ESCUELA INTERNACIONAL DE POSTGRADOS

TRABAJO FINAL DE MASTER : OPTIMIZACIÓN DE LAS DECICIONES DE RESERVA DE VUELOS

FECHA ENTREGA :AGOSTO DE 2024

ALUMNO: ELENA ARBIOL BARREDA 73260690D

TITULACIÓN: MÁSTER DE PROGRAMACIÓN AVANZADA EN PYTHON PARA BIG DATA,HACKING Y MACHINE LEARNING

CONTENIDO

Abstract

This master's thesis focuses on the development of a Python-based system for flight price prediction. To achieve this, the PyCaret library has been used to obtain a machine learning model with a preprocessed flight data set.

To obtain the result, it has been necessary a preprocessing of depleted data with the flight dataset. This includes tasks such as data cleaning, outlier removal and variable transformation. Through this process, a refined data set is obtained that can be used in predictive models.

The scikit-learn library is used to develop a machine learning model capable of predicting flight prices. scikit-learn allows exploring different classification algorithms and determining the best fit for a data set. In addition, it greatly simplifies the process of training and evaluating models.

In summary, this thesis focuses on the development of a Python-based flight price prediction system. This work includes the preprocessing of the flight dataset, the choice of the most appropriate classification algorithm. With this system we intend to contribute to improve the decision making when purchasing a flight.

KEYWORDS:, BigData, Machine Learning, PyCaret, Python

Resumen

Este trabajo de fin de máster se centra en el desarrollo de un sistema basado en Python para la predicción de precios de vuelos. Para lograrlo, se ha utilizado la biblioteca PyCaret para obtener un modelo de aprendizaje automático con un conjunto de datos de vuelos preprocesados.

Para obtener el resultado ha sido necesario un preprocesamiento de datos agotados con el conjunto de datos de vuelos. Esto incluye tareas como limpieza de datos, eliminación de valores atípicos y transformación de variables. A través de este proceso, se obtiene un conjunto de datos refinado que puede ser utilizado en los modelos predictivos.

La biblioteca scikit-learn se utiliza para desarrollar un modelo de aprendizaje automático capaz de predecir los precios de los vuelos. scikit-learn permite explorar diferentes algoritmos de clasificación y determinar el mejor ajuste para un conjunto de datos. Además, simplifica enormemente el proceso de formación y evaluación de modelos.

En resumen, esta tesis se centra en el desarrollo de un sistema de predicción de precios de vuelos basado en Python. Este trabajo incluye el preprocesamiento del conjunto de datos de vuelos, la elección del algoritmo de clasificación más apropiado. Con este sistema pretendemos contribuir a mejorar la toma de decisiones a la hora de comprar un vuelo.

KEYWORDS:, BigData, Machine Learning, PyCaret, Python,

Glosario

Big Data:	Conjunto de datos extremadamente grande y complejo que requiere técnicas especiales para su análisis.
Machine Learning:	Un subcampo de la inteligencia artificial que se enfoca en desarrollar algoritmos que permiten a las computadoras aprender y mejorar su rendimiento en tareas específicas.
Python:	Un lenguaje de programación ampliamente utilizado en ciencia de datos y desarrollo de aplicaciones.
NumPy:	Una biblioteca de Python utilizada para realizar operaciones numéricas en matrices y arreglos multidimensionales.
Pandas:	Una biblioteca de Python para el análisis y manipulación de datos tabulares.
Seaborn:	Una biblioteca de Python para realizar gráficos
Matplotlib:	Una biblioteca de Python para realizar gráficos
Scikit-Learn:	Una biblioteca de Python que ofrece herramientas para el aprendizaje automático y la minería de datos.
TensorFlow:	Una plataforma de código abierto desarrollada por Google para la creación y entrenamiento de modelos de aprendizaje automático.
Dataset:	Un conjunto de datos que contiene información relacionada para su análisis.
Jupyter Notebook	Un entorno de programación interactivo que permite combinar código, texto y visualizaciones en un solo documento.
URL	Uniform Resource Locator, una dirección web que permite acceder a recursos en línea.
Multicolinealidad	Una relación estadística en la que dos o más variables independientes en un modelo de regresión están altamente correlacionadas.
Dataframe	Una estructura de datos tabular utilizada en programación para organizar y analizar datos.
Extreme Gradient Boosting	Un algoritmo de aprendizaje automático que utiliza árboles de decisión para mejorar la precisión del modelo.
Tab	Un componente de interfaz de usuario utilizado en aplicaciones web para organizar y mostrar contenido en pestañas.
Framework:	Un conjunto de herramientas y reglas que facilitan el desarrollo de aplicaciones y sistemas.
Paths	Rutas o direcciones utilizadas para acceder a archivos o recursos en sistemas de archivos.
Sidebar	Una barra lateral en una interfaz de usuario que proporciona acceso rápido a opciones y funciones adicionales.

