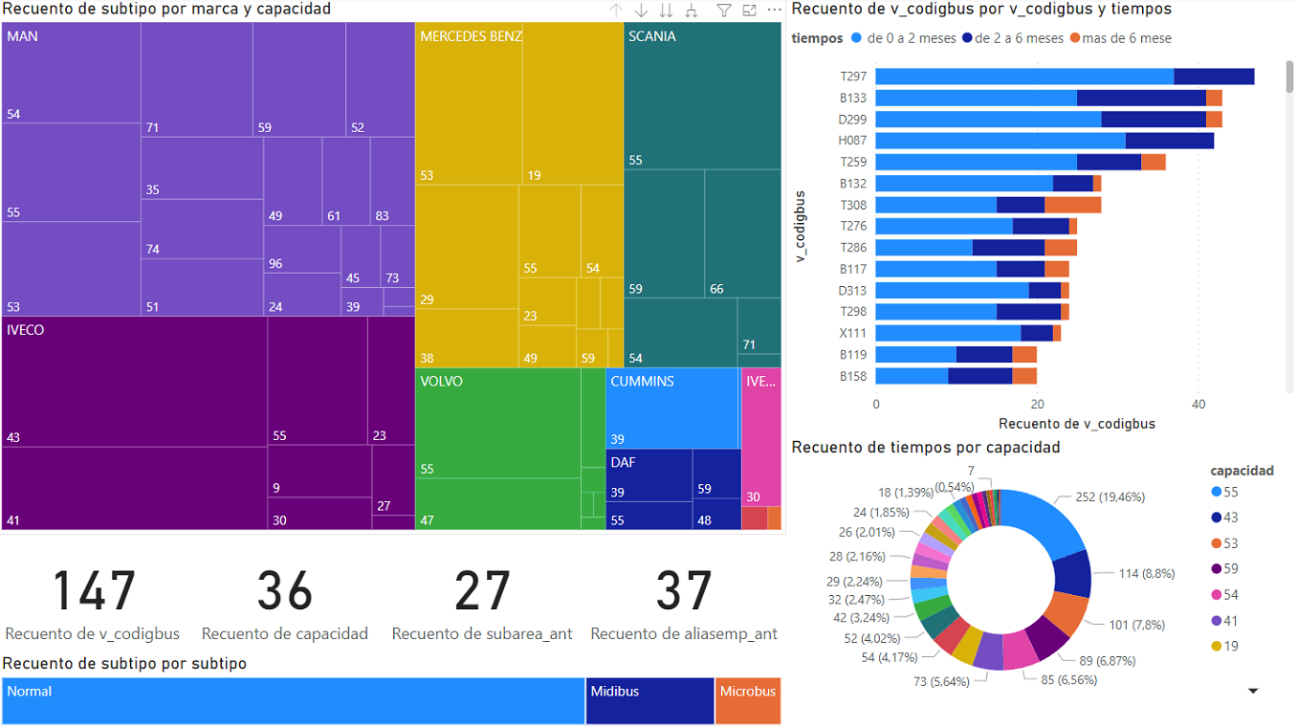


Figura 34. Cuadro de mando capacidades

Se ha implementado un cuadro de mando mostrado en la Figura 35 para poder localizar de forma rápida filtrando por características del autobús, en el ejemplo se está filtrando para ver los autobuses con 11 años

