ESCUELA INTERNACIONAL DE POSTGRADOS

**TRABAJO FINAL DE MÁSTER**

**TITULACION:** Máster de programación Avanzada en Python para Big Data, Hacking y Machine Learning

**ALUMNOS:**

* Jose Cabezas Pulgarín
* Pablo Ruiz Veredia
* Victor Simo Lozano. 53879955V

**FECHA DE ENTREGA:** septiembre 2023

Agradecimientos

Contenido

[Abstract 3](#_Toc139821192)

[Introducción 4](#_Toc139821193)

[1 Objetivos generales del trabajo 4](#_Toc139821194)

[2 Justifica la elección del tema. 4](#_Toc139821195)

[3 Resumen metodología 5](#_Toc139821196)

[3.1 Procedimientos 5](#_Toc139821197)

[3.2 Agile 5](#_Toc139821198)

[3.3 Control de versiones: Git 5](#_Toc139821199)

[4 Estructura 5](#_Toc139821200)

[Planteamiento del problema 6](#_Toc139821201)

[Objetivos del trabajo 7](#_Toc139821202)

[Metodología 7](#_Toc139821203)

[1 Preprocesamiento del Dataset 7](#_Toc139821204)

[1.1 Dataset global 7](#_Toc139821205)

[1.2 Forecast 7](#_Toc139821206)

[1.3 Crew 8](#_Toc139821207)

[1.4 Airplane(PABLO) 8](#_Toc139821208)

[2 Generar el modelo con pycaret 8](#_Toc139821209)

[2.1 Forecast(VICTOR) 8](#_Toc139821210)

[2.2 Crew 8](#_Toc139821211)

[2.3 Airplane(PABLO) 9](#_Toc139821212)

[3 App de Streamlit 9](#_Toc139821213)

[3.1 Archivo .Streamlit 9](#_Toc139821214)

[3.2 Directorio Foretast 9](#_Toc139821215)

[3.3 Directorio Airplane 9](#_Toc139821216)

[3.4 Directorio Crew 9](#_Toc139821217)

[3.5 Directorio pages. 9](#_Toc139821218)

[3.6 Source(INCOMPLETO) 16](#_Toc139821219)

[3.7 0\_AAP\_Home. 16](#_Toc139821220)

[4 Unión de las partes y prueba en local 17](#_Toc139821221)

[5 Subida a la StreamlitCloud 17](#_Toc139821222)

[Evaluación de los resultados 18](#_Toc139821223)

[Conclusiones 19](#_Toc139821224)

[Referencias 20](#_Toc139821225)

[Anexos TFM 21](#_Toc139821226)

# Abstract

This master's thesis is focused on the development of a Python-based system for aircraft accident prediction (AAP). To achieve this, PyCaret library has been used to obtain a Machine Learning Model with a pre-processed aircraft dataset. In addition, Streamlit has been used to create a GUI able to introduce the model values and get the incident prediction.

To get the result, an exhausted data-preprocessing with the aircraft accident dataset has been necessary. This includes tasks such as data cleaning, outlier removal, and variable transformation. Through this process, is obtained a refined dataset that can be used in the predictive models.

PyCaret library is used to develop a machine learning model capable of predict the occurrence of aircraft accidents. PyCaret allows exploring different classification algorithms and determining the best fit for a dataset. In addition, it greatly simplifies the process of model training and evaluation.

Finally, Streamlit library is used to create an interactive user interface (GUI), which allows create a form where users can enter relevant data for forecasting, including date, aircraft type and airline, etc. Streamlit makes it easy to create the GUI, enabling rapid development and deployment of interactive web applications.

In summary, this thesis focuses on the development of a Python-based aircraft accident prediction system that uses Streamlit to create user input forms and PyCaret to train and evaluate predictive models. This work includes preprocessing the aircraft accident data set, choosing the most appropriate classification algorithm and creating a user interface. With this system, we aim to contribute to improving the safety of the aviation industry and prevent potential accidents.

*KEYWORDS: AAP, Streamlit, PyCaret, Machine Learning, BigData, Python*

# Introducción

## Objetivos generales del trabajo

El objetivo general del proyecto de Fin de Máster (TFM) es desarrollar un sistema de predicción de accidentes aéreos utilizando programación Python y Machine Learning. Haciendo uso de las bibliotecas Streamlit y PyCaret junto con un conjunto BigData de accidentes aéreos preprocesados.

El objetivo principal es ayudar a mejorar la seguridad en la industria de la aviación proporcionando herramientas para predecir accidentes de aeronaves. La capacidad de predecir posibles accidentes puede facilitar la toma de decisiones y la implementación de medidas preventivas por parte de las autoridades, las aerolíneas y otros actores de la industria. Además, se fija el objetivo de desarrollar un sistema eficiente y fácil de usar que utilice la programación Python y las bibliotecas Streamlit y PyCaret.

Además, se proporciona una interfaz intuitiva que permita a los usuarios ingresar datos relevantes y predecir con precisión la probabilidad de un accidente aéreo.

En resumen, este trabajo se centra en el desarrollo de sistemas de predicción de accidentes de aviación para mejorar la seguridad en la industria aeronáutica. Utilizando la programación de Python, las bibliotecas Streamlit y PyCaret y los conjuntos de datos preprocesados, ofreciendo así una herramienta eficiente y fácil de usar para predecir la probabilidad de accidentes aéreos.

## Justifica la elección del tema.

La elección de los temas del TFM se basa en varias razones importantes. La seguridad en la industria de la aviación es un tema muy importante para proteger la vida humana y la economía mundial. Los accidentes de aeronaves pueden tener consecuencias devastadoras, que incluyen la pérdida de vidas, daños a la propiedad y consecuencias legales y financieras.

Por lo tanto, el desarrollo de un sistema de predicción de accidentes de aviación es una medida importante para prevenir y reducir los riesgos asociados y mejorar la seguridad en esta importante área. Además, los avances en el procesamiento de datos y los métodos de aprendizaje automático brindan nuevas oportunidades para resolver problemas complejos, como la predicción de accidentes de aeronaves. Usando las capacidades de programación de Python y bibliotecas especializadas como Streamlit y PyCaret, se puede desarrollar sistemas eficientes y precisos.

La razón por la que se opta por utilizar conjuntos de datos de accidentes de aeronaves recopilados previamente es también la disponibilidad de información valiosa y la reducción de costos y tiempo asociados con la recopilación de nuevos datos.

Al analizar los patrones y los factores de riesgo en estos datos históricos, se puede crear modelos predictivos confiables que ayuden a comprender mejor los accidentes de aeronaves e implementar medidas preventivas más efectivas. Es por ello que el desarrollo de un sistema de predicción de accidentes de aeronaves es una herramienta invaluable para los tomadores de decisiones de la industria de la aviación, como los reguladores, las aerolíneas y los investigadores de seguridad.

En resumen, al proporcionar información precisa sobre la probabilidad de incidentes, se puede implementar estrategias de seguridad más efectivas y se puede tomar medidas preventivas proactivas. Es por ello que, la selección del tema para el TFM se basa en la **importancia de mejorar la seguridad en la industria de la aviación**, los avances tecnológicos en aprendizaje automático, el uso de datos existentes y la contribución a la toma de decisiones en esta área.

## Resumen metodología

Para conseguir el resultado final del TFM, además de realizar un trabajo estructurado entre todos los integrantes del grupo de trabajo, se hace uso de diferentes herramientas que facilitan este trabajo, como es el uso de un **control de versiones** basado en tecnología Git y un método de trabajo **Agile** con una metodología Scrum.

### Procedimientos

El TFM se ha realizado por un total de tres integrantes, utilizando un enfoque colaborativo para desarrollar un sistema de predicción de accidentes de aviación. Cada miembro del equipo se enfoca en construir un modelo predictivo utilizando un grupo diferente de columnas de datos. Donde cada integrante se hace responsable de los datos de la tripulación, otro de los datos de vuelo y un tercero de la meteorología.

Primero, cada miembro del equipo crea individualmente su propio modelo de predicción. Usando grupos específicos de columnas, usamos algoritmos y técnicas de aprendizaje automático para entrenar un modelo capaz de predecir la probabilidad de un accidente aéreo.

Luego se crear la interfaz de usuario para cada uno de estos modelos páginas haciendo uso de la biblioteca Streamlit, con esta, cada miembro del equipo diseña un formulario para ingresar los datos de su modelo.

Adicionalmente, se crea una página final para combinar y analizar los resultados obtenidos de cada modelo. En esta, se aúna los resultados de cada algoritmo y quedan representados globalmente. Donde se evalúa la posibilidad de un accidente se tiene en cuenta por el número de accidentes obtenidos en los modelos individuales.

### Agile

### Control de versiones: Git

## Estructura

La estructura que se utiliza en el TFM se basa en un enfoque colaborativo de tres personas, detallada a continuación:

* **Distribución de la responsabilidad**

Cada miembro del equipo asume la responsabilidad de una parte específica del proyecto. Uno de ellos se centra en los datos de la tripulación, el segundo en los datos de vuelo y el tercero en los datos meteorológicos. Esta división permite un enfoque más especializado y eficiente de cada área.

* **Construcción de modelos individuales**

Cada miembro del equipo trabaja individualmente para construir modelos predictivos utilizando grupos específicos de columnas. Se usa algoritmos y técnicas de aprendizaje automático para entrenar y adaptar estos modelos para predecir la probabilidad de un accidente aéreo.

* **Desarrollo de páginas individuales en Streamlit**

Se usa la biblioteca Streamlit de Python para crear páginas individuales donde cada miembro del equipo diseña un formulario para ingresar datos específicos para su modelo. En estas páginas, los usuarios pueden ingresar información relacionada con la tripulación de vuelo, la aeronave y el clima, respectivamente.

* **Página final para análisis global**

Además de las páginas individuales, se crea una página final en Streamlit donde se recopila los resultados de cada modelo para su análisis. En esta página, se explica los resultados de cada algoritmo y se brinda una visión global de la probabilidad de accidentes aéreos, teniendo en cuenta la información de todos los modelos.

Con este enfoque colaborativo, se permite aprovechar la experiencia personal y el conocimiento de cada miembro del equipo sobre diferentes aspectos del problema de predicción de accidentes de aviación. Donde juntos, se logra desarrollar un sistema más completo y sólido que cubra aspectos clave como la tripulación, la aeronave y los datos meteorológicos.

# Planteamiento del problema

En la formulación del problema, existe el desafío de mejorar la seguridad en la industria de la aviación mediante la predicción de accidentes de aeronaves. Los accidentes de aeronaves tendrán importantes consecuencias, como pérdidas, daños materiales y consecuencias financieras. Por lo tanto, es muy importante desarrollar herramientas para predecir la ocurrencia de estos eventos y tomar medidas preventivas para evitarlos.

A pesar de las numerosas medidas y protocolos de seguridad vigentes en la industria de la aviación, los accidentes aún ocurren y es necesario abordar el problema de manera proactiva. Actualmente, las decisiones de gestión de seguridad de la aviación se basan en datos históricos y análisis manual, que pueden ser limitantes y no brindan una imagen completa del riesgo.

El objetivo principal de nuestro TFM es desarrollar un sistema de predicción de accidentes aéreos utilizando técnicas de aprendizaje automático y datos relevantes. La idea es aprovechar la disponibilidad de conjuntos de datos de accidentes de aviones recopilados previamente y utilizar algoritmos de aprendizaje automático para crear modelos predictivos.

El modelo nos permite estimar la probabilidad de un accidente aéreo en función de varios factores, como la información de la tripulación, los datos de la aeronave y las condiciones meteorológicas. Anticipar la probabilidad de un accidente puede conducir a medidas proactivas para mejorar la seguridad, como ajustar los procedimientos, realizar cambios en el diseño de la aeronave o mejorar la capacitación y educación de la tripulación de vuelo.

Nuestra declaración del problema TFM se centra en la necesidad de mejorar la seguridad en la industria de la aviación mediante la predicción de accidentes de aeronaves. Utilizando técnicas de aprendizaje automático y datos relacionados, nuestro objetivo es desarrollar un sistema predictivo que proporcione información valiosa para la toma de decisiones y medidas preventivas, ayudando así a reducir el riesgo y garantizar la seguridad en la industria de la aviación.

# Objetivos del trabajo

El objetivo general de nuestro trabajo es desarrollar un sistema de predicción de accidentes aéreos utilizando técnicas de aprendizaje automático y datos relacionados. El objetivo de la iniciativa es mejorar la seguridad en la industria de la aviación y prevenir activamente los accidentes y sus consecuencias. Para lograr este objetivo general, formulamos una serie de objetivos secundarios que se derivan del objetivo principal y justifican y permiten que se desarrollen.

En primer lugar, decidimos realizar un análisis exploratorio exhaustivo de los datos existentes sobre accidentes de aeronaves. A través de este análisis, tratamos de comprender la estructura de los datos, identificar patrones importantes, detectar valores atípicos e identificar posibles correlaciones entre variables.

El segundo objetivo se centra en el preprocesamiento de datos. Aquí, nos enfocamos en realizar tareas de limpieza, manejo de valores faltantes, estandarización de variables y transformaciones necesarias para garantizar la calidad y consistencia de los datos utilizados en los modelos predictivos.

El tercer objetivo es que cada miembro del equipo tenga la tarea de crear un modelo de predicción individual utilizando un grupo específico de columnas de datos. Estos paquetes incluyen información de la tripulación, datos de vuelo y datos meteorológicos. Para hacer esto, usamos algoritmos de aprendizaje automático para entrenar y adaptar un modelo en un intento de predecir la probabilidad de un accidente aéreo individual.

El siguiente objetivo es implementar la interfaz de usuario utilizando la biblioteca Streamlit. La interfaz está diseñada para ser intuitiva y fácil de usar, lo que permite a los usuarios ingresar datos relevantes en cada modelo de predicción creado por los miembros del equipo. Esta implementación facilita la interacción con el sistema y aumenta su usabilidad.

Finalmente, el objetivo principal es integrar los resultados de los modelos individuales y realizar un análisis global. Este análisis nos permite considerar combinaciones de resultados individuales para explicar conjuntamente la probabilidad de un accidente aéreo. Este enfoque le permite obtener una imagen más precisa y confiable de los posibles accidentes. En resumen, nuestro trabajo se basa en el desarrollo de un sistema de predicción de accidentes aeronáuticos, donde los objetivos secundarios son realizar análisis exploratorios, preprocesar los datos, crear modelos individuales, implementar una interfaz de usuario y realizar un análisis global de los resultados buscando complementarse entre sí para lograr mejores objetivos generales de seguridad en la industria de la aviación.

# Metodología

## Preprocesamiento del Dataset

### Dataset global

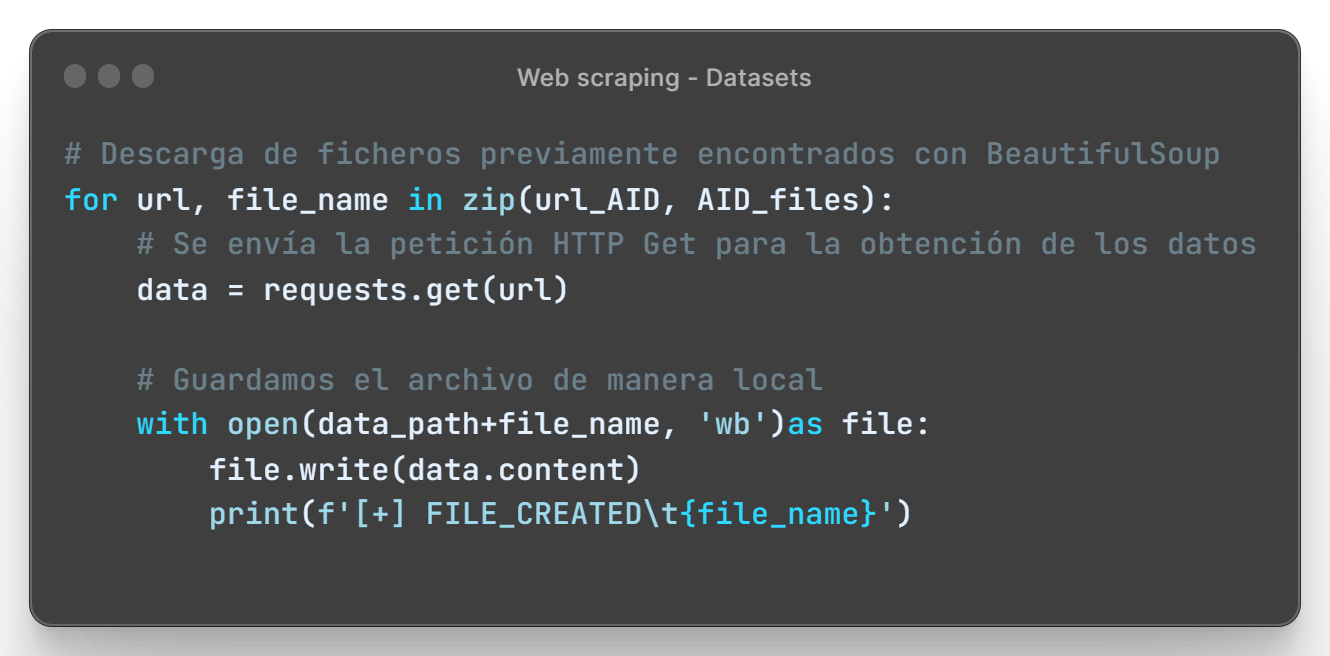
Los modelos de predicción de accidentes que se desarrolla en el proyecto, parten de un conjunto de datos común obtenido de (Administration, s.f.).

En este repositorio, se puede encontrar los datos en formato *txt* desde el año 1975 hasta el presente. Para hacer uso de los datos agrupados todos en un único conjunto de datos, se realiza un código en un Jupyter Notebook que permita realizar dicha operación para el uso común de los datos que se pretende.

Primeramente, se realiza un web scraping con el que se es capaz de acceder a los ficheros *txt* de la fuente. En este primer paso se almacena en una lista, todos aquellos ficheros que se ha encontrado disponibles, para finalmente recorrer la lista y descargarlos en nuestro directorio local accediendo directamente a su url.

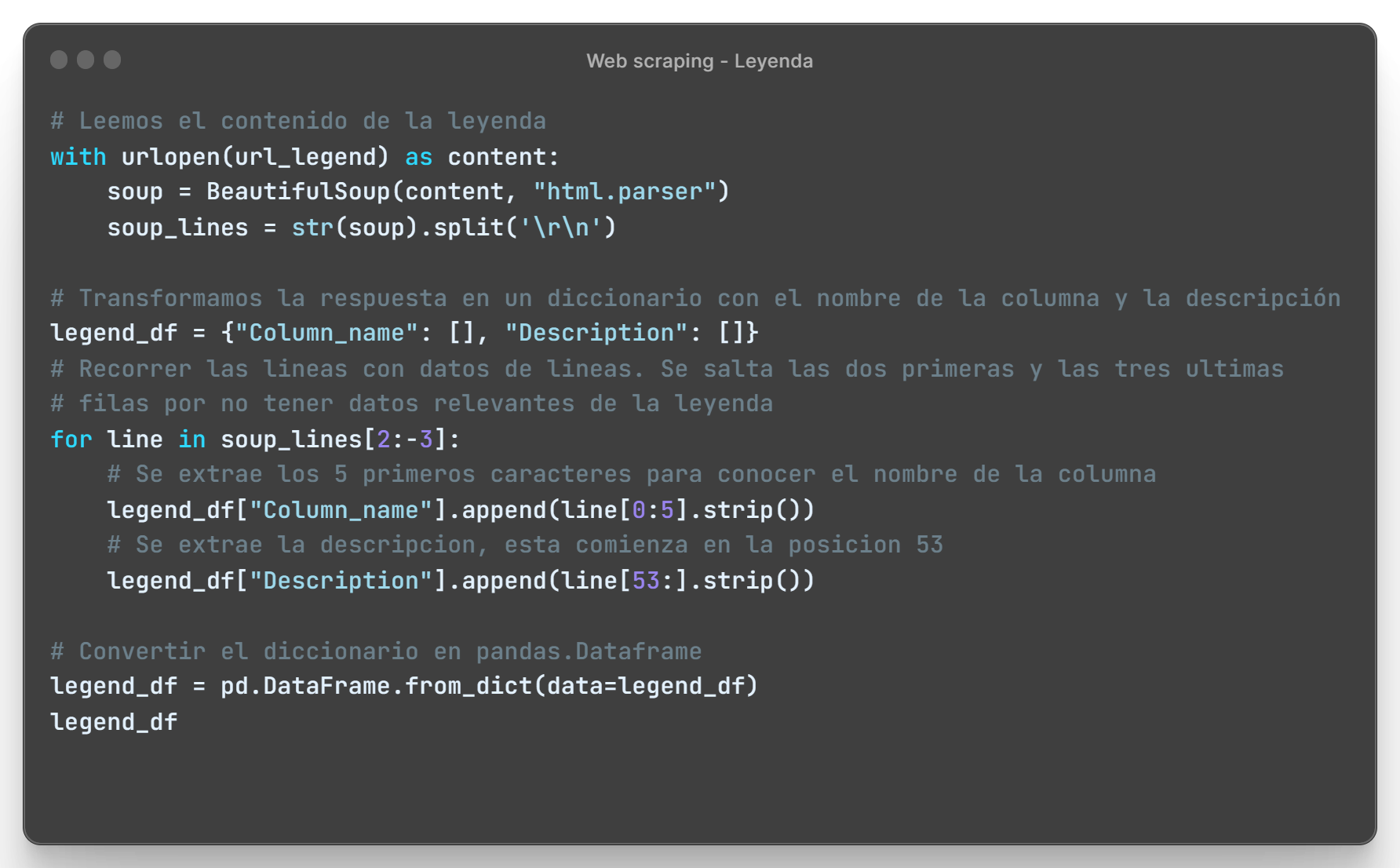


Script 1 Web scraping – Datasets ficheros



Script 2 Web scraping - Datasets descarga

Del mismo modo que con los ficheros de datos, se realiza web scraping para la leyenda de los datos, almacenada en *txt* en la fuente de datos. Una vez se obtenga esta, se guarda en el directorio local en formato *csv* para poder trabajar con ella más adelante. Esta leyenda nos permite conocer el titulo de las columnas, que como se verá más adelante, en la fuente de los datos se define de forma abreviada por c + numero identificativo.