Introducción

1. Objetivos generales del trabajo

El objetivo general del proyecto de Fin de Máster (TFM) es desarrollar un sistema de predicción de precios de vuelos utilizando programación en Python y Machine Learning, haciendo uso de las bibliotecas Scikit-Learn junto un conjunto BigData de información de vuelos en formato.

El objetivo principal es ayudar a planificar un vuelo, eligiendo el mejor momento para volar para poder ahorrar en los costes además de tiempo.

Así mismo, puede ser de utilidad para que las aerolíneas puedan conocer también la predicción de los importes de los vuelos de la competencia y poder así realizar ofertas interesantes para captar posibles viajeros.

2. Justifica la elección del tema

Relevancia para los viajeros

La optimización de las decisiones de reserva de vuelos es un tema de crucial importancia para los viajeros. En un contexto donde los precios de los vuelos pueden variar significativamente en cortos periodos de tiempo, contar con estrategias optimizadas para realizar reservas puede suponer un ahorro considerable y mejorar la experiencia de viaje. Este trabajo se enfoca en desarrollar modelos y herramientas que permitan a los viajeros tomar decisiones más informadas, maximizando el valor de su dinero y minimizando el estrés asociado a la planificación de viajes.

Complejidad y Desafíos del Mercado de Vuelos

El mercado de vuelos es extremadamente dinámico y está influenciado por una multitud de factores como la oferta y demanda fluctuantes, políticas de precios complejas de las aerolíneas, temporadas altas y bajas, y eventos imprevistos que pueden alterar drásticamente las tarifas aéreas. La optimización de las decisiones de reserva enfrenta el desafío de considerar todos estos factores para prever los momentos más adecuados para realizar una compra, lo cual requiere el uso de técnicas avanzadas de análisis de datos y predicción.

Avances Tecnológicos

El campo de la inteligencia artificial, el aprendizaje automático y el análisis de grandes volúmenes de datos ha avanzado notablemente en los últimos años. Estos avances permiten desarrollar modelos predictivos que pueden analizar grandes conjuntos de datos históricos y actuales para identificar patrones y tendencias en los precios de los vuelos. La aplicación de estas tecnologías a la optimización de reservas puede proporcionar a los viajeros recomendaciones precisas y personalizadas, mejorando significativamente su capacidad para encontrar las mejores ofertas.

Beneficios económicos para los viajeros

Una optimización efectiva de las decisiones de reserva de vuelos puede resultar en ahorros económicos directos para los viajeros. Al poder identificar los momentos óptimos para realizar una reserva, los viajeros pueden evitar precios altos y aprovechar tarifas más bajas. Esto es particularmente beneficioso para los viajeros frecuentes y aquellos con

presupuestos limitados, permitiéndoles destinar más recursos a otros aspectos de su viajo o a aumentar la frecuencia de sus viajes.

Impacto en la experiencia del usuario

La incertidumbre y el estrés asociados con la variabilidad de los precios de los vuelos pueden afectar negativamente la experiencia de los viajeros. Un sistema que optimiza las decisiones de reserva no solo facilita la planificación del viaje, sino que también aumenta la satisfacción del usuario al reducir la ansiedad por encontrar el mejor precio. Además, herramientas de optimización bien diseñadas pueden ofrecer interfaces amigables y recomendaciones claras, haciendo que el proceso de reserva sea más sencillo y agradable.