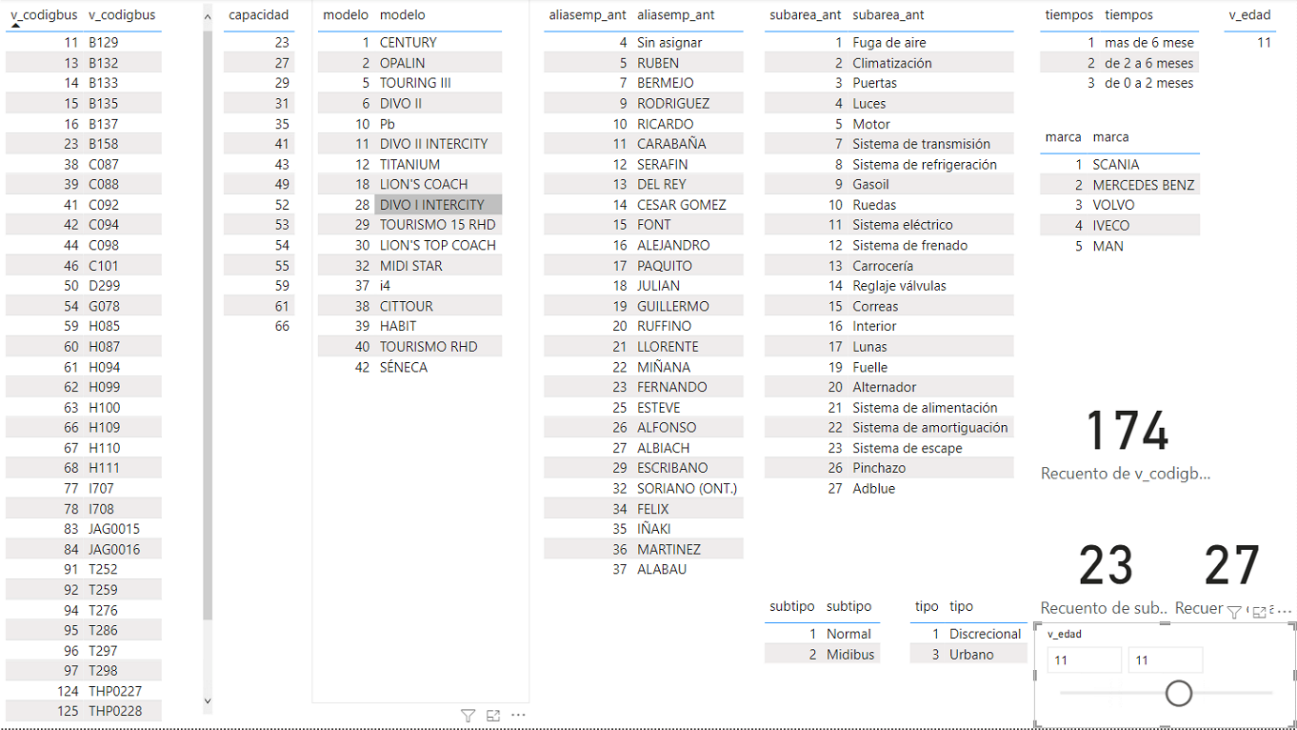


Figura 35. Cuadro de mando filtrado de características. Figura Propia

# Conclusiones

Como conclusión se puede observar que con las predicciones a partir de las características del autobús no obtenemos los valores deseados, pero si podemos decir que el modelo antepone la seguridad de los autobuses en el caso de errar, pudiendo mejorar dichos valores de predicción añadiendo datos que se puedan relacionar con el desgaste de piezas relacionadas con una reparación de motor, o bien aumentando el número de observaciones ya que no era muy alto.

Como se observa en la Figura 28 vista anteriormente el número de días desde la anterior operación es la variable más importante para tomar la decisión para averiguar el tiempo que va a pasar desde una avería por subárea motor a la próxima avería por subárea motor.

La capacidad, el código del autobús y la subárea anterior reparada también aportan información relevante para poder predecir. Con el modelo Random Forest

Como se ha explicado en el apartado de resultados el modelo Random Forest, para este caso particular es similar en precisión a los otros modelos, y no es muy alta en las clases que más tiempo tarda el autobús en ir a taller, de la variable ‘tiempos’, se ha elegido un modelo que anteponga la seguridad de los autobuses a la precisión, esto quiere decir, que se ha priorizado que prediga con más frecuencia el rango de tiempo entre ‘entre 0 y 2 meses’ aunque sea erróneamente, ya que es más 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.