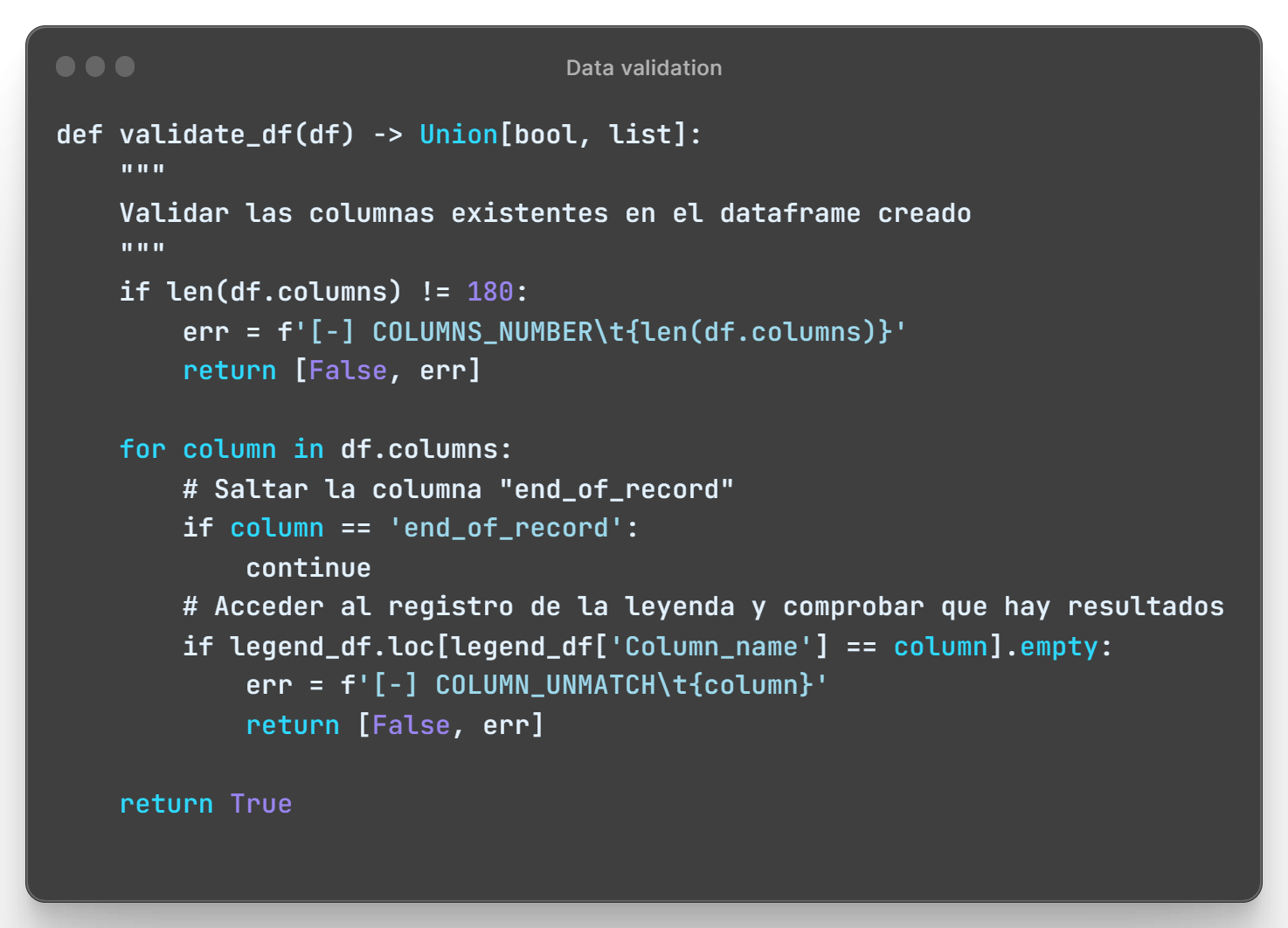


Script 3 Web scraping – Leyenda

Una vez se ha obtenido todos los ficheros, se realiza los pasos para poder crear el conjunto de todos los datos en un único dataset. Mediante un análisis previo de los datos obtenidos, se sabe que el numero de columnas existente es de 180, dado que hay un gran número de columnas, se realiza una función que valide la creación del dataset global. En esta función ***validate\_df*** se realiza las siguientes comprobaciones:

* Longitud del fichero es de 180 columnas.
* Las columnas se encuentran en la leyenda.

Con estas validaciones, en el caso de no ser cumplidas, se devolverá un error y no se continuará con la creación del dataframe global hasta sanar los errores.



Script 4 Validación de dataframes

Una vez se tiene el conjunto de datos global, como es de esperar en cualquier proceso de recogida de datos, existe valores que no son los deseados o valores nulos. Es por ello que previo a dar por finalizado el dataframe global, se realiza una serie de comprobaciones y limpieza de datos.

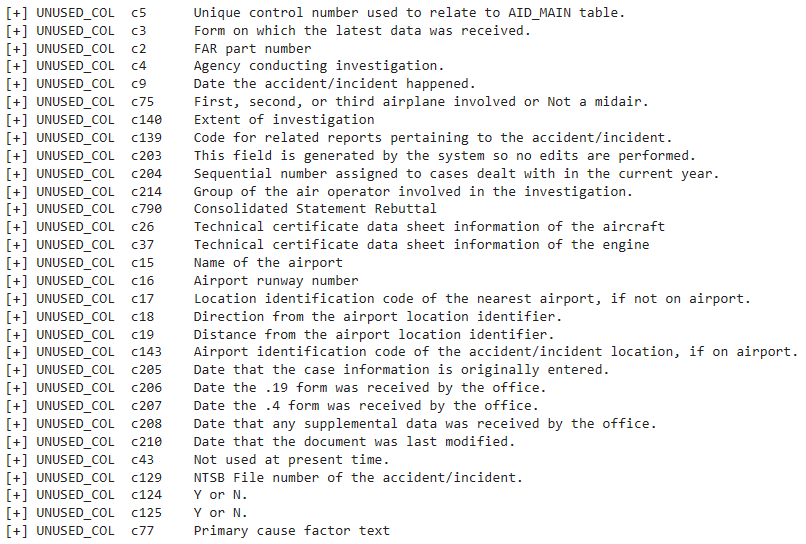
Para la limpieza de los datos, se hace uso de una función creada ***clean\_data***, con la que poder imputar como valores ***NaN*** aquellos nulos o vacíos. De este modo, teniendo como NaN todos los valores indeseados, se puede realizar posteriormente una limpieza de las columnas que contenga todos sus datos nulos. Como resultado de esta operación, se elimina la columna *end\_of\_record* quedando, por lo tanto, un dataframe global de 179 atributos y 214887 registros. Un total de más de 38 millones de datos.



Script 5 Dataframe global – Limpieza

Dado el elevado número de datos, se realiza a continuación, un cribado de todos los datos, de modo que se pueda reducir la dimensión del conjunto de datos. En este cribado de los datos, se tiene en cuenta aquellas columnas que no van a aportar ninguna información útil a ninguno de los modelos a desarrollar o su información no es analizable.

Tras el proceso de eliminación de columnas, un total de 30 columnas son eliminadas, se queda una dimensión de 149 atributos, lo que hace un total de 32 millones de datos, reduciéndose en un 15% el conjunto de datos.



1 Columnas eliminadas

Una vez con la dimensión deseada del conjunto de datos, se realiza el guardado de dataframe. Por el elevado número de registros, se valora realizarlo en formato “***.parquet”*** por estar preparado para conjunto de datos **BigData,** para ello, se uso de la librería **pyarrow**.

Durante la conversión del fichero, salta una serie de errores ***ArrowInvalid*** ya que el modo de proceder para generar el fichero es realizar una conversión binaria de los datos. Por lo tanto, es necesario que todos los datos de una columna tengan el mismo formato. Para poder subsanar las excepciones y convertir finalmente el fichero, se realiza una serie de conversiones en las columnas afectadas.

### Forecast

#### Obtención de datos

Como se menciona, previamente se ha obtenido los datos de la fuente y preprocesados para poder trabajar en la creación de los modelos, partiendo de la misma base.

En este caso, para obtener los datos en un Dataframe, se accede a los datos previamente almacenador en formato “*.parquet”.*

Además, se crea el Dataframe, leyenda, es decir, la descripción de las columnas para trabajar con la información completa.



Script 6 Obtención de los datos

#### Exploratory Data Analysis (EDA)

**Reducción de dimensionalidad**

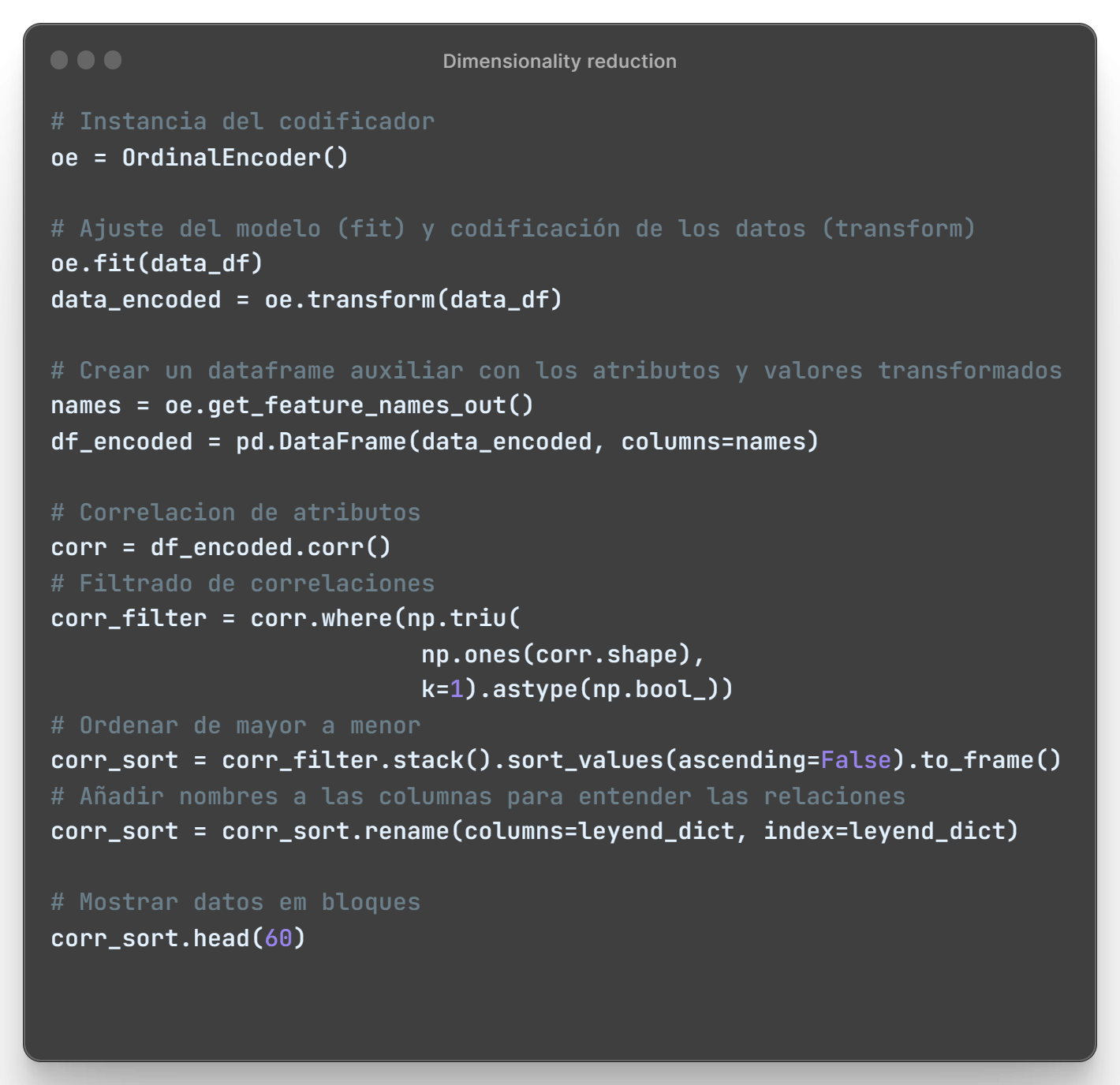
Dado que el número de columnas es elevado y alguna de estas no aporta información al caso de estudio, se realiza primero la selección de los atributos de estudio dejando así un Dataframe que sea más reducido en cuanto a sus atributos de partida para poder trabajar el análisis de datos con mayor facilidad y relación al modelo necesario.

Como primer estudio para ver la relación de los datos y tomar la decisión de los atributos a seleccionar, se comprueba la correlación de los atributos. Esto es, la dependencia que tiene una columna con respecto a las otras si estas varían.

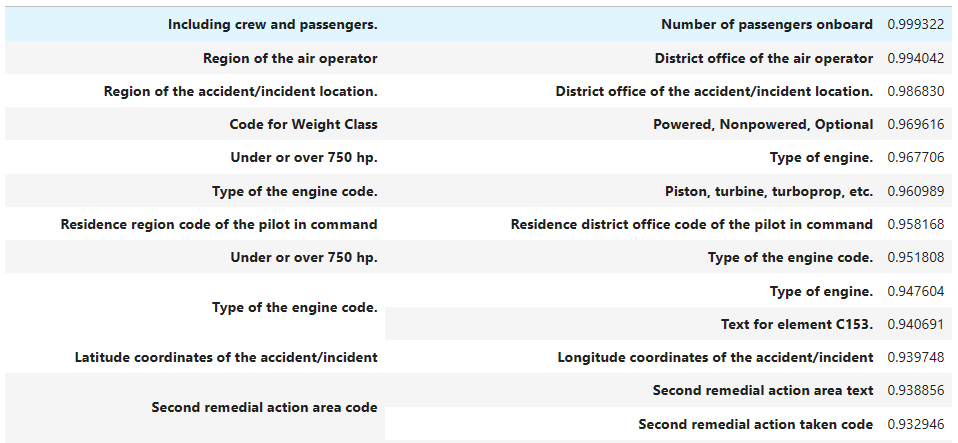
Para este estudio, se emplea el método *corr* de Pandas. Pero, previamente se requiere codificar las variables no numéricas ya que esta función, únicamente actúa sobre las columnas numéricas, y es este momento, se quiere ver la relación de todas las columnas. Para ello, se hace uso del módulo *Preprocessing* de **scikit-learn**.

Con este módulo se puede hacer uso de métodos que permite codificar los atributos categóricos a numéricos. Se utiliza el encoder **OrdinalEncoder** que transforma el valor en el ordinal para la columna, dando como resultado valores numéricos de 0 a n-1 característica.

Para poder representar los datos, se realiza un Dataframe. Este Dataframe se ajusta a partir de los datos obtenidos de la matriz de correlaciones obtenida con el método *corr*. Concretamente, se elimina la diagonal principal, por ser todos sus datos de correlación igual a 1, así como los datos por debajo de esta al ser duplicados de los de arriba de esta diagonal.



Script 7 Reducción de dimensionalidad



2 Ejemplo de correlaciones

Se representa los 180 primeros datos del Dataframe para visualizar las correlaciones. Analizando los resultados, se observa que las primeras 60 correlaciones, hace referencia a atributos que no tienen relación con el caso de estudios, estos datos hacen referencia a número de personas muertas o heridas o incluso el número de tripulantes y pasajeros. Estas fuertes correlaciones se entienden lógicas ya que, frente a un incidente o accidente, no existe discrepancia en el tipo de persona del avión.

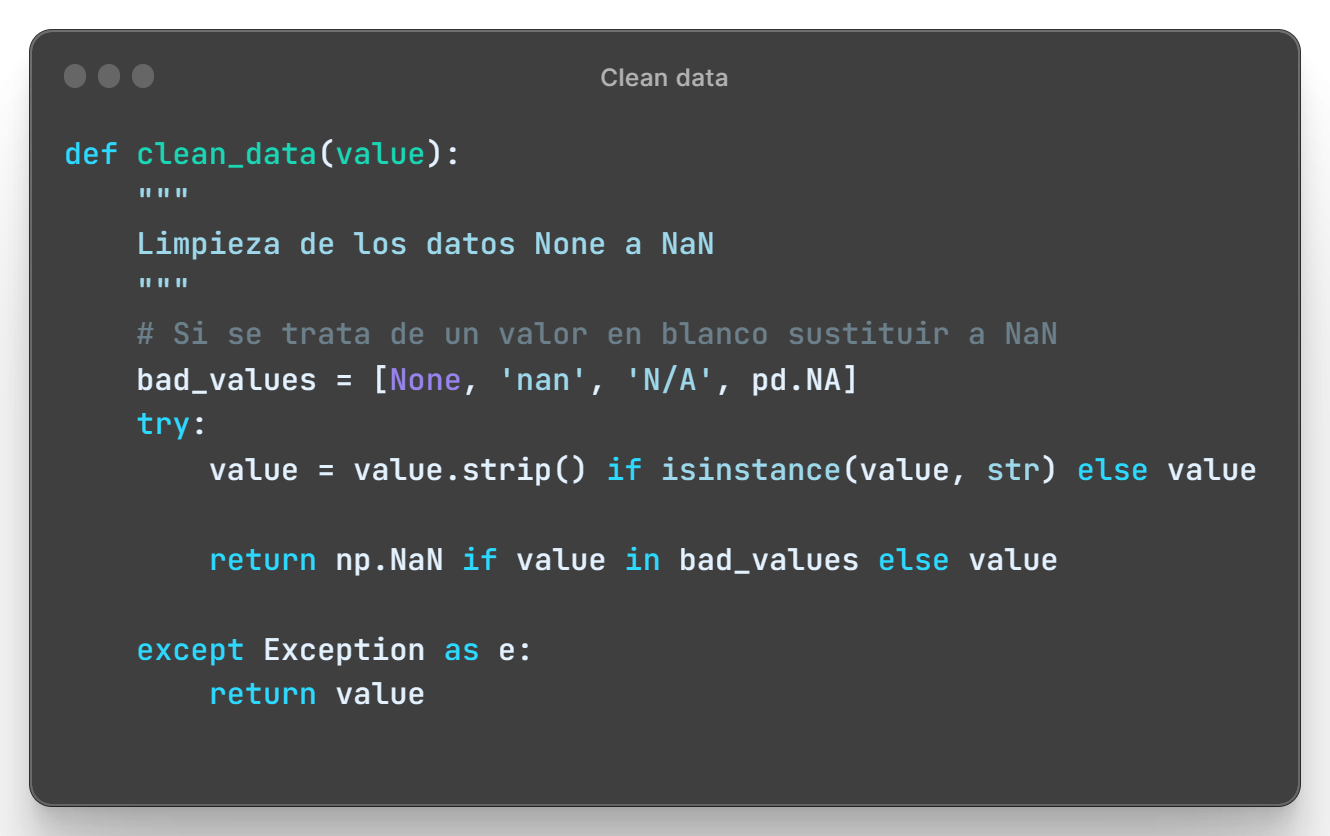
En las siguientes correlaciones, se aporta un mayor número de relaciones interesantes para el estudio del modelo. Se puede extraer que las **Primary flying condition** tienen una importante correlación con el resultado del suceso. Además, los atributos climáticos como **velocidad y dirección del viento** toman relevancia.

También se observa que el tipo de vuelo y la calificación del piloto tienen cierta relación. Esto se considera interesante ya que se entiende de la calificación/experiencia de un piloto puede ser determinante en determinadas situaciones climáticas.

Teniendo en cuenta las correlaciones y el modelo de datos necesario para la predicción, se selecciona las columnas en base a:

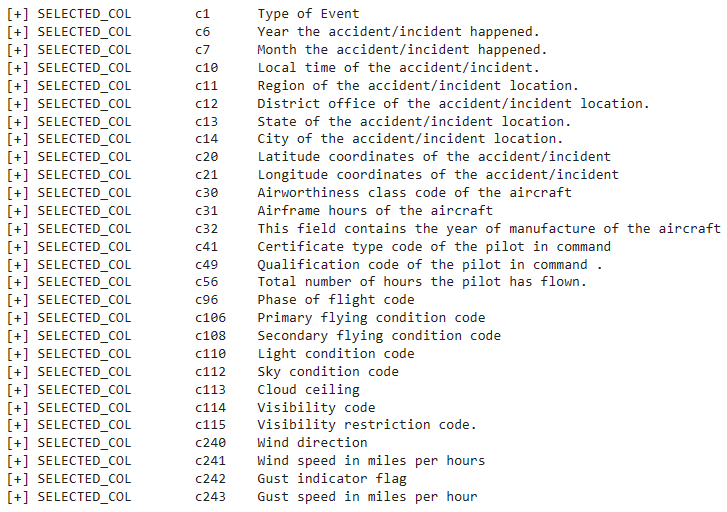
* Condiciones atmosféricas como: visibilidad, viento, altura de las nubes.
* Condiciones básicas del avión como: aeronavegabilidad del avión, horas de vuelo o antigüedad del avión.
* Aptitudes del piloto como: horas de vuelo o tipo de piloto.
* Condiciones primarias de vuelo, fase de vuelo.

Para crear el Dataframe del modelo, se emplea una función que permite realizar limpieza de los datos de cada atributo que se escoge. Esto se debe a que en la conversión del fichero *“parquet”* se definen algunos valores en diferentes formatos, pero todos ellos, datos nulos. Para ello, se hace uso de la función *clean\_data* *(Script 8 Limpieza de datos).*



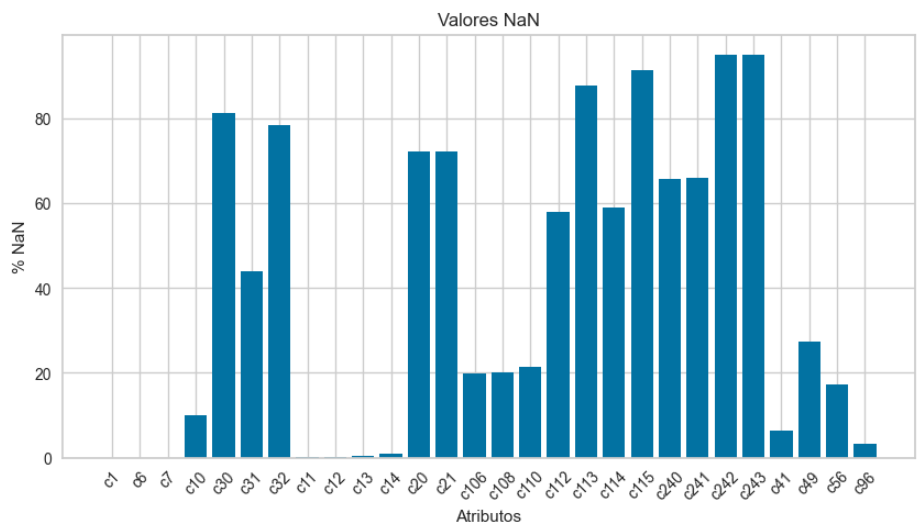
Script 8 Limpieza de datos

Para generar el Dataframe del modelo, se parte del global y se elimina de este aquellas columnas que no están entre las candidatas. Finalmente, se obtiene un conjunto de datos de 28 atributos.