Contribución al conocimiento académico y aplicaciones prácticas

El estudio de la optimización de decisiones de reserva de vuelos no solo tiene implicaciones prácticas inmediatas, sino que también contribuye al cuerpo de conocimiento académico en áreas como la investigación operativa, la economía del comportamiento y la ciencia de datos. Los modelos y técnicas desarrollados en este trabajo pueden ser aplicados y extendidos a otros dominios del transporte y la logística, generando un impacto positivo más allá del sector de los viajes aéreos.

Conclusión

La elección del tema de optimización de las decisiones de reserva de vuelos para los viajeros se justifica plenamente por su alta relevancia y potencial de impacto positivo. Este trabajo no solo busca mejorar la experiencia de los viajeros y proporcionarles beneficios económicos directos, sino que también contribuye al avance académico en áreas tecnológicas y científicas clave. En resumen, abordar este tema representa una oportunidad para mejorar significativamente la manera en que las personas planifican y disfrutan sus viajes, haciendo uso de tecnologías avanzadas y enfoques innovadores.

3. Resumen metodología

Para abordar la optimización de las decisiones de reserva de vuelos para los viajeros, se empleará una metodología estructurada en varias fases clave. Estas fases integran técnicas de análisis de datos, modelado predictivo y validación empírica para desarrollar y evaluar un modelo que pueda predecir los mejores momentos para realizar reservas de vuelos. A continuación, se describe el enfoque metodológico:

3.1. Recolección de Datos

Se recopilarán datos históricos de precios de vuelos provenientes de múltiples fuentes, incluyendo motores de búsqueda de vuelos, sitios web de aerolíneas y plataformas de viajes. Los datos incluirán información sobre:

- Precios de vuelos a lo largo del tiempo
- Fechas de viaje
- Rutas de vuelo
- Compañías aéreas
- Factores externos relevantes (p.ej., eventos especiales, temporadas altas y bajas)

3.2. Limpieza y Preparación de Datos

Los datos recolectados serán sometidos a un proceso de limpieza para eliminar inconsistencias, valores atípicos y datos incompletos. Posteriormente, se prepararán los datos para el análisis, lo que incluye:

- Normalización de precios
- Codificación de variables categóricas (p.ej., aerolíneas, rutas)
- Generación de variables derivadas (p.ej., tiempo hasta la fecha de vuelo)

3.3. Análisis Exploratorio de Datos

Se realizará un análisis exploratorio de datos (EDA) para identificar patrones y tendencias en los precios de los vuelos. Esto implicará:

- Visualización de la evolución de precios a lo largo del tiempo
- Identificación de períodos de alta y baja demanda
- Análisis de la variabilidad de precios según diferentes rutas y aerolíneas

3.4. Desarrollo del Modelo Predictivo

Se emplearán técnicas de aprendizaje automático para desarrollar un modelo predictivo que pueda estimar el mejor momento para reservar vuelos. Los pasos incluyen:

- Selección de algoritmos de aprendizaje supervisado (p.ej., regresión lineal, árboles de decisión, redes neuronales)

- Entrenamiento del modelo usando los datos históricos preparados

- Validación del modelo mediante técnicas como validación cruzada y división de datos en conjuntos de entrenamiento y prueba

3.5. Evaluación y Validación del Modelo

La precisión y efectividad del modelo predictivo serán evaluadas utilizando métricas de rendimiento como el error medio absoluto (MAE) y el error cuadrático medio (MSE). Se realizarán pruebas adicionales para:

- Comparar el desempeño del modelo con estrategias de reserva tradicionales

- Evaluar la robustez del modelo en diferentes escenarios y condiciones del mercado

3.6. Implementación y Prueba Piloto

Se desarrollará una herramienta prototipo basada en el modelo predictivo que ofrezca recomendaciones a los usuarios sobre el mejor momento para reservar vuelos. Esta herramienta será sometida a una prueba piloto con un grupo de usuarios para:

- Recibir retroalimentación sobre la usabilidad y efectividad de las recomendaciones