# Referencias

1. Freund, Y., Schapire, R., & Abe, N. (1999). A Short Introduction to Boosting. Journal-Japanese Society for Artificial Intelligence, 14(771-780), 1612. Doi:10.1.1.124.5117
2. De Oliveira, J. V., & Pedrycz, W. (Eds.). (2007). Advances in Fuzzy Clustering and its Applications. John Wiley & Sons, Inc. New York, USA
3. Bishop, C. (2007). Pattern Recognition and Machine Learning (Information Science and Statistics), Springer, New York.
4. AGRESTI, A. (2007): An introduction to Categorical Data Analysis. Wiley, New York.
5. Leek, J. (2015). The Elements of Data Analytic Style. J. Leek. —Amazon Digital Services, Inc.
6. Shalev-Shwartz, S., & Ben-David, S. (2014). Understanding machine learning: From theory to algorithms. Cambridge university press
7. VanderPlas, J. (2016). Python data science handbook: essential tools for working with data. O'Reilly Media, Inc
8. Fayyad, U., Piatetsky-Shapiro, G., Smyth, P. (1996). The KDD Process for Extracting Useful Knowledge from Volumes of Data. Communications of the ACM, November 1996/ Vol 39, Nº 11, 27–34.
9. Siegel, E. (2013). Analítica predictiva. Predecir el futuro utilizando Big Data. Anaya Multimedia-Anaya Interactiva.
10. Mayer-Schönberger, V., Cukier, K. (2013). Big data. La revolución de los datos masivos. Turner
11. Alfaro Cortez, Esteban; Gamez Martinez, Matias; Garcia Rubio, Noelia;. (2002). Una Revisión de los Métodos de Agregación de Clasificadores. Plaza de la Universidad, s/n. 02071 Albacete.: Universidad de Castilla-La Mancha.
12. Arnejo Calviño, H. (2017). Métodos para la mejora de predicciones en clases desbalanceadas en el estudio de bajas de clientes (CHURN). España.
13. Beltrán Pascual, M. (2015). Diseño e implementación de un nuevo clasificador de préstamos bancarios a través de la minería de datos. Madrid.
14. Breiman, L. (2001). Random Forest. California: Statistics Department
15. Brownlee, J. (Agosto de 2015). Tactics to Combat Imbalanced Classes in Your Machine Learning Dataset. Obtenido de <http://machinelearningmastery.com/tactics-to-combatimbalanced-classes-in-your-machine-learning-dataset/>
16. Cánovas García, F., Alonso Sarría, F., & Gomariz Castillo, F. (2016). MODIFICACIÓN DEL ALGORITMO RANDOM FOREST PARA SU EMPLEO EN. Málaga.
17. Cortez, E. A. (2006). Combinación de clasificadores mediante el método boosting, una aplicación a la predicción del fracaso empresarial en España. Madrid: Castilla La Mancha
18. Medina Merino, Rosa ; Ñique Chacón, Carmen;. (2017). Bosques aleatorios como extensión de los árboles de clasificación con los programas R y Python. Lima.
19. López Pineda, A. (2008). Algoritmos de balanceo de clases en problemas de clasificación binaria de conjuntos altamente desproporcionados. Mexico.
20. Xie, Y., Li, X., Ngai, E., & Ying, W. (2008). Customer churn prediction using improved balanced random forests. Hong Kong: Department of Management and Marketing,.
21. Sergio P. Santos and Jose Aifredo F. Costa. (2008). “A comparison between hybrid and nonhybrid classifiers in diagnosis of induction motor faults”. Proceedings. IEEE 11th International Conference on Computational Science and Engineering, CSE 2008. 2008. isbn: 9780769531939. doi: 10.1109/CSE.2008.60.
22. Gareth James et al. An introduction to Statistical Learning. 2000. isbn: 978-1-4614-7137- 0. doi: 10.1007/978-1-4614-7138-7. arXiv: arXiv:1011.1669v3.