3 Columnas seleccionadas

Con el Dataset que será el conjunto de datos de estudio. Se realiza una serie de analisis



#### Data Preprocessing (DP)

#### Feature Engineering (FE)

#### Data Modeling

### Crew

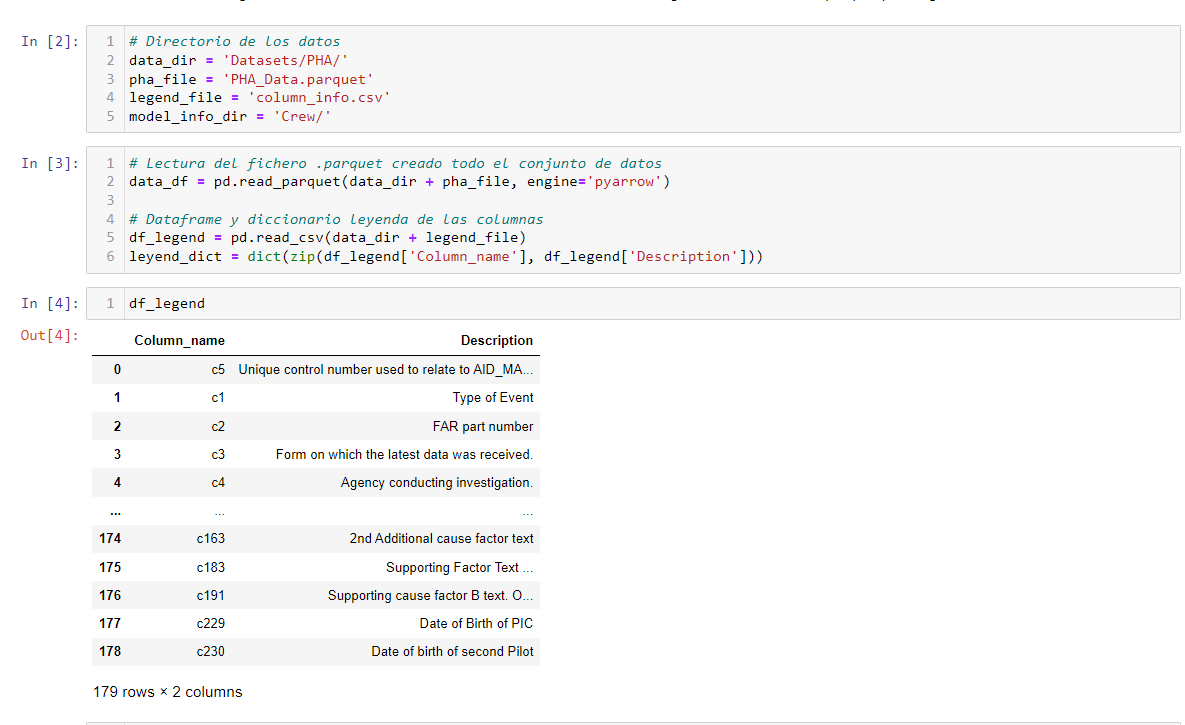
#### Obtención de datos

En primer lugar, se obtienen los datos y se carga el modelo.

Para el estudio de los datos, se ha accedido a los datos publicos de la

<https://av-info.faa.gov/dd_sublevel.asp?Folder=%5CAID> Administracion

Federal de la Aviación que recogen los sucesos ocurridos desde 1975 hasta la actualidad.



#### Exploratory data Analysis (EDA)

**Reducción de dimensionalidad y *multicolinealidad:***

Dado que el número de columnas es elevado y alguna de estas no aporta información a nuestro caso de estudio, se realiza primero la selección de los atributos de estudio dejando así un dataframe que sea más reducido en cuanto a sus atributos de partida para poder trabajar el análisis de datos con mayor facilidad.

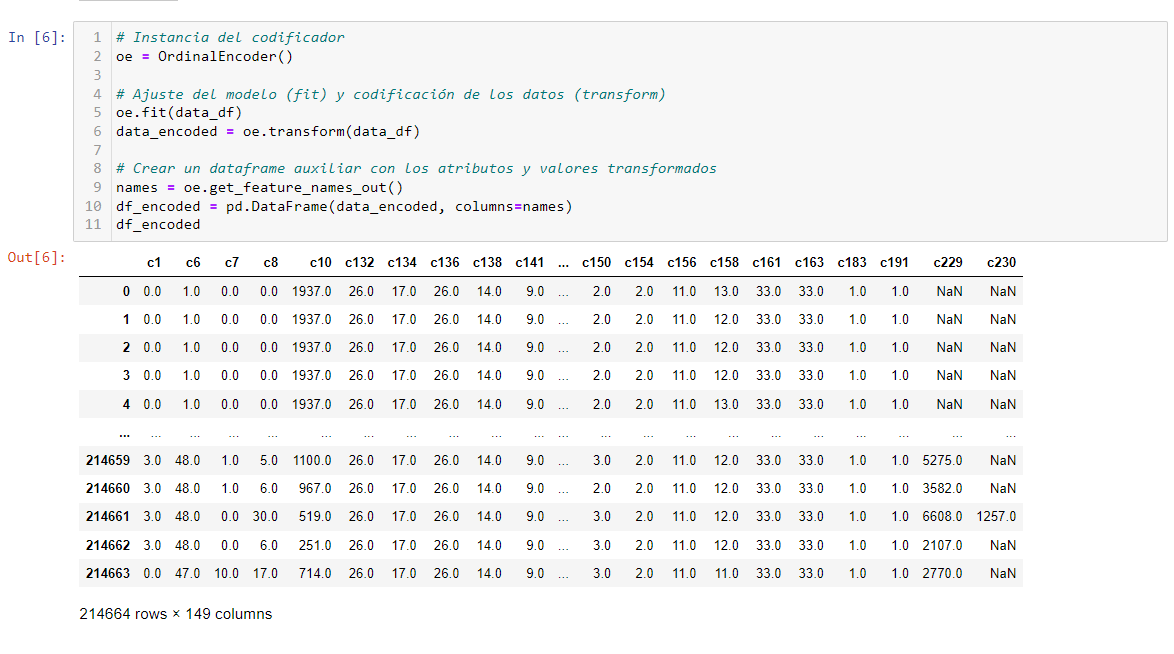
Si las variables predictoras tienen una alta correlación entre sí, puede haber problemas de multicolinealidad. La multicolinealidad ocurre cuando hay una fuerte relación lineal entre las variables independientes, lo que puede dificultar la interpretación de los coeficientes del modelo y conducir a estimaciones inestables. En tales casos, puede ser deseable eliminar una de las variables correlacionadas o combinarlas para crear una nueva variable

También es importante la interpretación y simplicidad; una alta correlación entre las variables predictoras puede dificultar la interpretación de los resultados. En ese caso, puede ser preferible trabajar con variables menos correlacionadas para facilitar la comprensión de cómo cada una afecta a la variable objetivo.

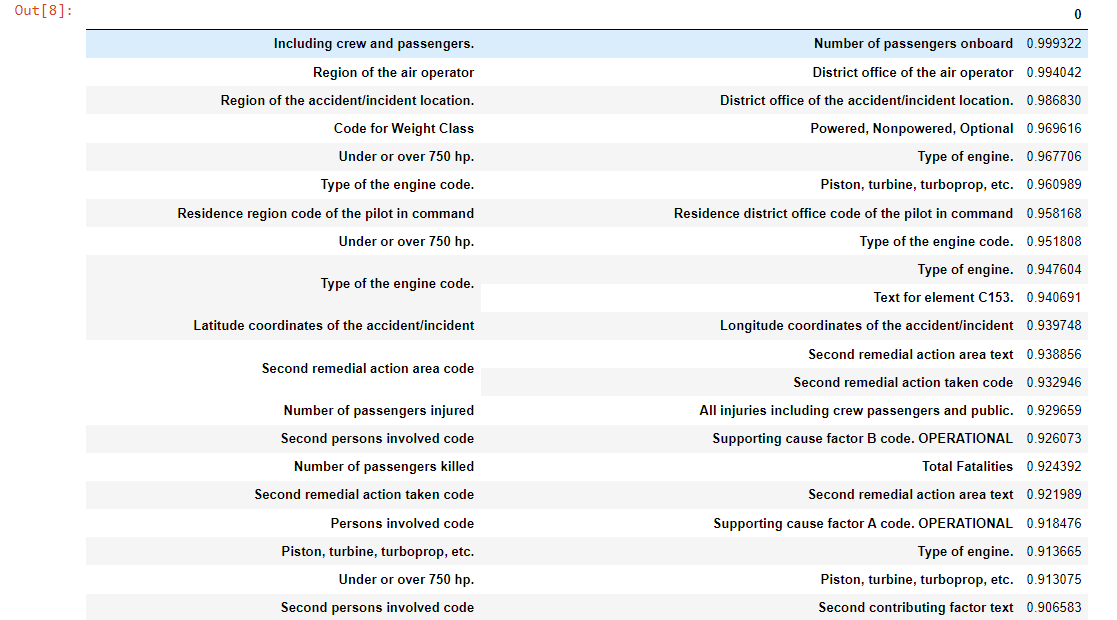
Como primer estudio para ver la relación de los datos y ayudarnos a tomar la decisión de los atributos a seleccionar, se observa la relación de las columnas de forma gráfica. Esto es, la dependencia que tiene una columna con respecto a las otras si estas varían.

Para este estudio, se emplea el método corr de Pandas. Pero, previamente se requiere codificar las variables no numéricas ya que esta función, únicamente actúa sobre las columnas numéricas, y es este momento, queremos ver la relación de todas las columnas. Para ello, se hace uso del módulo preprocessing de scikit-learn. Con este módulo se puede hacer uso de métodos que nos permitirá codificar los atributos categóricas a numéricos.

Se utiliza el encoder OrdinalEncoder que transforma el valor en el ordinal para la columna, dando como resultado valores numéricos de 0 a n-1 característica.



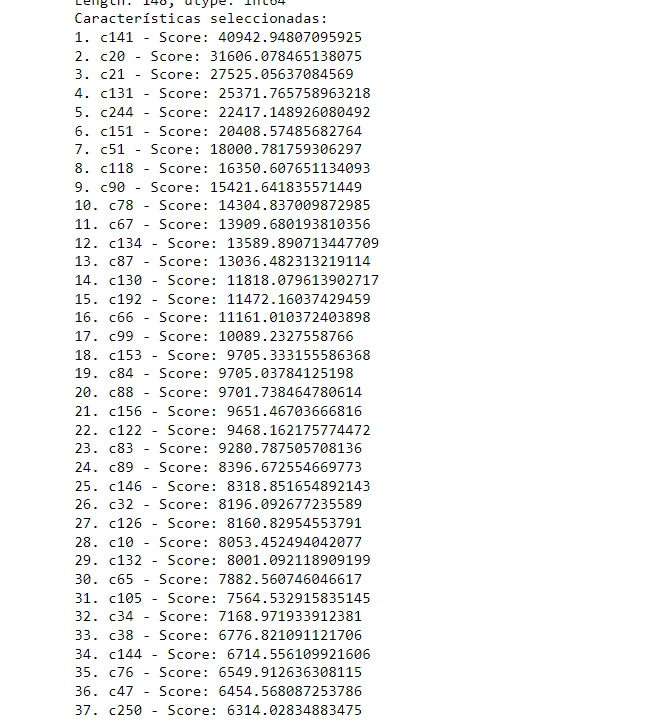




Si todas las variables predictoras están altamente correlacionadas entre sí pero también tienen una fuerte correlación con la variable objetivo, esto puede ser beneficioso. Significaría que todas las variables están capturando información relevante para predecir la variable objetivo. Sin embargo, si algunas variables no están correlacionadas con la variable objetivo pero sí entre sí, es posible que esas variables no estén aportando información adicional al modelo y podrían eliminarse.



Como se puede apreciar hay variables con alta coorelación con la variable objetivo por lo que podemos tener una idea de que variables son potencialmente interesantes.



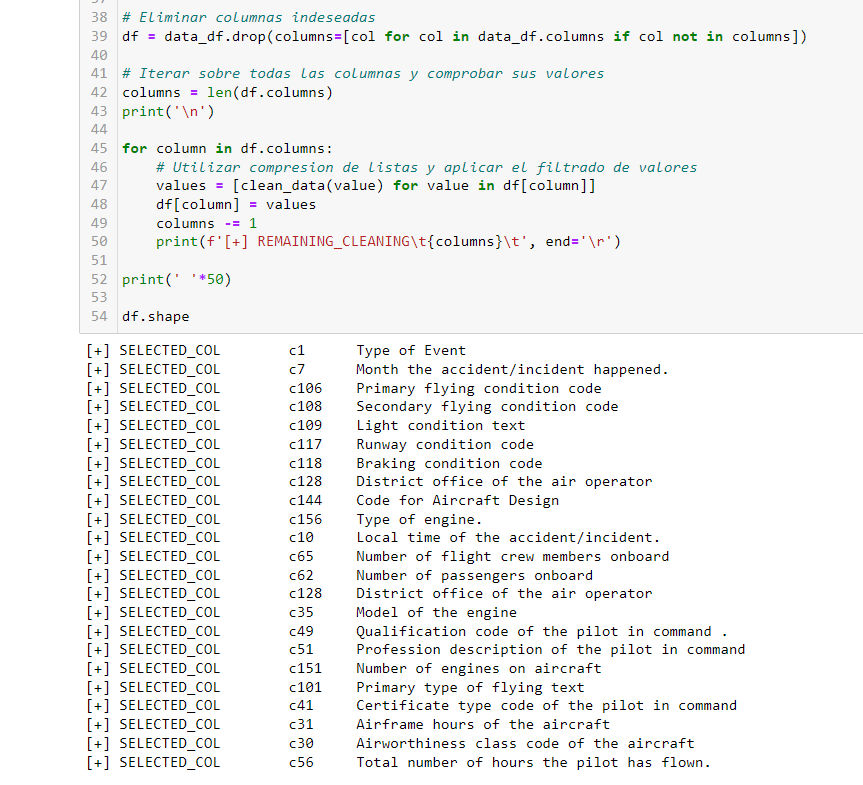
**Selección de atributos**

Si las variables están altamente correlacionadas y una de ellas falta o se modifica, es más probable que otras variables correlacionadas puedan proporcionar información similar. Esto puede hacer que tu modelo sea más robusto y estable en presencia de cambios o ruido en los datos.

De estas 50 variables con alta relación con la variable objetivo escogeremos teniendo en cuenta que queremos realizar una predicción en relación con los datos de la tripulación, pasajeros y algunas características adicionales relacionadas.

Esta lista de variables con gran correlación sobre la variable objetivo debemos filtrar las variables que tienen una alta correlación entre sí pero no con la variable objetivo ya que como hemos mencionado anteriormente pueden ser redundantes y no aporta ningún beneficio extra.

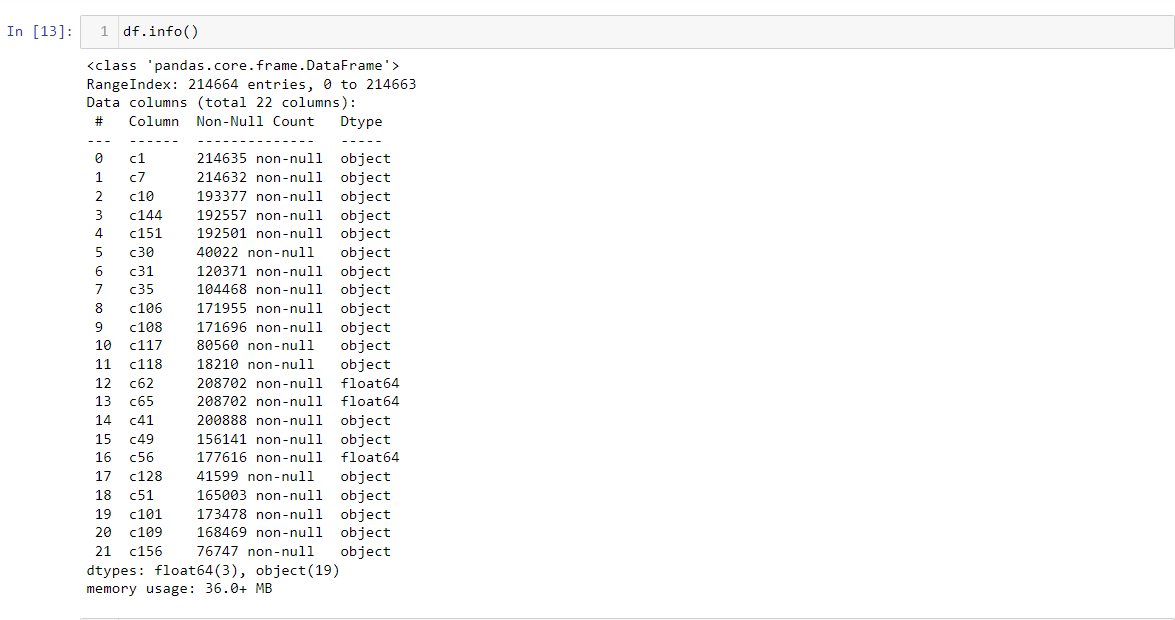




**Análisis conjunto de datos**

Para tener un contexto de los datos que se tiene, se realiza una serie de visualización de datos.

La primera de estas acciones consiste en describir los datos que tenemos, en su tipo y cuántos de estos datos son válidos, o al menos, no desconocidos (NaN).

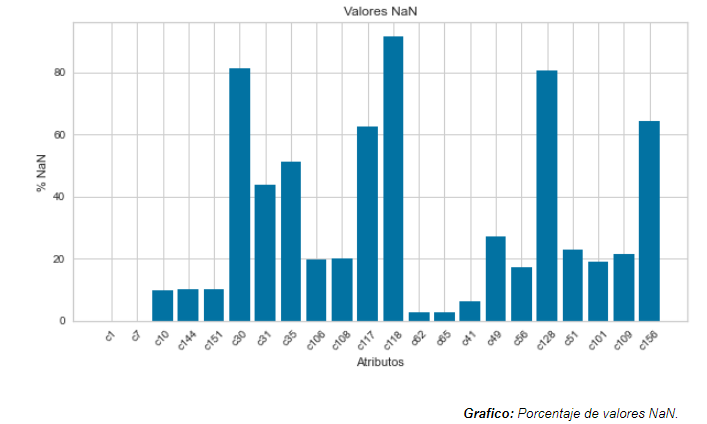




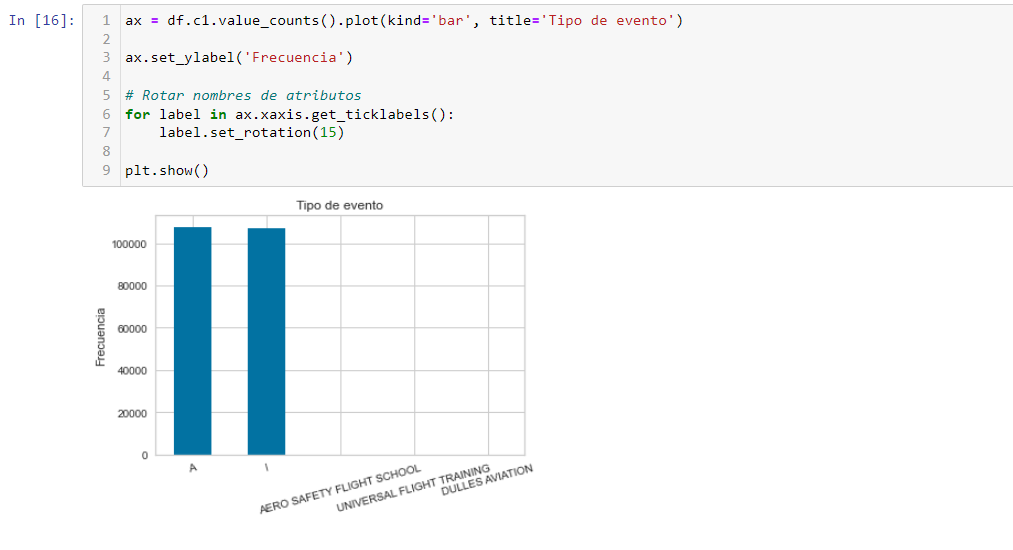
De forma gráfica, y desde el concepto de porcentaje de valores NaN en las columnas, se muestra el siguiente gráfico.



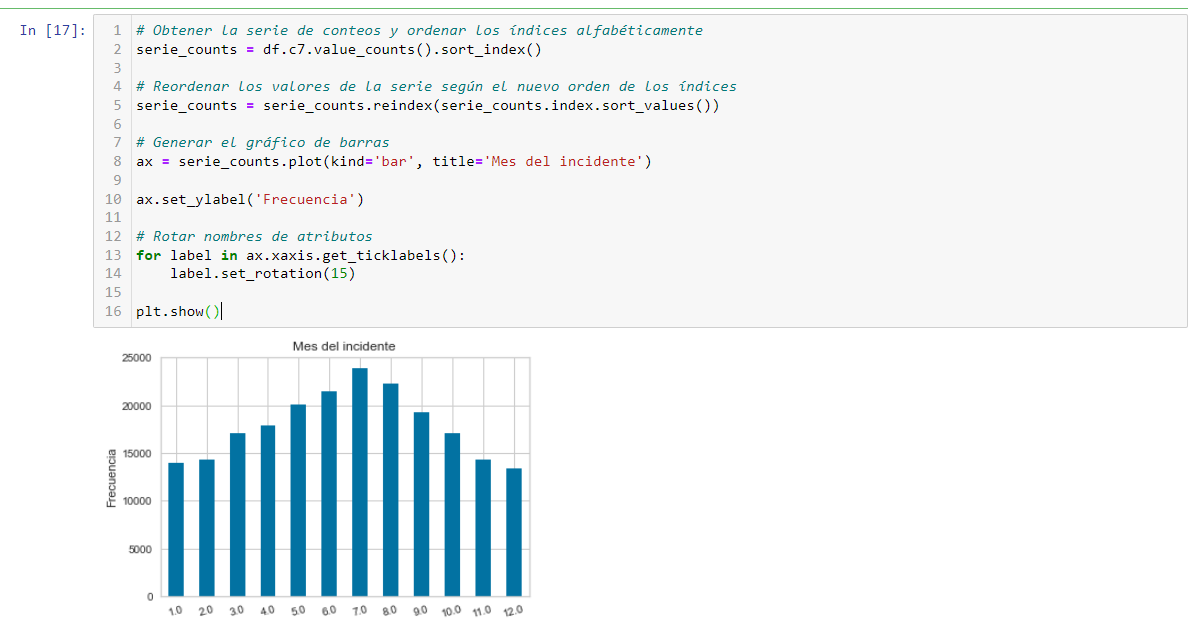
Se puede observar que en algunas columnas hay bastantes datos faltantes.



El modelo que se desea realizar, debe de ser capaz de predecir el tipo de incidente que ocurrirá. Para ver si tenemos unos datos balanceados, a continuación se muestra una gráfica de la predicción a realizar, donde podemos observar que si existe un balance de datos.

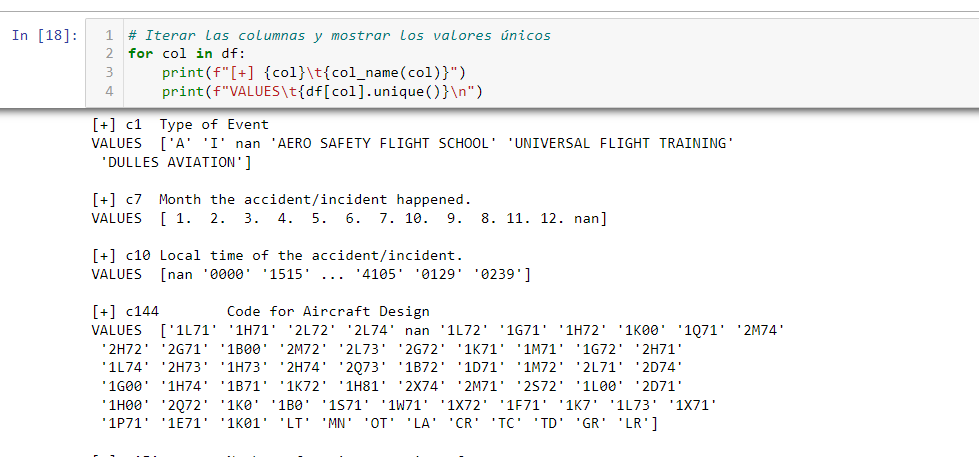


A continuación se muestra el balanceo de datos para la columna del mes del incidente. Teniendo en cuenta el tipo de análisis a realizar, se realiza una previa visualización de la distribución de los sucesos por esta característica.



**Descripción de valores**

Dado el elevado número de datos faltantes y sabiendo que existe valores que están en columnas diferentes por el estudio realizado para obtener el dataset de trabajo. A continuación se detalla los valores que existe para cada uno de los atributos del modelo de datos obtenido.



Como se puede observar, existe la mitad de columnas que contiene más del 50% de datos nulos o vacíos que habrá que tener en cuenta a la hora de la selección de caracteristicas para nuestro estudio. Además de valores atípicos para su columna por encontrarse desplazados.

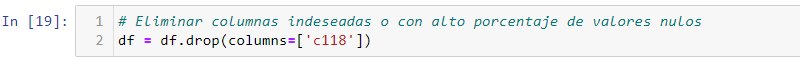
Se observa que las clases de estudio (Accidente o Incidente) están balanceadas

Por último, analizando el tipo de datos para el dataframe, se observa que la gran mayoría de atributos hace referencia a tipo objeto. Se debe a que determinados de estos atributos, tiene datos categóricos que mezcla números y letras en sus datos. No obstante, se analiza más adelante cuál de estas son realmente este tipo de columnas y cuáles son fruto de un error en los datos.

#### Data Preprocessing(DP)

Con las conclusiones extraídas en el apartado anterior, en este apartado se realiza las correcciones necesarias para corregir estos errores.

**Eliminar características**

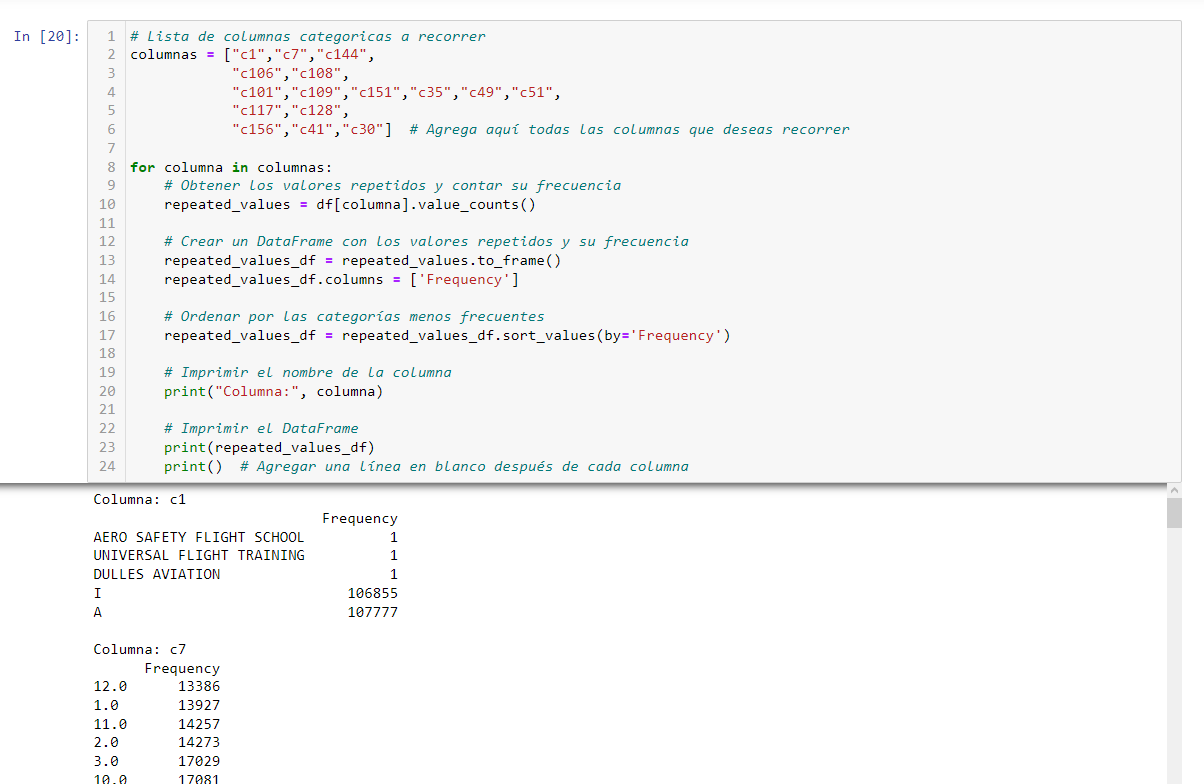
La columna c118 tiene demasiados valores nulos, por lo que una imputación de valores podría afectar negativamente a la predicción. 

**Valores desplazados o atipicos**

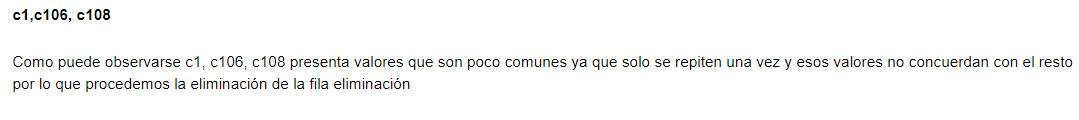
A continuación, se define los valores que ha de tener cada una de las columnas categóricas para encontrar en que otras columnas puede existir y asi poder relocalizarlas. Esta busqueda se realiza sobre todo el dataset, de este modo, se podrá importar esos valores a las columnas del dataset de estudio.

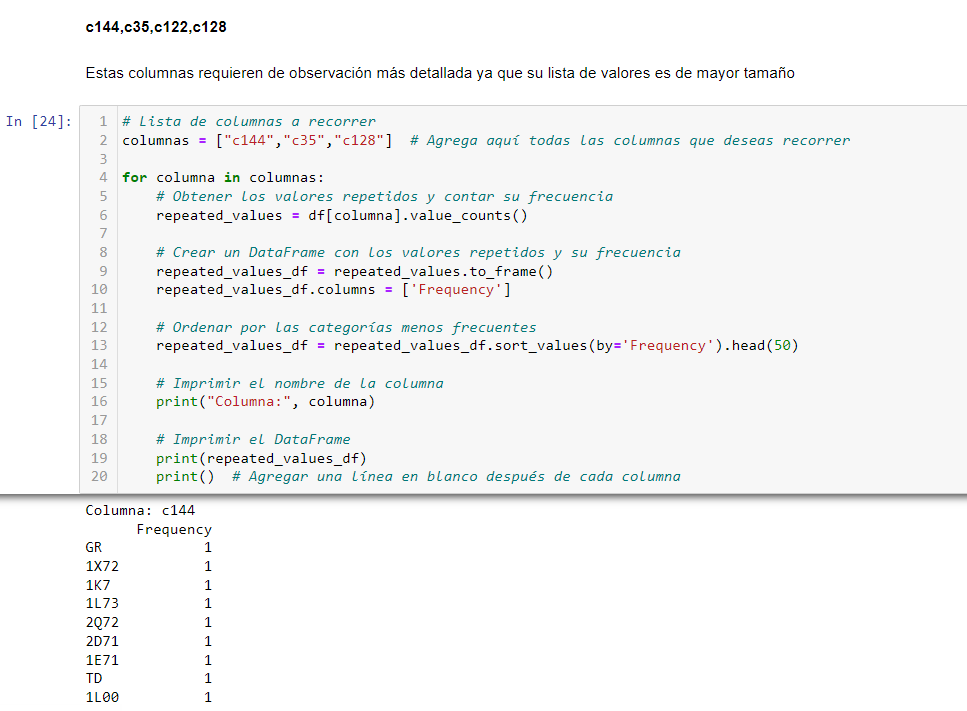
**Análisis**

Vamos a contar los valores de las columnas que a priori parecen categóricas para poder identificar algunos valores que no deberían estar en determinadas columnas debido a un desplazamiento, generando un valor atípico.



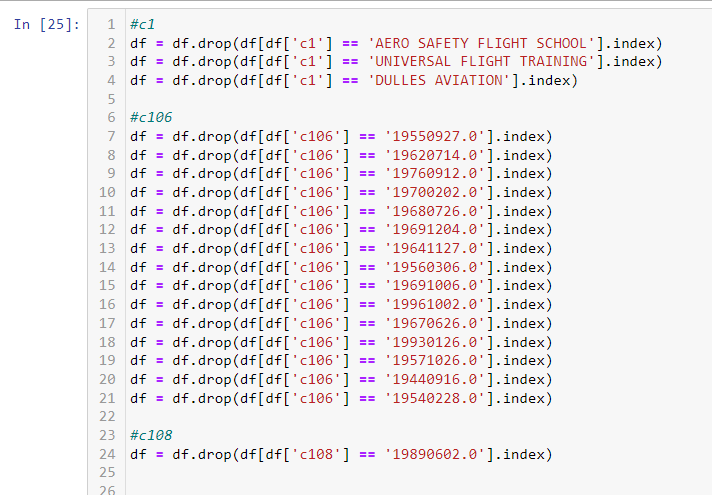






Tras ver los resultados, se puede apreciar que estas columnas no presentan ninguna alteración.

**Acciones**

Tras poderse comprobar que c1, c106 y c108 presentan valores atípicos se procede a borrar las filas que alteran la variable ya que son pocos registros los afectados y no alteraría la predicción. 

**Variables numéricas.**

Tras la comprobación no aparenta tener datos atípicos o registros desplazados



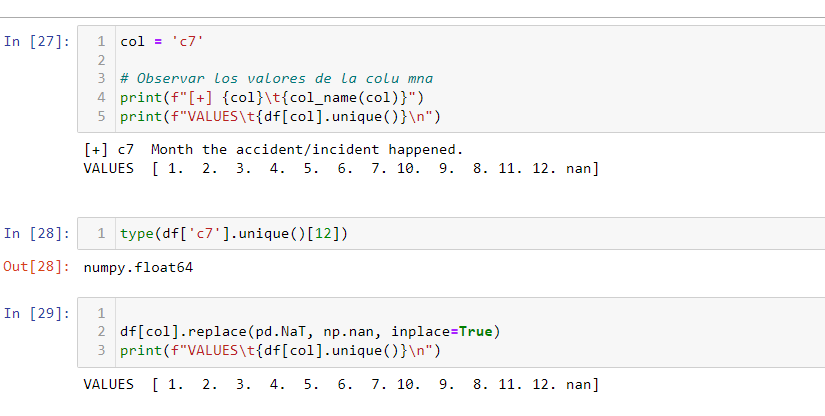
**Formateo de datos**

Ya tenemos comprobado el desplazamiento y valores atipicos, por lo que el siguiente paso sería la verificación y adecuación de los datos para que pueda ser interpretado correctamente por el algoritmo de predicción.

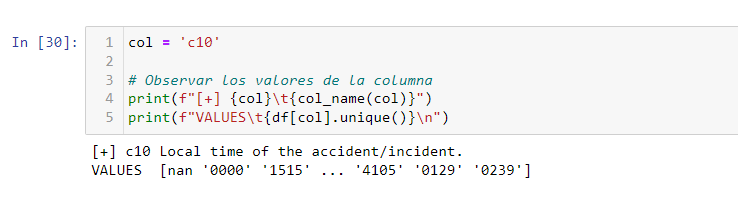
**\*NOTA**

Los datos aquí indicados como categóricos o numéricos, no son necesariamente de dicha índole para el estudio del modelo, esta agrupación se ha realizado por tipo de datos contenido pero no por significado para el modelo final

*c7 Month the accident/incident happened*

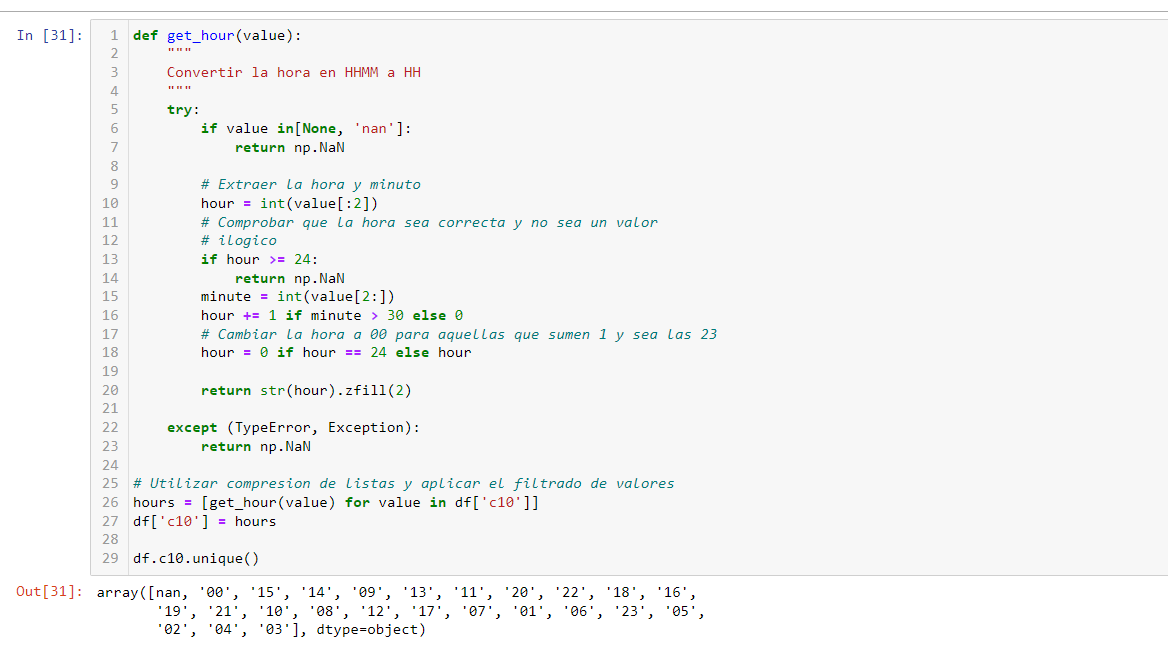


*c10 Local time of the accident/incident*

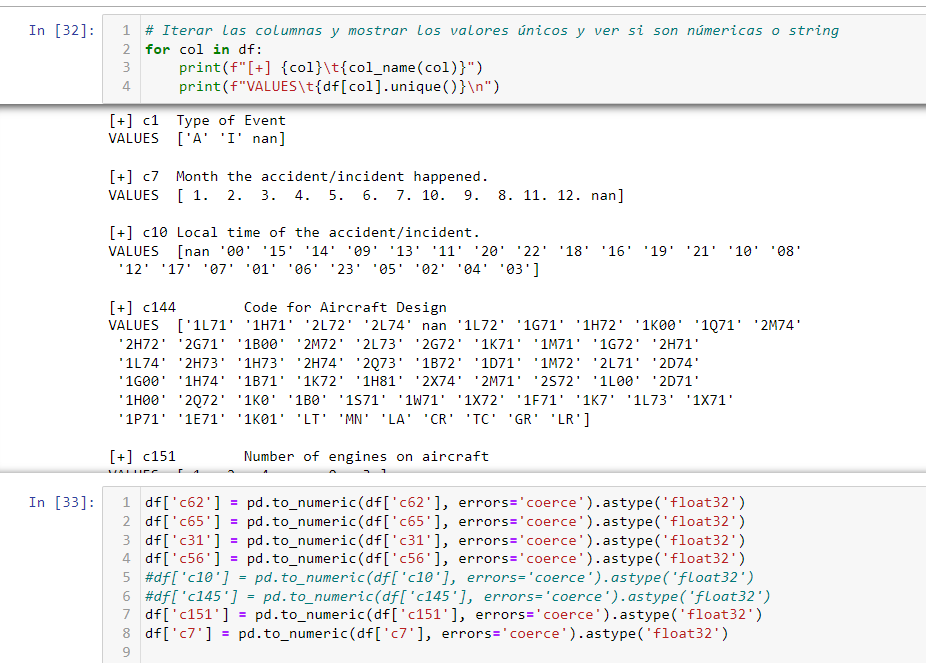


La columna c10 contiene los datos de la hora del incidente, pero para poder lidiar con todos los datos, se convierte los valores dados en formato HHMM en HH. Para ello, se recorre todos los valores y se modifica su valor, además, si los minutos son más de 30, se sumará una hora al resultado.

Se toma esta decisión para no sobrepasar al modelo de características para este atributo.

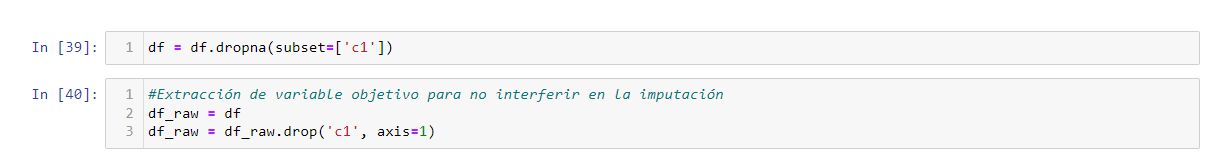


**Conversión de columnas mal identificadas como string a numéricas.**

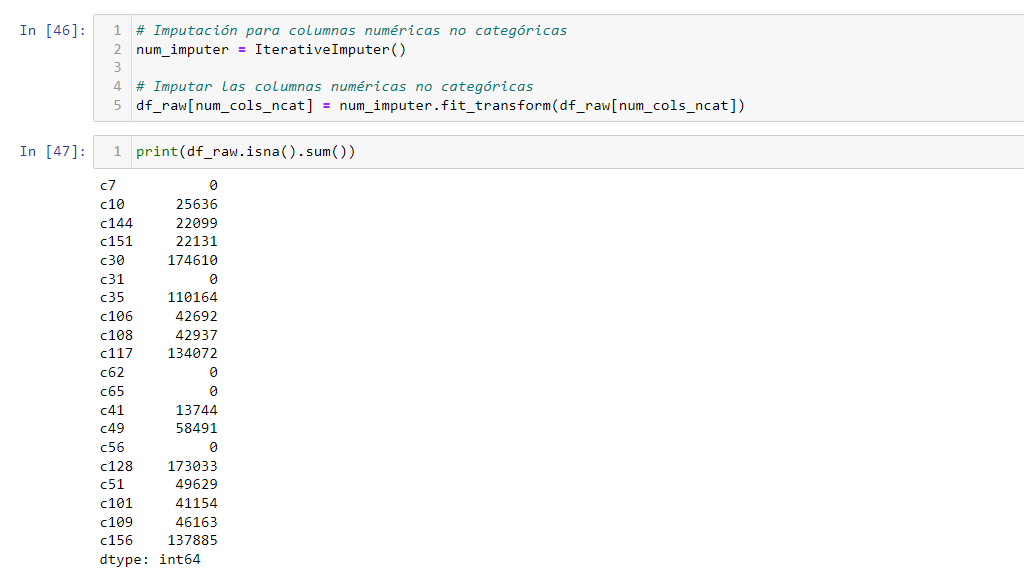


**Imputación de valores**

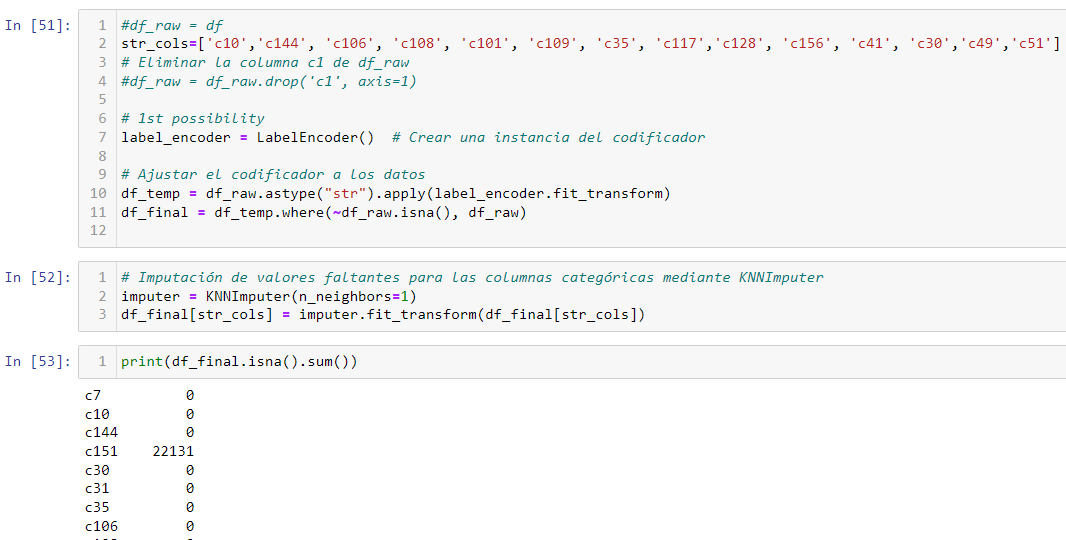
Primeramente debe de aislarse la variable objetivo de los datos a imputar ya que puede perjudicar gravemente a la predicción ya que los datos imputados pueden generar una relación muy fuerte con la variable generando ruido que el algoritmo de predicción puede mal interpretar.