23. Yoav Freund and Robert E. Schapire. “A decision-theoretic generalization of on-line learning and an application to boosting”. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). Vol. 904. Springer Verlag, 1995, pp. 23–37. isbn: 9783540591191. doi: 10.1006/jcss. 1997.1504.
24. Guolin Ke et al. (2017). “LightGBM: A highly efficient gradient boosting decision tree”. Advances in Neural Information Processing Systems.
25. Erik Strumbelj and Igor Kononenko. (2014). “Explaining prediction models and individual pre- ˇ dictions with feature contributions”. Knowledge and Information Systems issn: 02193116. doi: 10.1007/s10115-013-0679-x.
26. David Conde et al. “Isotonic boosting classification rules”. Accepted for publication Advances in Data Analysis and Classification (2020). doi: 10.1007/s11634-020-00404-9.
27. Cano, G., Luque, M. ;, Sendra, A. ;, Ruiz, L., Ramón, J., Roldán, C., … Manuel, C. : (2019). Mantenimiento Predictivo Mediante Tecnicas de Machine Learninig. AEIPRO, 23 International Congress on Project Management and Engineering, 03(020), 721–730. Retrieved from <http://dspace.aeipro.com/xmlui/handle/123456789/2293>
28. Cano, G., Luque, M. ;, Sendra, A. ;, Ruiz, L., Ramón, J., Roldán, C., … Manuel, C. : (2019). Mantenimiento Predictivo Mediante Tecnicas de Machine Learninig. AEIPRO, 23 International Congress on Project Management and Engineering, 03(020), 721–730. Retrieved from <http://dspace.aeipro.com/xmlui/handle/123456789/2293>
29. Janitza, S., Tutz, G., & Boulesteix, A.-L. (2014). Random Forests for Ordinal Response Data: Prediction and Variable Selection. In Elsevier. Retrieved from <http://www.stat.unimuenchen.dehttp//www.ibe.med.uni>
30. KAMBATLA, Karthik, et al. Trends in big data analytics. Journal of Parallel and Distributed Computing, 2014, vol. 74, no 7, p. 2561-2573.
31. O'Reilly Media, Inc. Big Data Now: 2012. " O'Reilly Media, Inc.", 2012.
32. ABC. (2016, febrero) Big data: ¿vidas privadas al alcance de todos?. URL: http://www.abc.es/tecnolo- gia/informatica-software/20131028/abci-entrevista-data201310221252.html
33. BBVA Innovation Center. (2016, febrero), Proyecto Big Data. URL: http://www.centrodeinnova- cionbbva.com/proyectos/big-data
34. BASANTA-VAL, P., et al. Improving the predictability of distributed stream processors. Future Generation Computer Systems, 2015, vol. 52, p. 22-36.
35. SALMERON, J. (2016, enero). "¿Qué herramientas necesitas para iniciarte en Big Data?". Recuperado el 21 de Septiembre de 2018, de inLabFIB: https://in- lab.fib.upc.edu/es/blog/que-herramientas-necesitas-para-iniciarte-en-big-data
36. Richard Szeliski. (2010). Computer Vision: Algorithms and Applications. SpringerVerlag New York, Inc., New York, NY, USA, 1st edition.
37. Pedregosa, F., Varoquaux, Ga"el, Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... others. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Re- search, 12(Oct), 2825–2830.
38. Sylvain Arlot and Alain Celisse. (2009). A survey of cross-validation procedures for model selection. 4, 40–79. DOI:https://doi.org/10.1214/09-SS054
39. W Hu, W Hu, and S Maybank. (2008). AdaBoost-Based Algorithm for Network Intrusion Detection. IEEE Trans. Syst. Man, Cybern. Part B 38, 2 (2008), 577–583. DOI:https://doi.org/10.1109/TSMCB.2007.914695
40. Sanjay Kumar Palei and Samir Kumar Das. (2009, enero). Logistic regression model for pre- diction of roof fall risks in bord and pillar workings in coal mines: An approach. Saf. Sci. 47,88–96. DOI:https://doi.org/10.1016/J.SSCI.2008.01.002