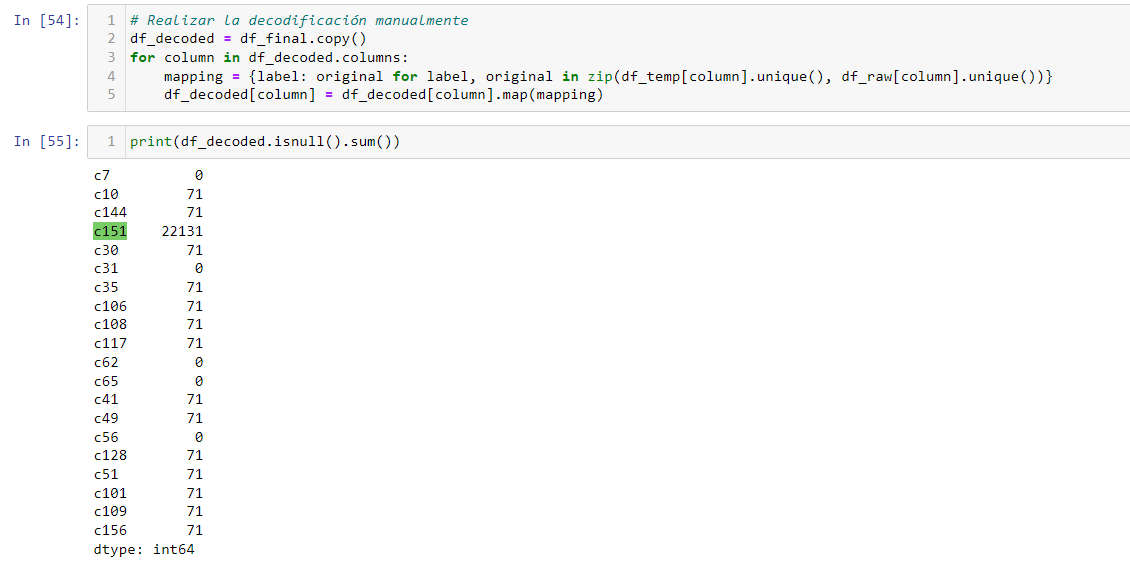
Imputamos con IterativeImputer de valores de columnas numéricas



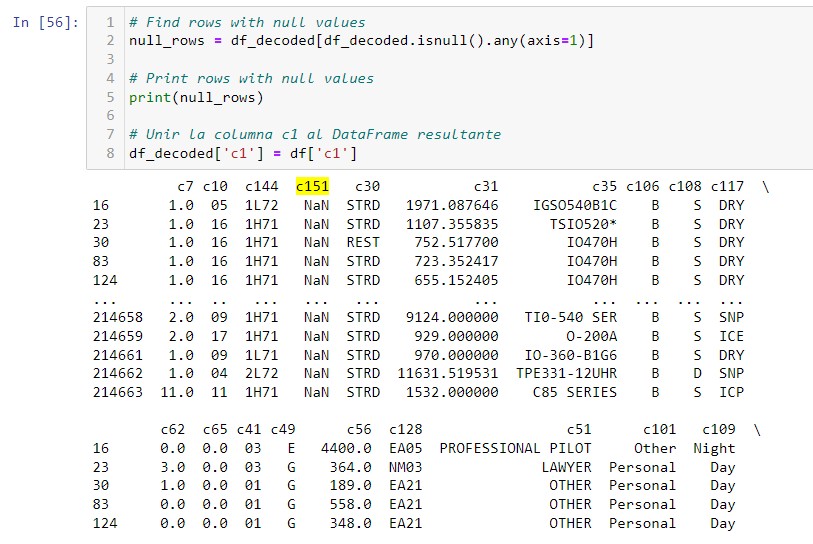
Ahora vamos a imputar las columnas tipo string para ello utilizaremos KnnImputer y codificaremos los datos mediante LabelEncoder, almacenando los valores originales para poder decodificarlos tras la imputación.



Ahora tras la imputación, realizamos la decodificación

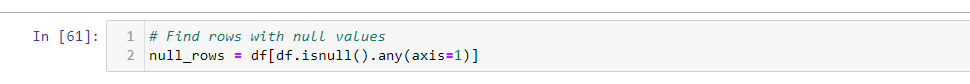


Los valores que no han podido imputarse se eliminarán dada la baja repercusión que tendrían sobre el dataset. Además al dataset imputado se le debe de volver a unir la variable objetivo.



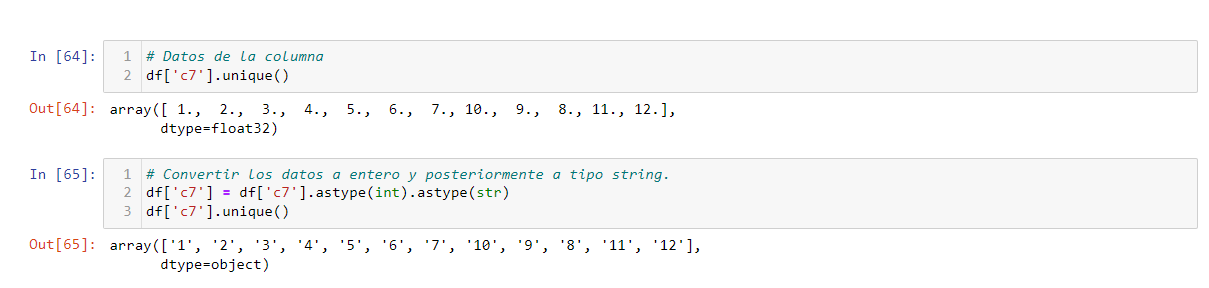


Tras la unión c1 tendrá registros que no se han borrado y que deben borrarse para que no tenga valores NaN.



**Conversión de datos**

Antes de continuar después de la imputación debemos de formatear la columna c7 como string.



#### Feature Engineering (FE)

A continuación, se realiza una serie de modificaciones en los datos para que estos puedan ser interpretados por el modelo.

**Codificación de datos categóricos**

Existe una serie de columnas con datos categóricos que se deberá de codificar para poder tratar sus datos y realizar nuestra predicción del modelo.

Hasta ahora se ha trabajado con las columnas categóricas y numéricas en base a sus datos, pero, para realizar la codificación necesaria, se va a realizar sobre todas las columnas que necesariamente serán categóricas para realizar el modelo.

Existen tres técnicas de Data Encoding dependiendo de las características de los atributos:

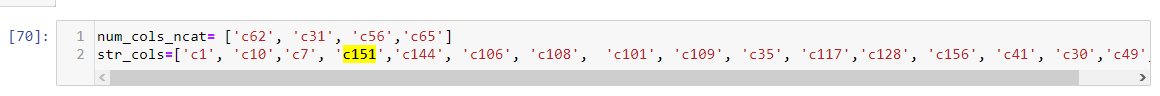
-OrdinalEncoder: Se utiliza para codificar variables categóricas que no tienen una relación de orden natural. Devuelve una única columna que va de 1 a n\_caracteristicas-1.

-LabelEncoder: Se utiliza para codificar variables categóricas que tienen una relación de orden natural. Devuelve una columna de 1 a n\_caracteristicas-1. Su uso se realiza principalmente para transformar la variable dependiente.

-OneHotEncoder: Se utiliza para codificar variables categóricas que no tienen una relación de orden natural. Devuelve una columna para cada atributo codificado, indicando 0 o 1 según la presencia de la característica para cada registro.

A continuación se muestra los tipos de datos categóricos del dataset para seleccionar que técnica emplear con cada atributo.

Para la variable dependiente u objetivo, c1, así como para los atributos de entrada cuyos datos tienen un orden de relación, se emplea LabelEncoder, col\_le. Para los atributos cuyas clases no están relacionadas, se emplea OrdinalEncoder. Para el resto de columnas se ha realizado un OrdinalEncoder, col\_oe.





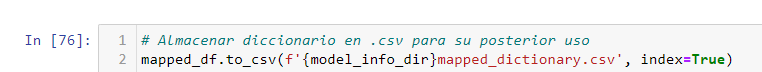
**LabelEncoder**

Para la transformación de los datos, se guarda el diccionario de los datos que se va a mapear para poder emplearlo en la entrada de nuevos datos.



**OrdinalEncoder**





**Normalización atributos numéricos**

En base a la desviación estandar de determinadas columnas, que son elevadas, se realiza el normalizado de los datos y el escalado entre 0 y 1.



#### Data Modeling

Crear el modelo de predicción con el conjunto de datos obtenido.

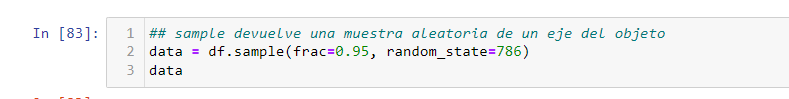
**División de los datos**

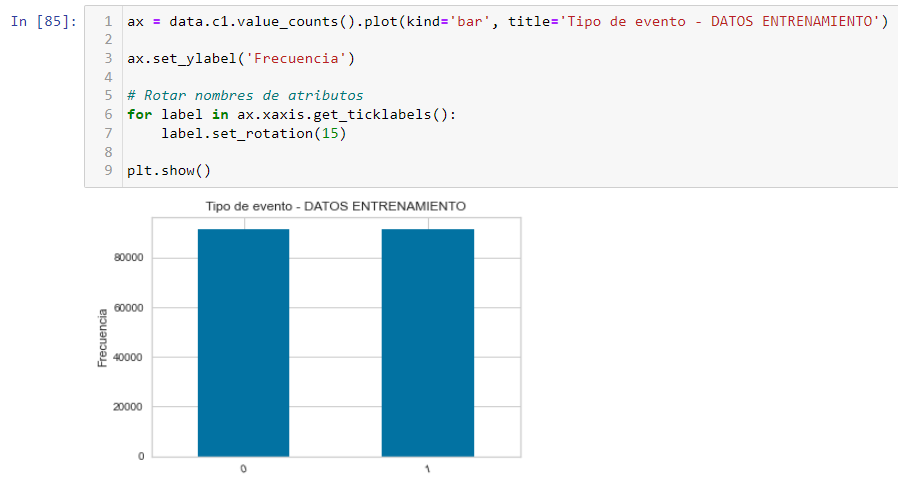
Para el modelado, se emplea unos datos de entrenamiento (train) y unos de prueba (test). Una vez el modelo está realizado, existe los datos de validación, estos no son vistos por el modelo durante el train o test.

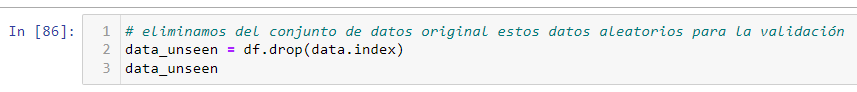
A continuación, se parte los datos en:

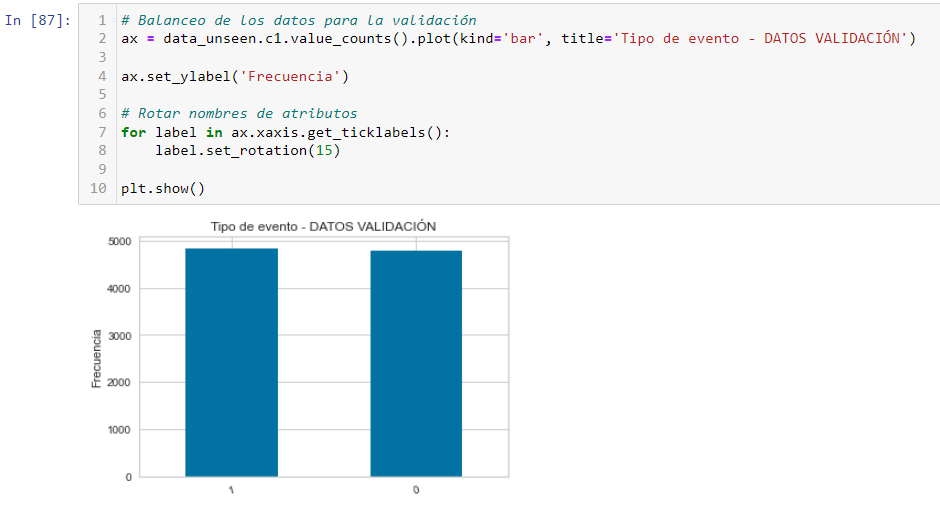
**data**: datos con los que se entrenará y testeará el modelo.

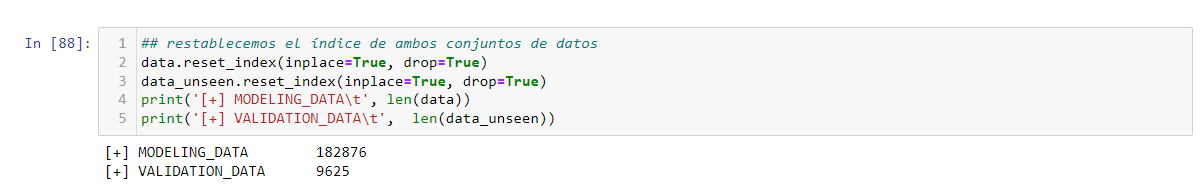
**data\_unseen**: datos que se emplearán para la validación (5% de los datos).











### Airplane(PABLO)

#### Obtención de datos

#### Exploratory Data Analysis (EDA)

#### Data Preprocessing (DP)

#### Feature Engineering (FE)

#### Data Modeling

## Generar el modelo con pycaret

### Forecast(VICTOR)

#### PyCaret SETUP

#### Comparación de modelos

#### Creación del modelo

#### Ajustar el modelo

#### Guardar el modelo

### Crew

#### PyCaret SETUP

Para la configuraciones de nuestro clasificador, se emplea el metodo setup() de PyCaret, en el cual se le indica aquellos parámetros necesarios para realizar los modelos:

**fold\_strategy**: Permite seleccionar una técnica de validación cruzada.

**preprocess**: Existe la posibilidad de introducir datos ya preprocesados y por lo tanto indicar unicamente el pipeline de transformaciones que se desea (valor a False). O por el contrario, que PyCaret preprocese los datos para realizar imputacioens, codificacion de caracteristicas... (valor a True).

**pca**: Se permite el uso de reducción de dimensionalidad con el uso de Principal Component Analysis.

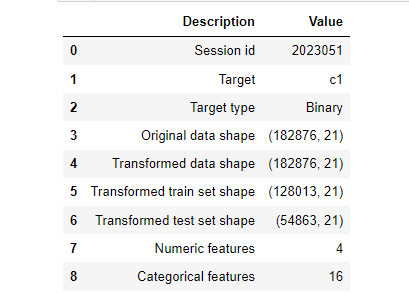
**low\_variance\_threshold:** Ignorar varianzas bajas en aquellos casos en los que existe una característica dominante sobre otras relacionadas y no existe mucha variación en la información de estas características.

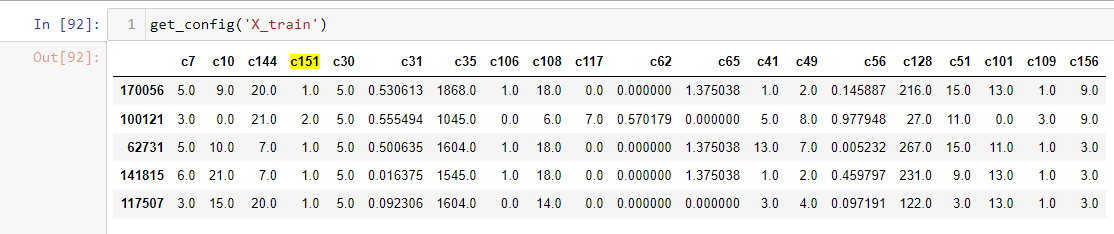
**remove\_multicollinearity**: Permite lidiar con las características que están fuertemente ligadas con otras del dataset.

**fix\_imbalance**: Cuando existe un conjunto de datos desbalanceado, se permite balancear este mediante diferentes técnicas.

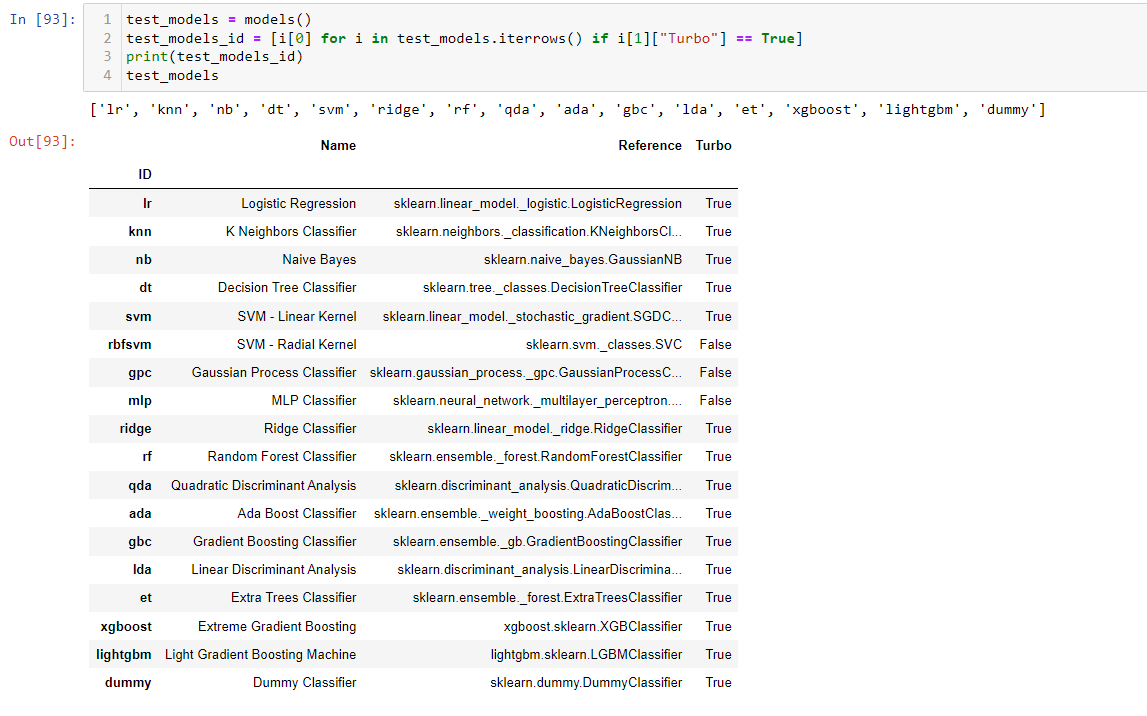
**remove\_outliers**: Eliminación de outliers en el conjunto de los datos.







#### Comparación de modelos



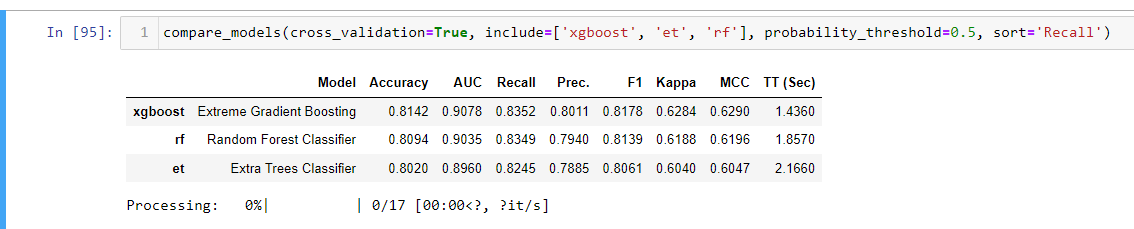
De los modelos disposnibles, se realiza una comprobación de sus predicciones ordenados por la Sensibilidad (Recall) para poder compararlos y escoger el mejor de ellos.

Dada la importancia de la predicción, es importante escoger un modelo cuya proporción de verdaderos positivos en relación con el total de muestras positivas en el conjunto de datos sea lo más elevada posible.



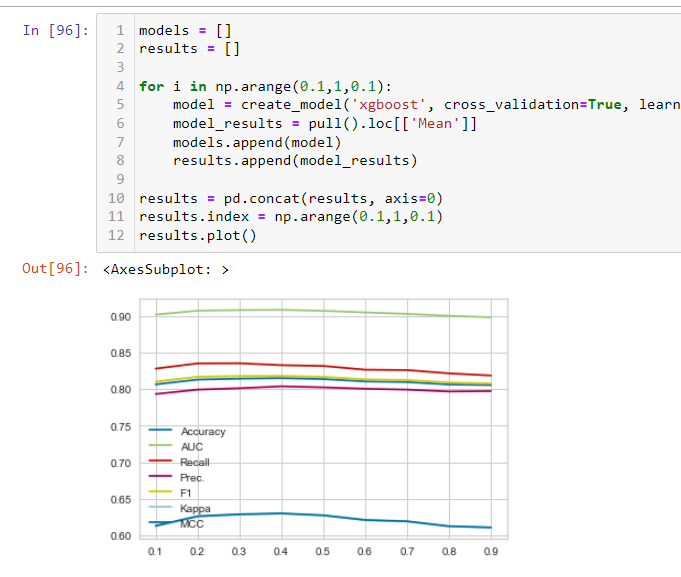
Teniendo en cuenta los resultados obtenidos, en cuanto a métricas y tiempos de ejecución, se decide escoger un top 3 para crear los modelos, poder ajustarlos y finalmente realizar el modelo con el mejor de ellos.

A continuación, se muestra la comparación de los modelos, esta vez, empleando la validación cruzada con 10 folds como se ha indicado en los ajustes.

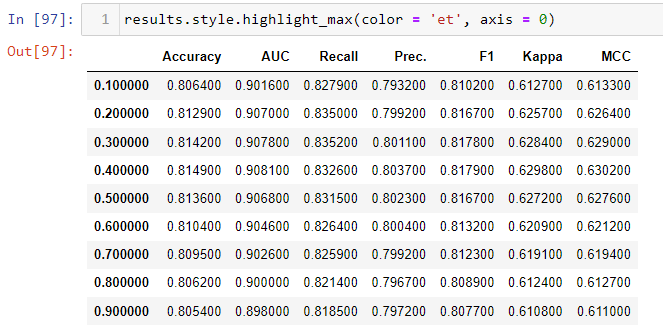


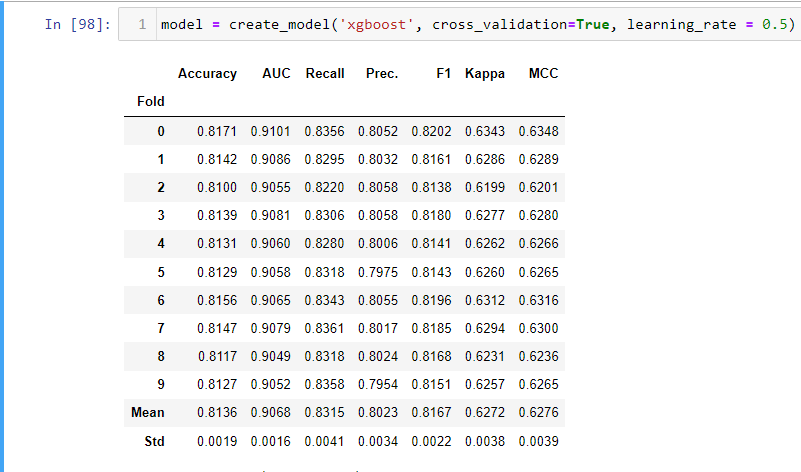
#### Creación del modelo

En base a los resultados anteriores, se escoge el modelo Extreme Gradient Boosting para realizar las predicciones.



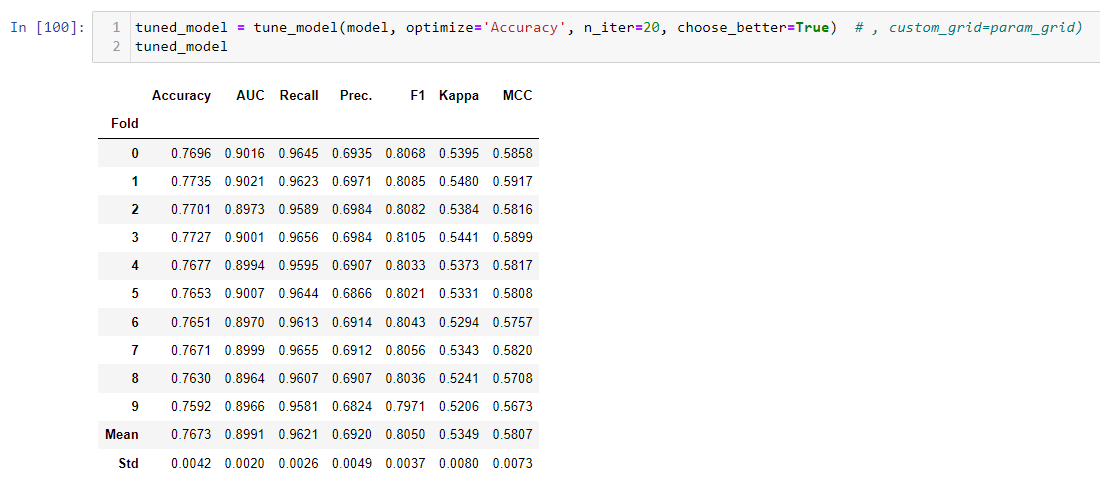
Tras la creación del modelo teniendo en cuenta su tasa de aprendizaje, se establece esta en 0.5.



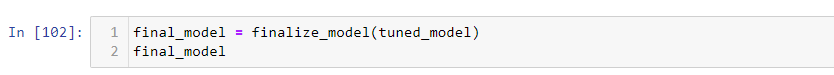


#### Ajustar el modelo

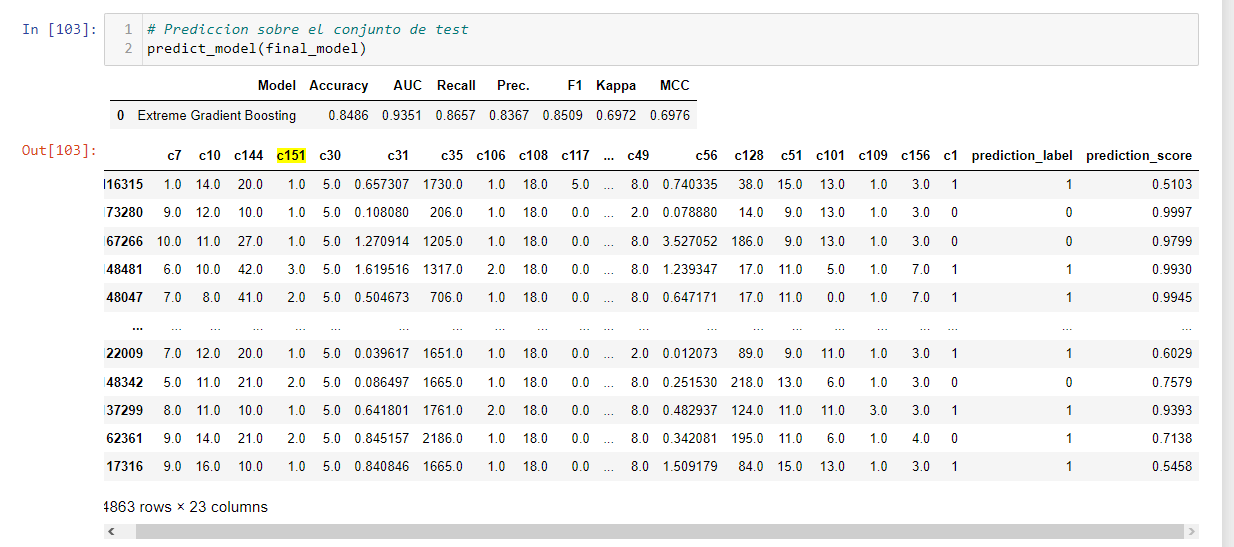


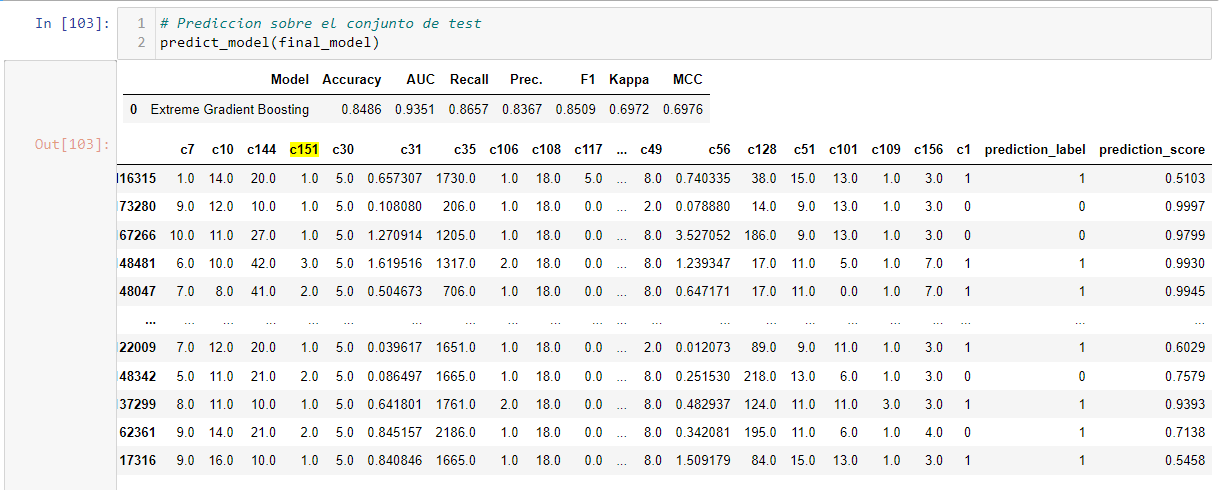


Con el mejor de los modelos seleccionado y ajustado, se ajusta el modelo al conjunto de todos los datos, incluyendo test y train. Realizando finalmente una predicción sobre el conjunto de datos que se extrajo en un principio y no se han usado durante todo el proceso de creación del modelo.



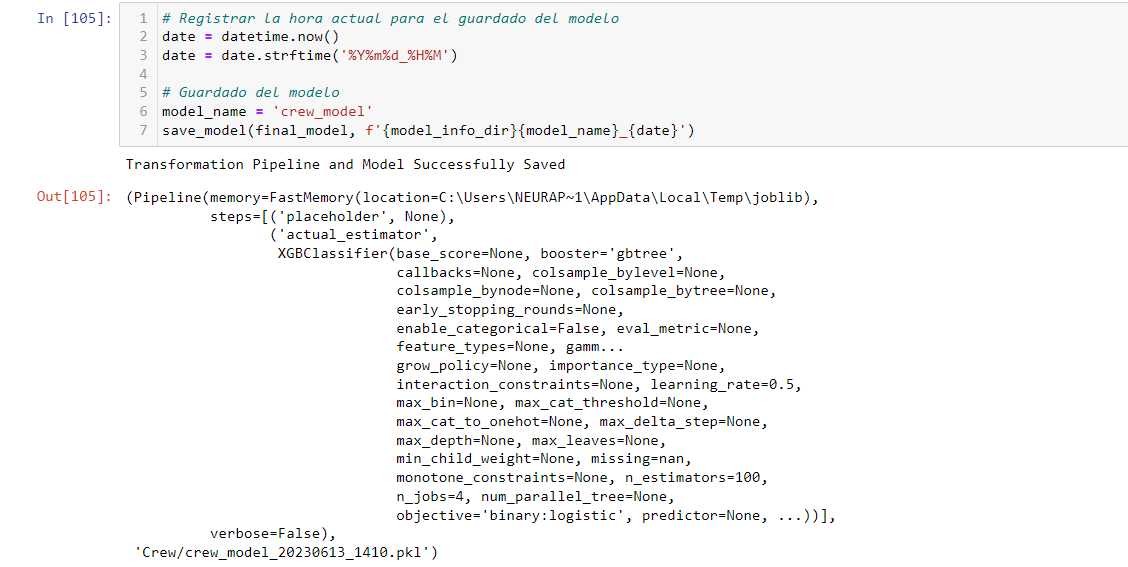
Ya con el modelo creado, pasamos a realizar las predicciones con los datos que se ha reservado para ello.



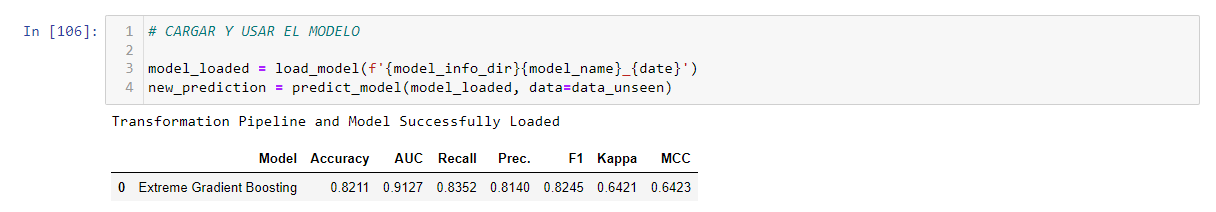


#### Guardar el modelo

Tras todo el proceso de modelado y ajuste del mismo, para poder utilizar el modelo en un entorno de producción, es necesario guardar el mismo. Para ello, se emplea las propias herramientas de PyCaret.



Ya estaría preparado para cargar y usar nuestro modelo.



### Airplane(PABLO)

#### PyCaret SETUP

#### Comparación de modelos

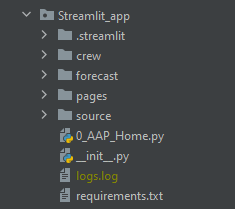
#### Creación del modelo

#### Ajustar el modelo

#### Guardar el modelo

## App de Streamlit

Para la app de Streamlit se ha generado esta estructura:

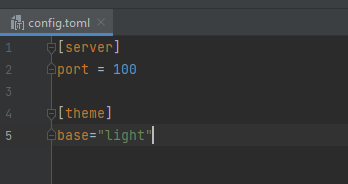


### Archivo .Streamlit

Como puede verse, el primer directorio “.streamlit” contiene el archivo de configuración.



En este archivo de configuración de la app aparece el puerto en el que correrá la aplicación y el tema a utilizar.



### Directorio Foretast

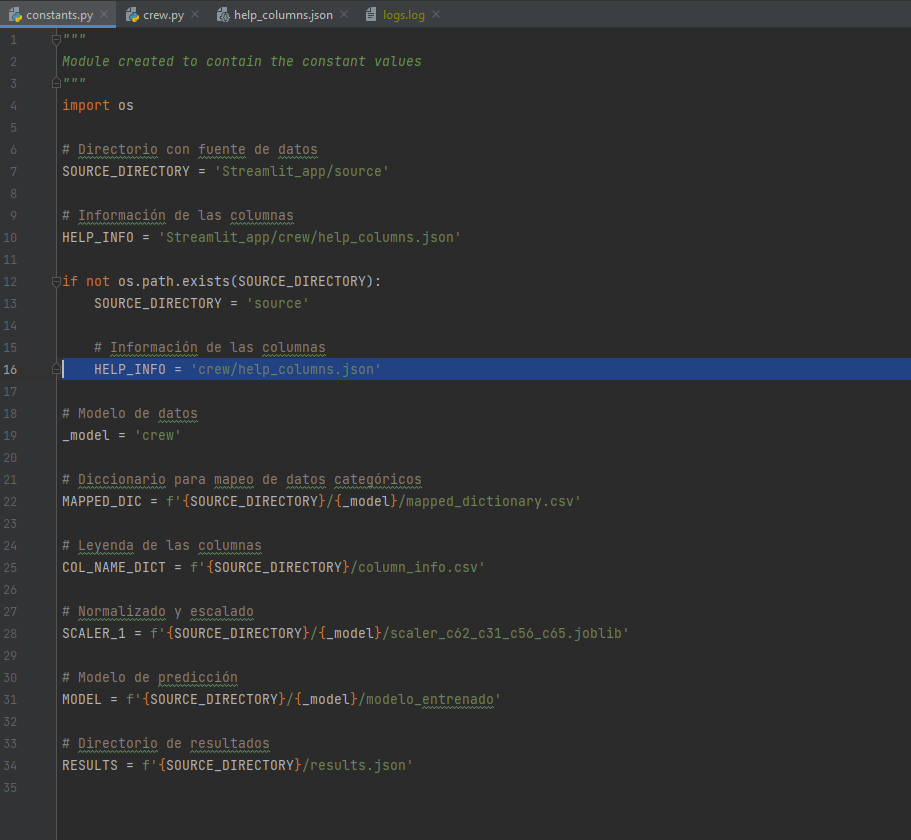
### Directorio Airplane

### Directorio Crew

#### Constants.py



En el directorio crew tenemos el archivo constants.py el cual contiene las rutas de archivos que necesitaremos para cada una de las funciones para la sección de “crew” de la aplicación.

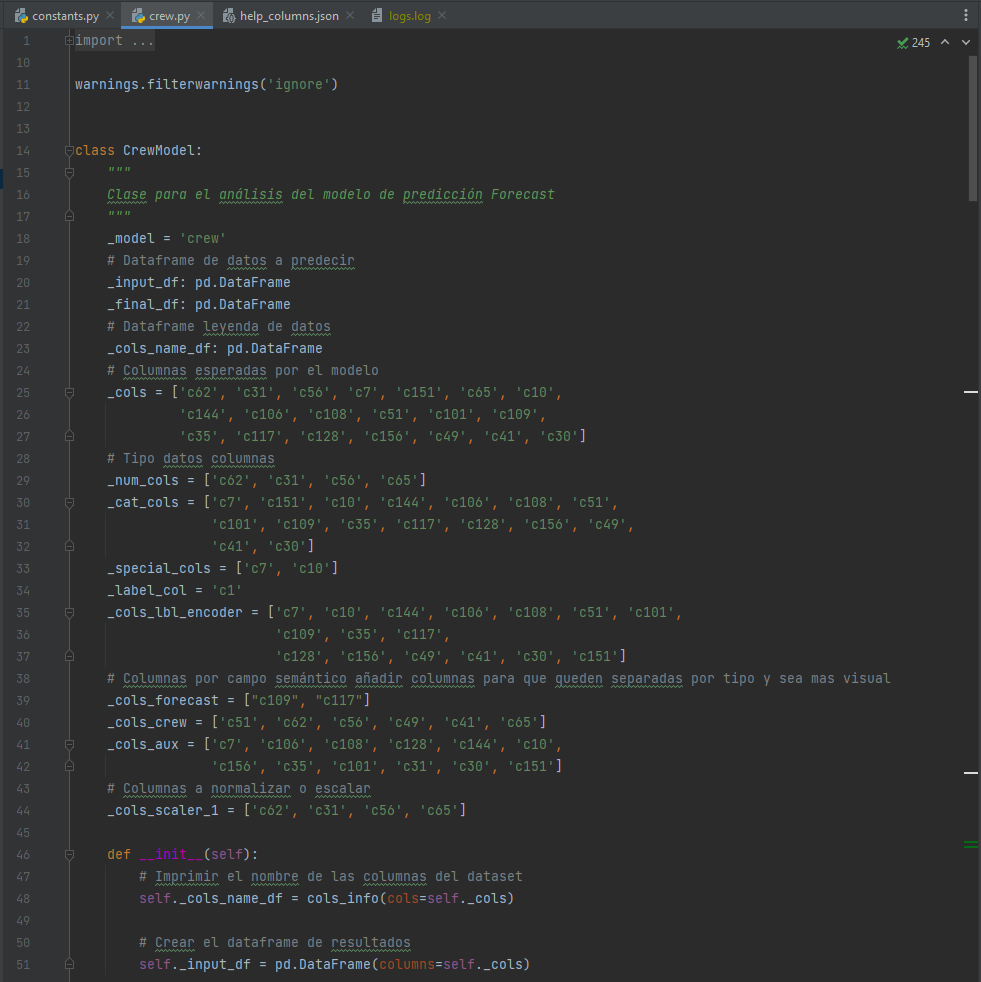


#### Crew.py

En este directorio también tenemos al archivo crew.py el cual contiene todas las funciones que se utilizarán en la sección “crew”.

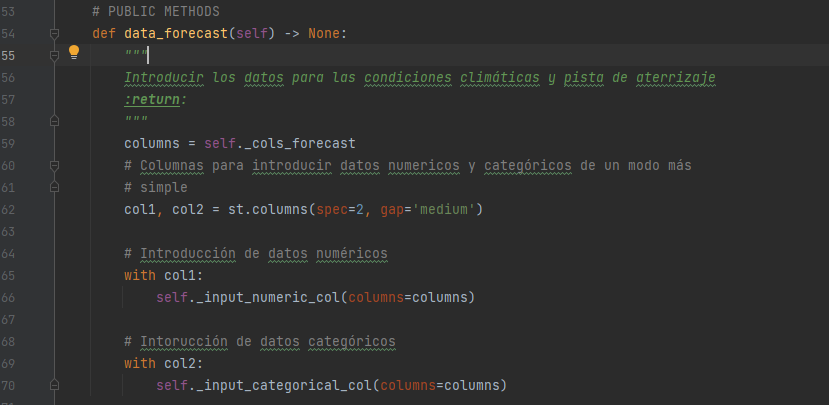
Primeramente se definen las columnas que se utilizarán dividiéndolas según sea numéricas o categóricas, segmentando además por tema de contenido para mostrar cada campo en las pestañas correspondientes. Además también se detallan las columnas que necesitan escalado y las que necesitan codificación.

Por ultimo en el método \_\_ini\_\_ se imprime las columnas del dataset y se crea el dataset de los resultados.

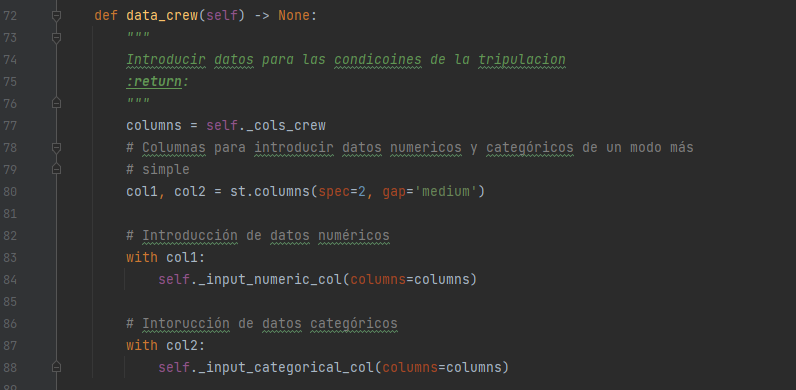


##### Métodos Publicos

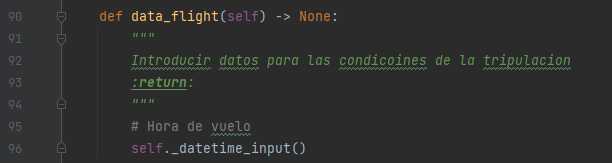
En data\_forecast se establece las columnas y disposión que aparecerá en la pestaña “forecast” de la sección “crew”.



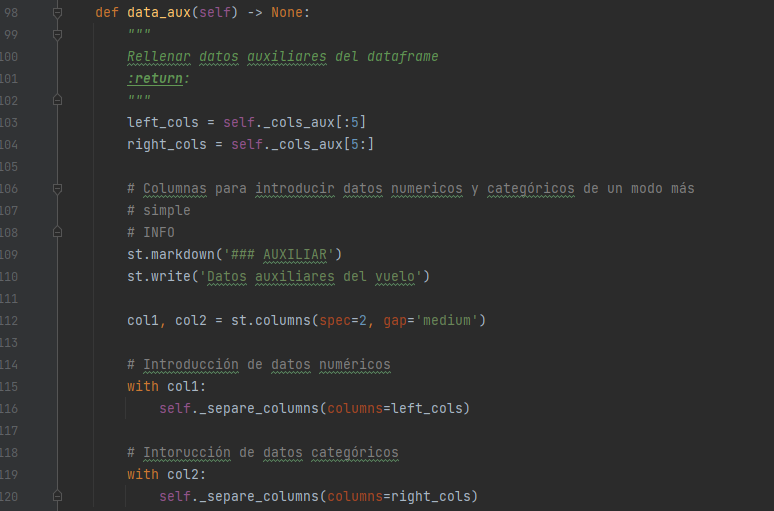
En data\_crew ocurre exactamente lo mismo que en el apartado anterior pero para la pestaña “crew”



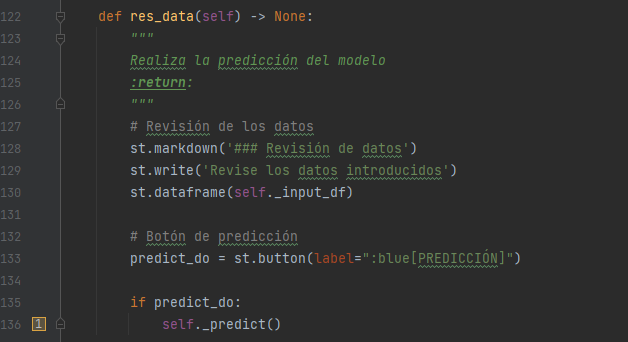
En data\_flight ocurre lo mismo pero para la pestaña “flight”



Y por último en data\_aux también se definen las posiciones de las columnas pero de la pestaña “auxiliar”

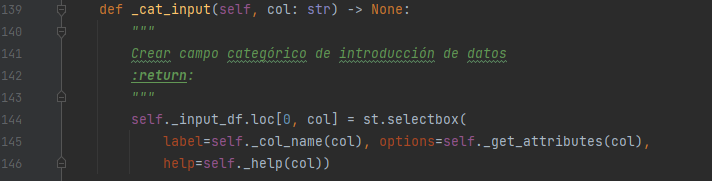


Con res\_data realizamos la predicción del modelo

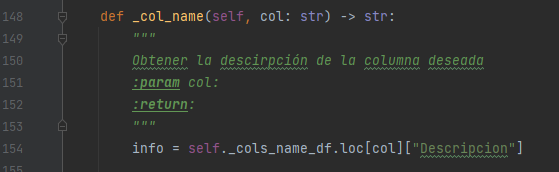


##### Métodos privados

Con \_cat\_input se crea un campo categorico para la introducción de datos



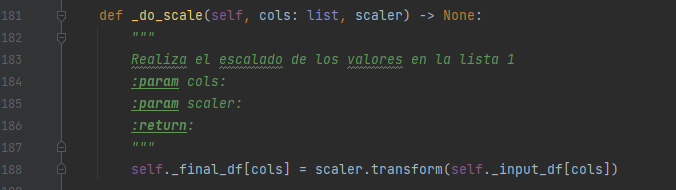
Con \_col\_name se obtiene la descripción de la columna deseada



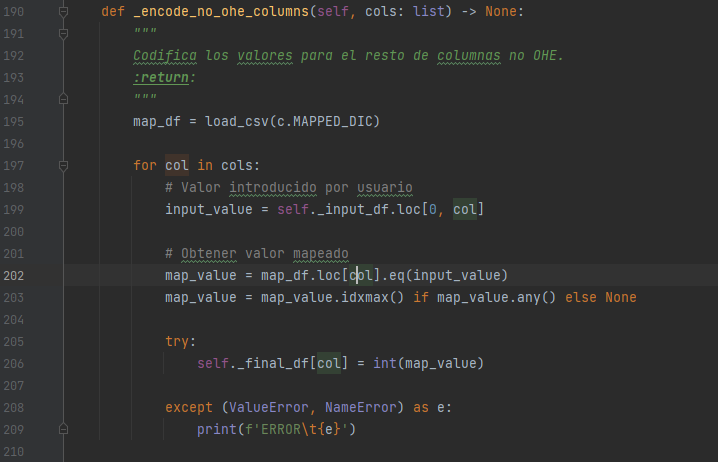
Con \_datetime\_input se establece los inputs de fecha y hora de vuelo.



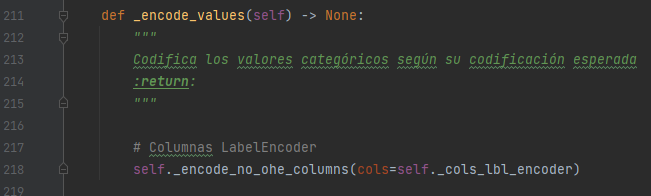
Con \_do\_scale realizamos el escalado de las columnas que lo requieren.



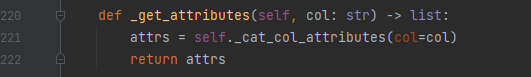
Con \_encode\_no\_ohe\_columns codificamos los valores para el resto de columnas en las que no se ha utilizado OneHotEncoder.



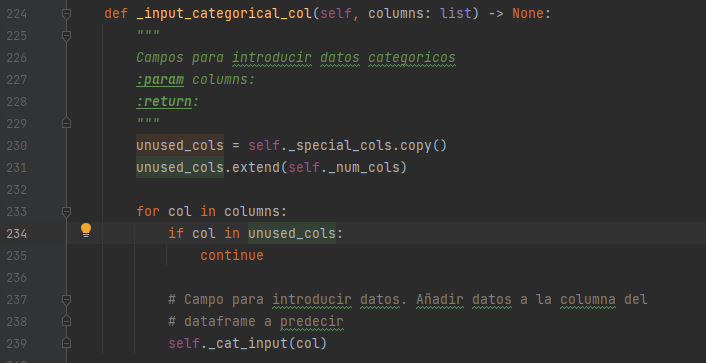
En \_encode\_values se codifican los valores categóricos según su codificación esperada.



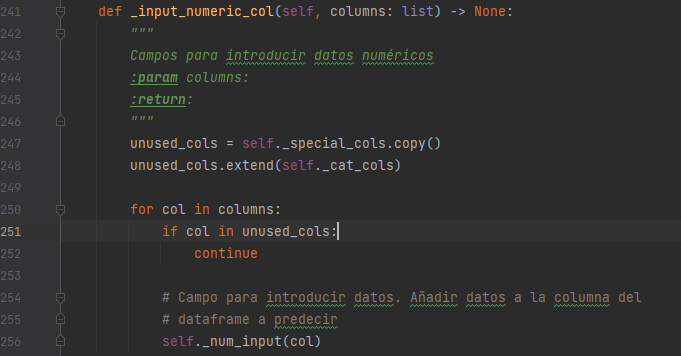
En \_get\_atributes se obtienen los atributos de las columnas categóricas.



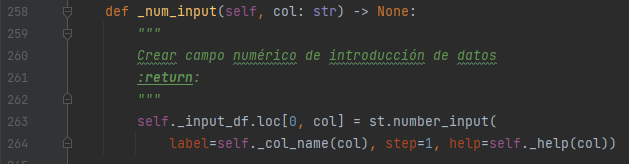
En \_input\_categorical\_col se establecen los campos para introducir datos categóricos.



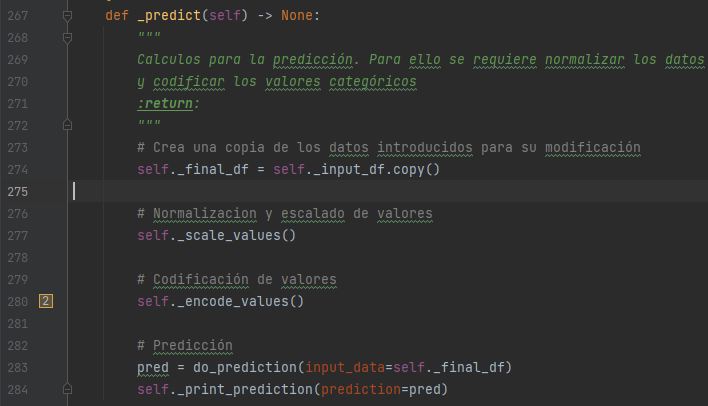
En \_input\_numeric\_col se establecen los campos para introducir datos numéricos.



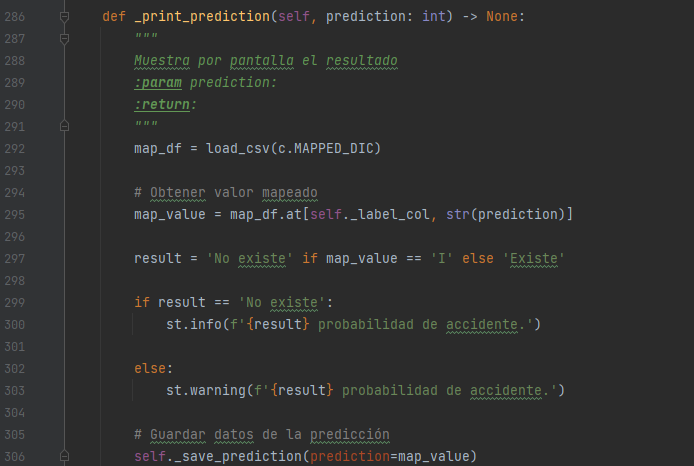
En \_num\_input se establecen los campos para introducir datos numéricos.



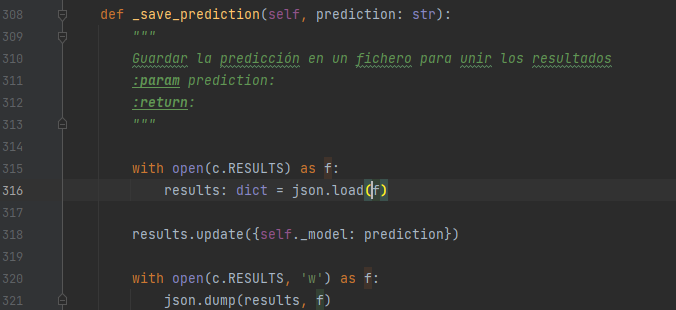
En \_predict se realizan los cálculos para la predicción. Para ello se requiere normalizar los datos y codificar los valores categóricos



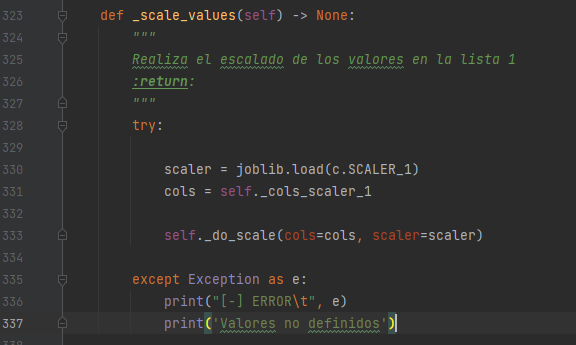
En \_print\_prediction se muestra por pantalla el resultado.



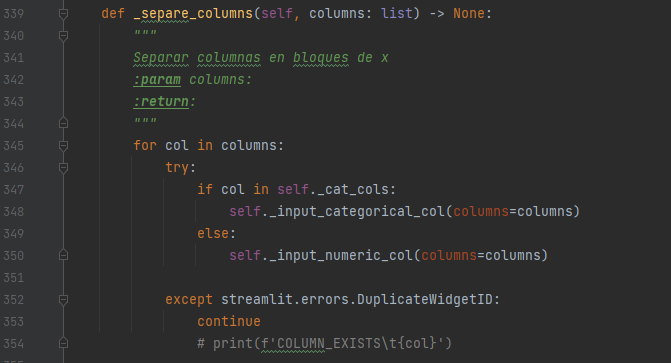
Con \_save\_prediction se guarda la predicción en un fichero para unir los resultados.



En \_scale\_values se realiza el escalado de los valores.

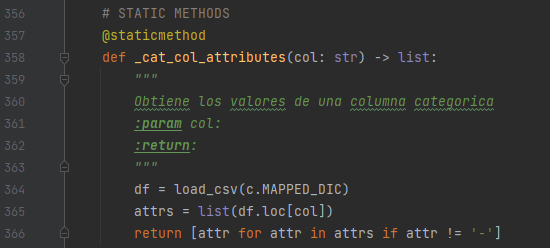


En \_separe\_columns se separan las columnas en bloques.

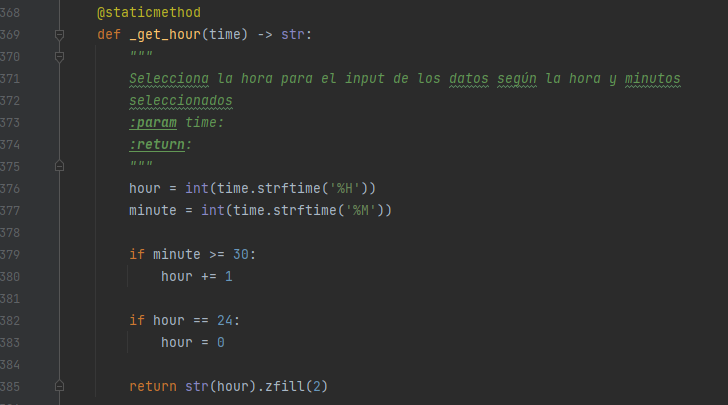


##### Métodos estáticos.

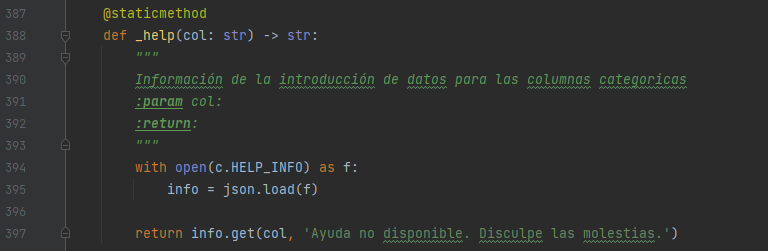
En \_cat\_col\_atributes se obtienen los valores de una columna catregórica.



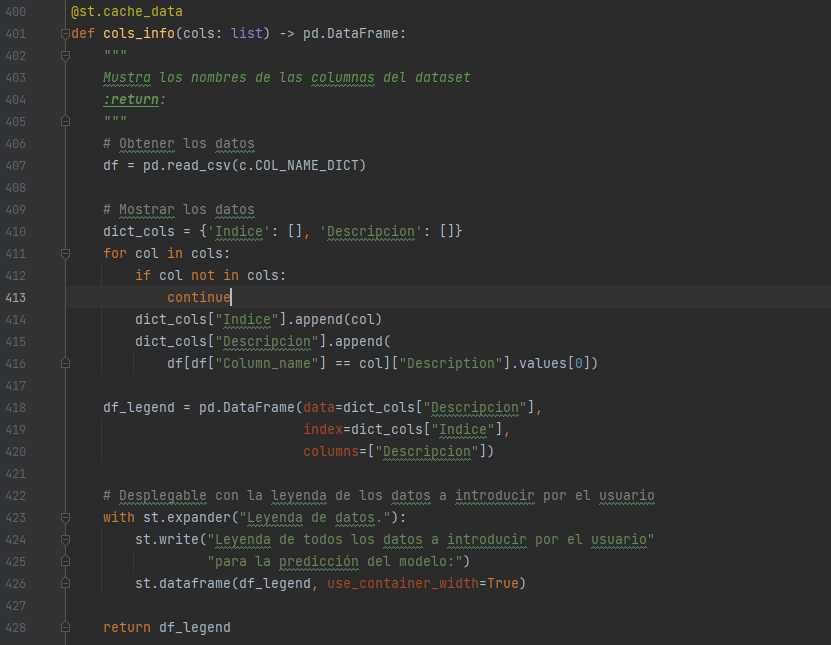
En \_get\_hour se selecciona la hora para el input de los datos según la hora y minutos seleccionados.



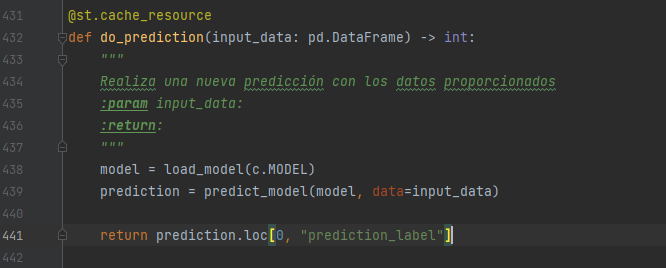
En \_help muestra la información de la introducción de datos para las columnas categóricas.



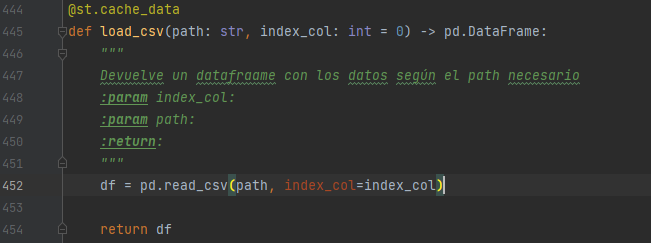
En cols\_info se muestra los nombres de las columnas del dataset



En do\_prediction se realiza una nueva predicción con los datos proporcionados.

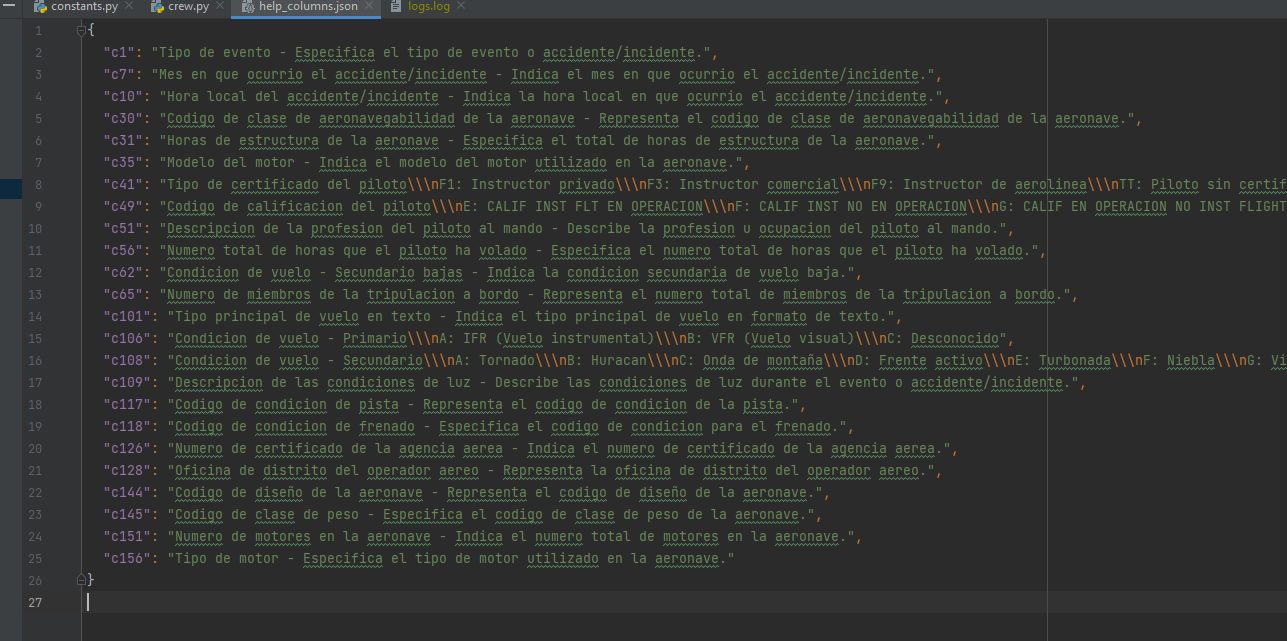


Por último en load\_csv devuelve un dataframe con los datos según el path necesario.



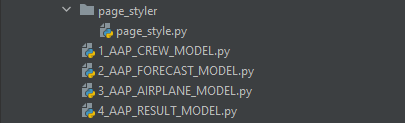
#### Help\_columns

Es un json donde se establece los valores de las columnas para el apartado de ayuda.



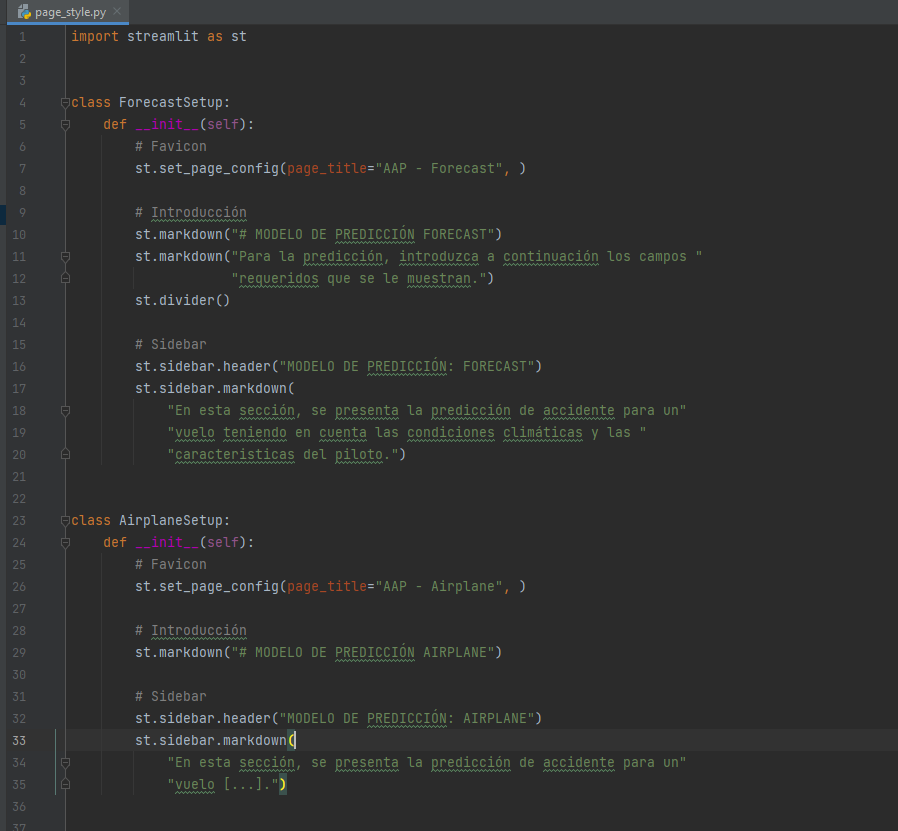
### Directorio pages.

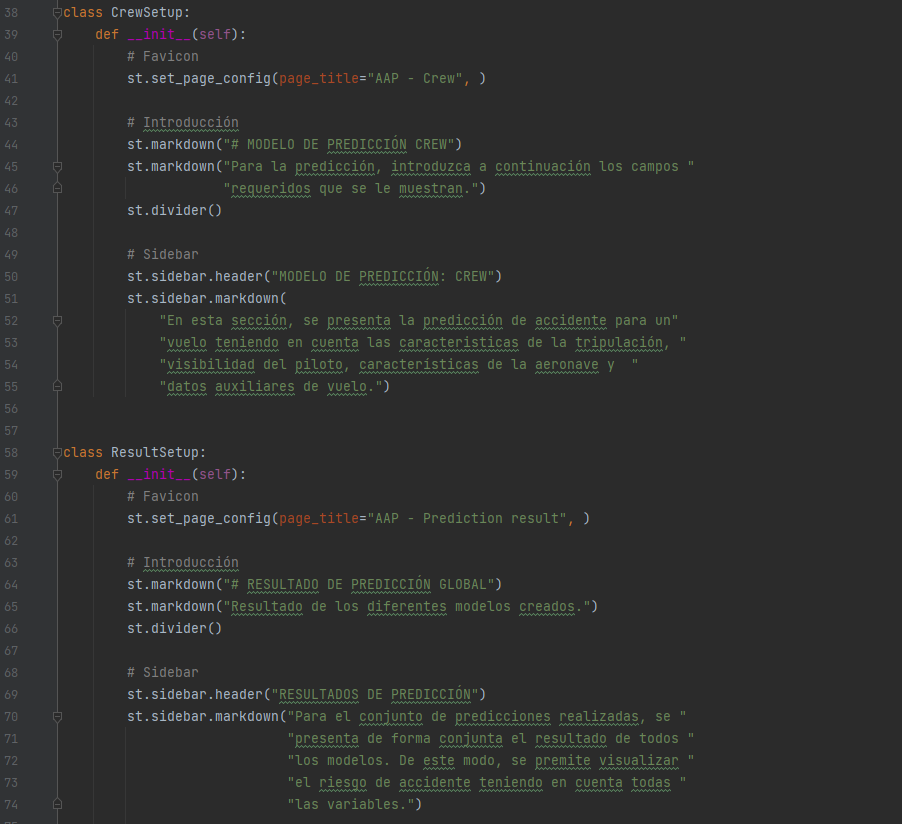
Contiene las páginas de cada sección.



#### Page\_styler.py

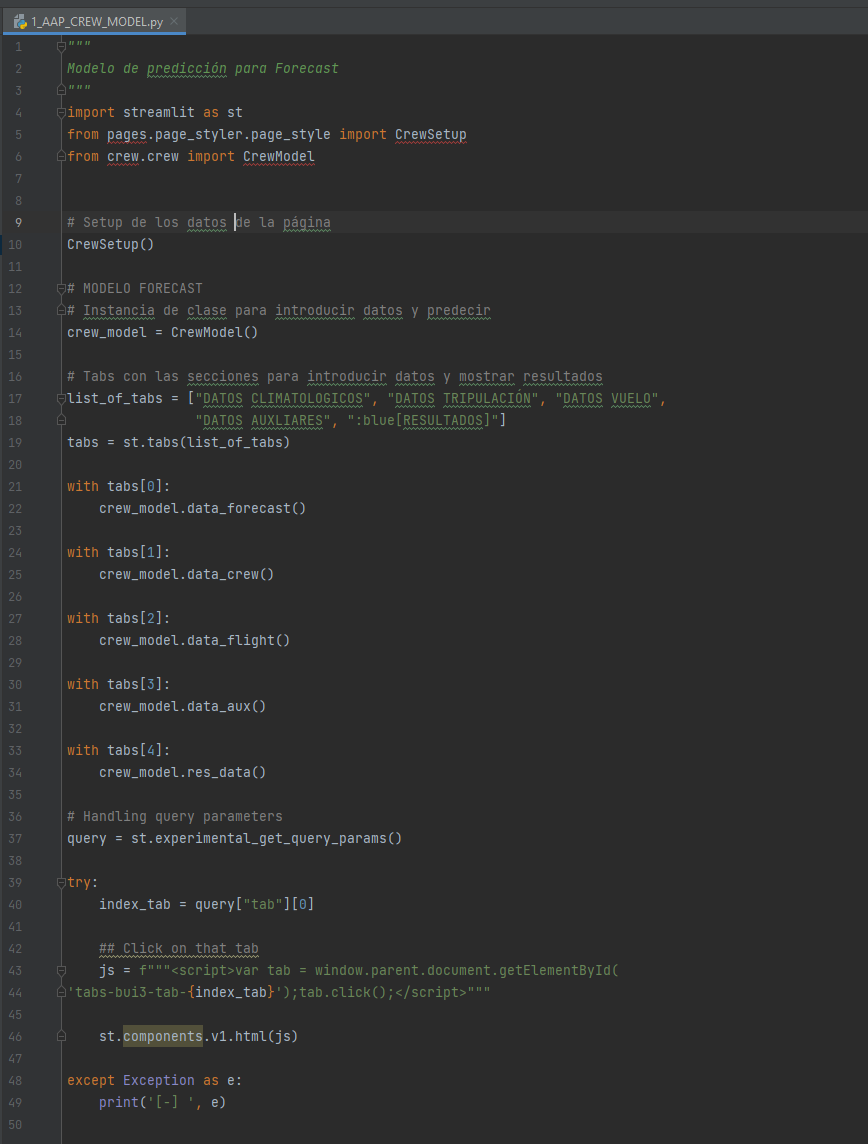
En este archivo se estable la descripción de cada sección para el menú izquierdo de navegación





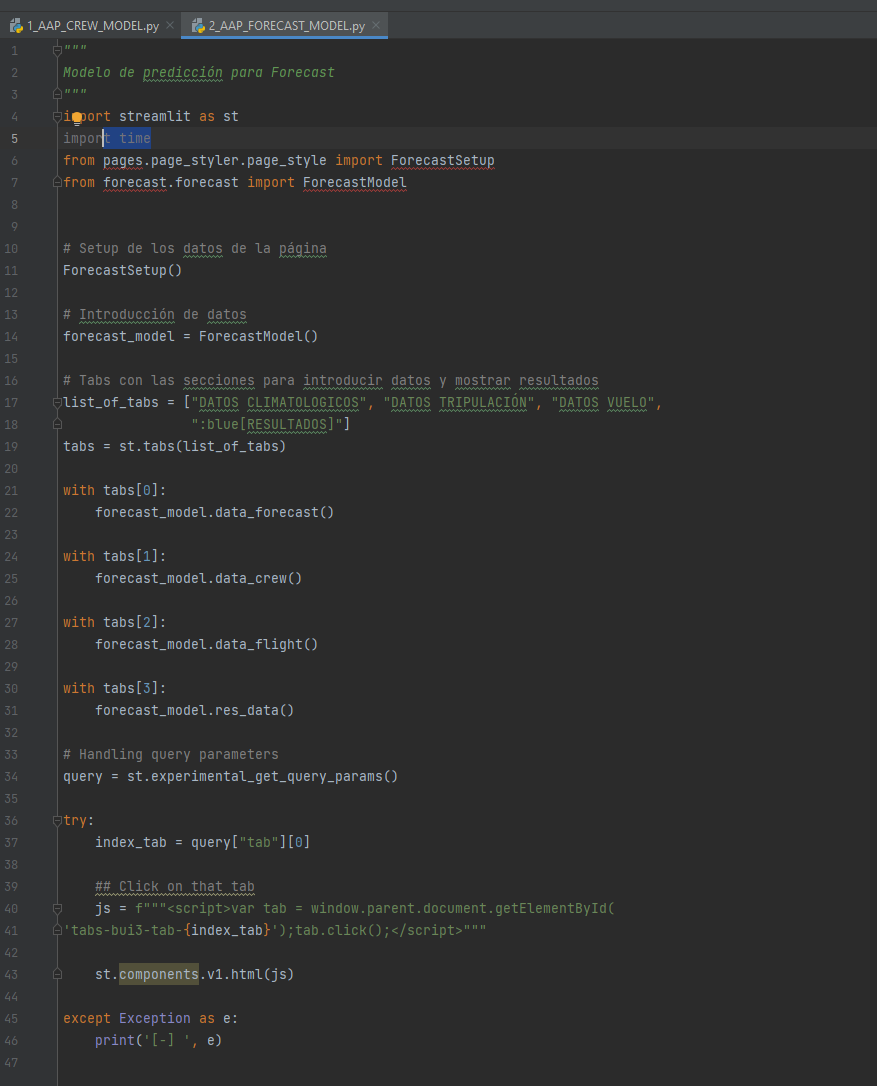
#### 1\_AAP\_CREW\_MODEL.py

En este archivo se configura la página de crew generando 5 tabs.



#### 2\_AAP\_FORECAST\_MODEL.py

En este archivo se configura la página de forecast generando 4 tabs.



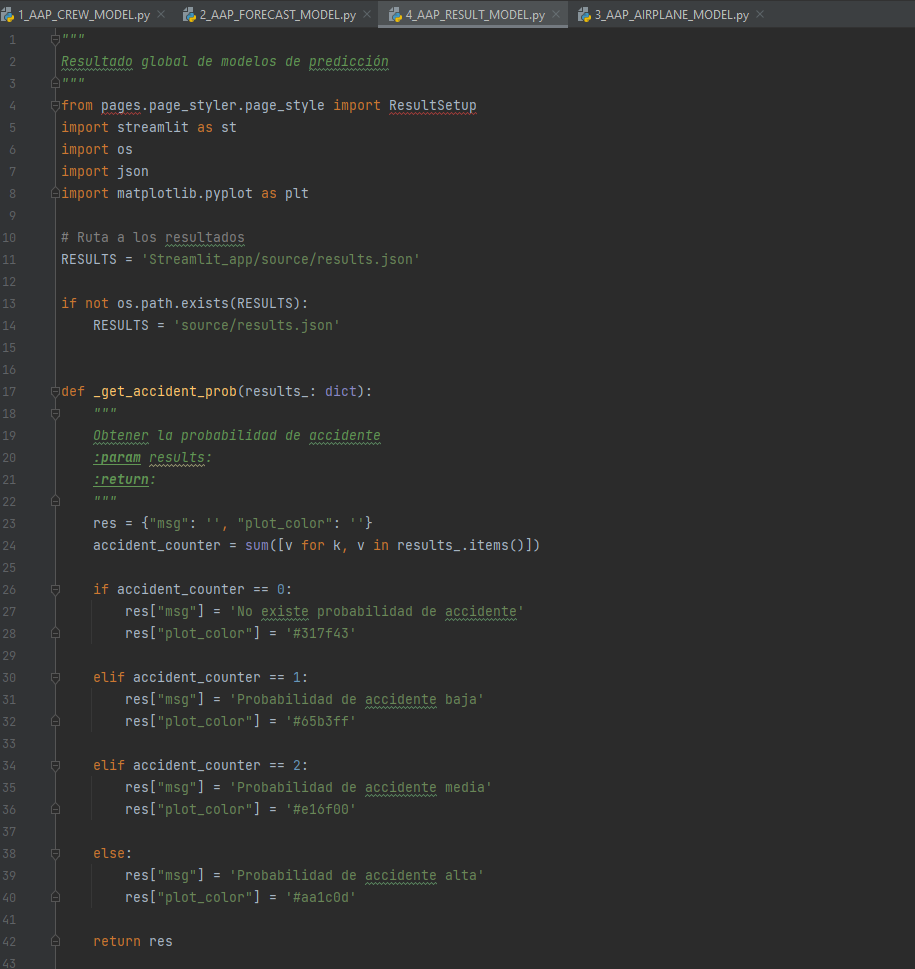
#### 3\_AAP\_AIRPLANE\_MODEL.py(PABLO)

#### 4\_AAP\_RESULT\_MODEL.py

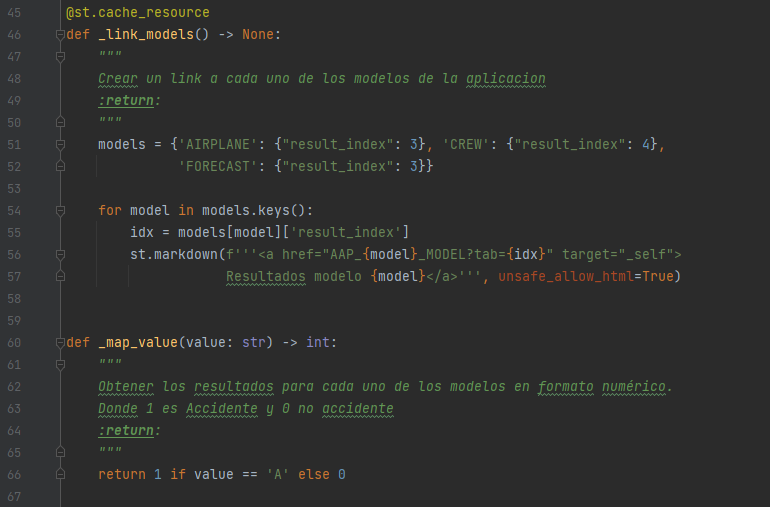
En este archivo se procesan los resultados obtenidos por cada modelo y se visualizan.

Primeramente se carga “result.json”el cual tiene almacenado los resultados de cada modelo.

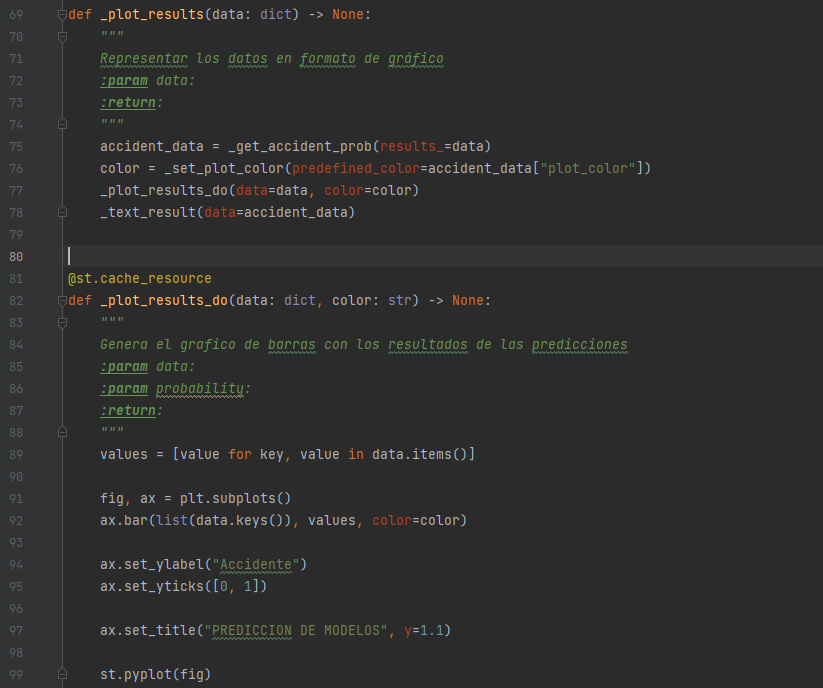
Posteriormente se establece las condiciones de color y los mensajes para cada una de las posibilidades en el método \_get\_accident\_prob.



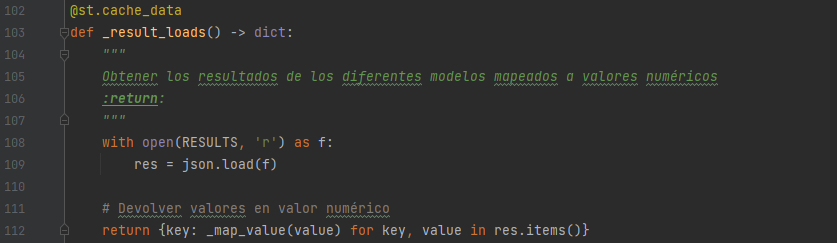
Seguidamente se crean estos métodos para crear links a cada uno de los modelos para mayor comodidad y se mapean los resultados.



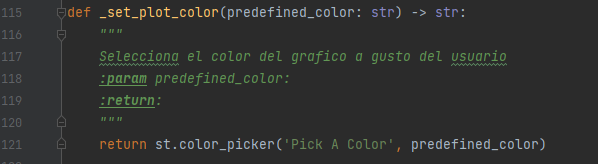
Con estos métodos se generan los gráficos según los resultados obtenidos.



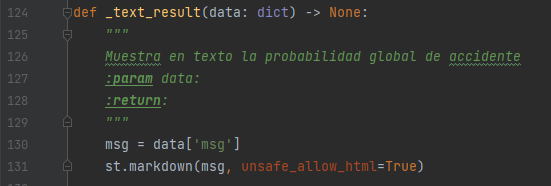
En este método se cargan los resultados



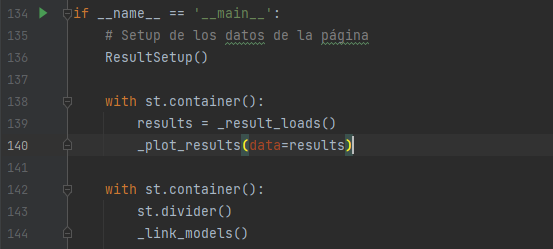
En este método se da la posibilidad de que el usuario selecciona el color del gráfico a su gusto



En este método se muestra la probabilidad global de accidente

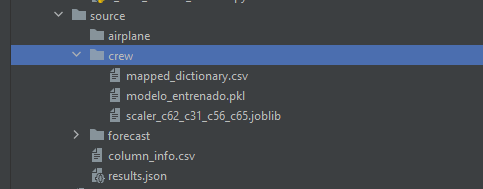


Por ultimo establecemos el Setup de los datos de la página



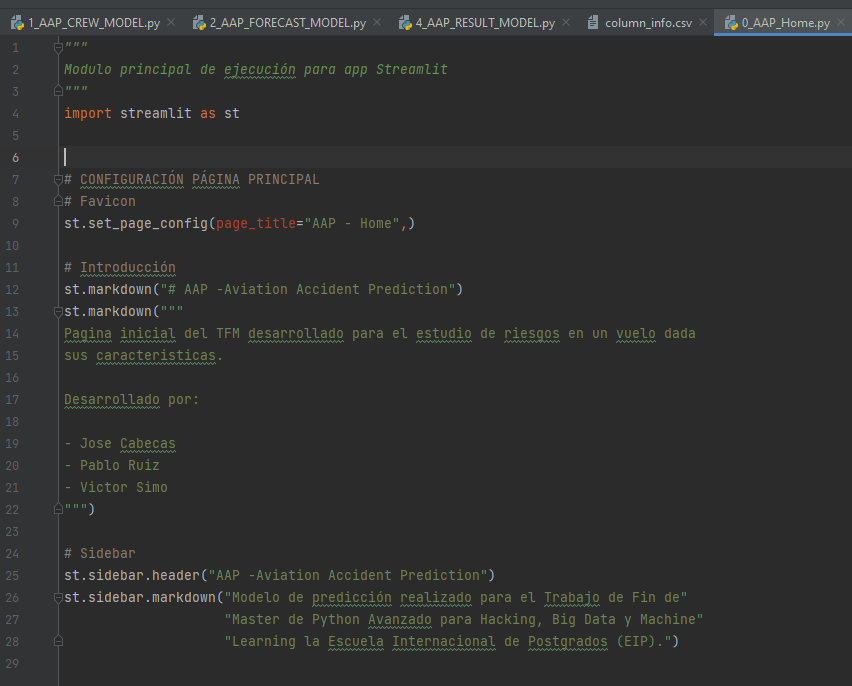
### Source(INCOMPLETO)

En el directorio Source tenemos almacenado los archivos que contienen los escalados, las codificaciones de label\_encoder y OneHotEncoder, los modelos ,el json de results.json mencionado anteniormente y column\_info.csv el cual contiene los nombres de las columnas.



### 0\_AAP\_Home.

En este archivo se configura la página principal de la app donde se muestra un resumen del proyecto.



## Unión de las partes y prueba en local

## Subida a la StreamlitCloud

# Evaluación de los resultados

En primer lugar, se realizó un análisis exploratorio detallado de los datos de accidentes de aviones. A través de este análisis, se identificaron patrones y tendencias significativas en los datos, lo que permitió comprender mejor la naturaleza de los accidentes y los factores que podrían contribuir a su ocurrencia.

Posteriormente, se procedió al preprocesamiento de los datos, abordando cuestiones como la limpieza de datos, el manejo de valores faltantes y la normalización de variables. Este proceso fue fundamental para garantizar la calidad y consistencia de los datos utilizados en los modelos de predicción.

Se construyeron modelos de predicción individuales para cada una de las agrupaciones de columnas de datos (tripulación, aeronave y datos climatológicos). Se aplicaron diversos algoritmos de aprendizaje automático para entrenar y ajustar estos modelos, con el objetivo de predecir la probabilidad de accidentes de aviones.

A través de la implementación de una interfaz de usuario utilizando Streamlit, se permitió a los usuarios introducir los datos relevantes en cada uno de los modelos de predicción. Esto facilitó la interacción con el sistema y brindó una experiencia amigable al usuario.

Al integrar los resultados de los modelos individuales y realizar un análisis global, se obtuvieron conclusiones significativas sobre la probabilidad de ocurrencia de accidentes de aviones. Se pudo establecer una visión más precisa y confiable al considerar la combinación de los resultados individuales. Esto permitió identificar patrones comunes y factores de riesgo que influyen en la probabilidad de accidentes.

En resumen, los resultados obtenidos de nuestro TFM demostraron la eficacia de los modelos de predicción construidos utilizando técnicas de aprendizaje automático y datos relevantes. La integración de estos modelos y el análisis global de los resultados proporcionaron una comprensión más profunda de la probabilidad de ocurrencia de accidentes de aviones, lo que contribuye a la mejora de la seguridad en la industria de la aviación. Los aspectos destacados de esta evaluación incluyen la identificación de patrones, la validación de los modelos y la identificación de factores clave para la prevención de accidentes. Estos hallazgos son fundamentales para tomar medidas preventivas y mejorar la seguridad en el sector aeronáutico.

# Conclusiones

Durante la investigación, hemos desarrollado con éxito un sistema de predicción de accidentes aéreos utilizando métodos de aprendizaje automático y datos relevantes. El sistema es una herramienta eficaz para evaluar la probabilidad de accidentes y ayuda a mejorar la seguridad de la industria de la aviación.

Realizamos un análisis exploratorio detallado de los datos de accidentes de aeronaves para identificar patrones y tendencias importantes. Esto ha llevado a una mejor comprensión de los factores que influyen en los accidentes, proporcionando una base sólida para medidas de prevención más eficaces.

El preprocesamiento completo de los datos es esencial para garantizar la calidad y la consistencia de los datos. Mediante el uso de tareas de limpieza, manejo de valores perdidos y normalización de variables, mejoramos la precisión del modelo predicho y aumentamos la confiabilidad de los resultados obtenidos. Se ha demostrado que los modelos predictivos que construimos para cada conjunto de columnas de datos (tripulación, aeronave y datos meteorológicos) son efectivos para predecir la probabilidad de un accidente de aeronave individual.

Este enfoque permite una comprensión detallada de los factores específicos que pueden influir en la ocurrencia de accidentes, lo cual es esencial para la prevención y la toma de decisiones informadas. La implementación de una interfaz de usuario intuitiva con Streamlit mejora la interacción con el sistema y facilita su uso a los usuarios. Esto facilita el ingreso de los datos apropiados en cada modelo de predicción y mejora la experiencia general del usuario al usar nuestro sistema.

Al integrar los resultados individuales de los modelos de predicción y analizarlos globalmente, obtenemos una comprensión más precisa y confiable de la probabilidad de accidentes aéreos. Esta evaluación conjunta tiene en cuenta la combinación de resultados individuales, lo que nos permite determinar la importancia de múltiples factores y obtener una evaluación más precisa de los riesgos asociados.

Con el TFM hemos desarrollado con éxito un sistema de predicción de accidentes aéreos que ayuda a mejorar la seguridad en la industria aeronáutica. Nuestros resultados confirman la validez de los modelos predictivos, la relevancia de los datos utilizados y la importancia de aplicar técnicas de aprendizaje automático para tomar decisiones informadas en seguridad aérea.

# Referencias

Administration, F. A. (s.f.). *Federal Aviation Administration - Moving america safety*. Obtenido de https://av-info.faa.gov/dd\_sublevel.asp?Folder=%5CAID

# Anexos TFM

Se trata de una sección que se utiliza para adjuntar la información en detalle.

En este caso, el fichero que contenga el código de su trabajo. Suele añadirse cualquier documento que por tamaño no se haya podido exponer en el trabajo y que aporte algún conocimiento o recurso adicional al resto del proyecto.