- Realizar ajustes y mejoras en el modelo y la interfaz de usuario

3.7. Análisis de Resultados y Conclusiones

Finalmente, se analizarán los resultados obtenidos durante la prueba piloto y se compararán con las expectativas iniciales. Se extraerán conclusiones sobre:

- La viabilidad y precisión del modelo predictivo

- Los beneficios potenciales para los viajeros

- Las posibles áreas de mejora y futuras líneas de investigación

Conclusión

La metodología propuesta para la optimización de las decisiones de reserva de vuelos combina técnicas avanzadas de análisis de datos y aprendizaje automático para desarrollar una herramienta práctica y eficaz para los viajeros. A través de un enfoque riguroso y sistemático, se busca proporcionar recomendaciones precisas que puedan ayudar a los usuarios a tomar decisiones informadas y económicas al reservar vuelos.

4. Estructura

Planteamiento del problema

En el sector del transporte aéreo, la volatilidad de los precios de los vuelos representa un desafío significativo tanto para los pasajeros como para las aerolíneas. Los precios de los vuelos pueden fluctuar considerablemente en función de múltiples factores, como la oferta y la demanda, las estrategias de precios de las aerolíneas, las temporadas de viaje y eventos imprevistos. Esta variabilidad genera una gran incertidumbre para los viajeros, quienes a menudo se enfrentan a la dificultad de decidir cuándo es el mejor momento para reservar sus vuelos al menor costo posible, dado que los precios de los vuelos no son constantes y pueden cambiar drásticamente en cortos períodos de tiempo, esta fluctuación dificulta que los pasajeros identifiquen el momento óptimo para realizar una reserva. Ante la falta de herramientas predictivas, aunque existen herramientas y sitios web que ofrecen comparaciones de precios, muchos de ellos no proporcionan predicciones precisas. Los viajeros deben considerar múltiples variables (rutas, fechas, aerolíneas, paradas, etc…) y no siempre tienen el tiempo o los recursos para analizar toda la información disponible. Esto puede resultar en decisiones subóptimas que no maximicen el ahorro económico. La incertidumbre y la presión de encontrar el mejor precio pueden causar estrés y ansiedad en los viajeros, afectando negativamente su experiencia de planificación de viajes. Por eso el objetivo principal de este estudio es desarrollar un modelo predictivo que permita optimizar las decisiones de reserva de vuelos para los viajeros.

Python como Lenguaje de Programación

La elección de Python como lenguaje de programación principal para abordar este problema de predicción de precios de vuelos aéreos se basa en diversas razones. En primer lugar, Python es ampliamente reconocido en la comunidad de ciencia de datos y aprendizaje automático por su versatilidad y robusta selección de bibliotecas. Esto permite aprovechar herramientas como NumPy, Pandas, Scikit-Learn y TensorFlow para el procesamiento de datos, la construcción de modelos predictivos y la implementación de interfaces de usuario interactivas. Además, Python cuenta con una comunidad activa y una abundancia de recursos en línea, lo que facilita la colaboración y el desarrollo continuo del proyecto.

En conjunto, estas decisiones técnicas respaldan el propósito de desarrollar un sistema objetivo y efectivo para predecir los precios de vuelos aéreos, facilitando la toma de decisiones relacionadas con la planificación de viajes de usuarios.

Objetivos del trabajo

El objetivo general del proyecto de Fin de Máster (TFM) es desarrollar un sistema de predicción de precios de vuelos, debido a que la optimización de las decisiones de reserva de vuelos es un tema de crucial importancia para los viajeros. En un contexto donde los precios de los vuelos pueden variar significativamente en cortos periodos de tiempo, contar con estrategias optimizadas para realizar reservas puede suponer un ahorro considerable y mejorar la experiencia de viaje. Este trabajo se enfoca en desarrollar modelos y herramientas que permitan a los viajeros tomar decisiones más informadas, maximizando el valor de su dinero y minimizando el estrés asociado a la planificación de viajes

En primer lugar, decidí realizar un análisis exploratorio de los datos existentes sobre precio de vuelos. A través de este análisis, pude comprender la estructura de los datos, identificar patrones existentes, detectar valores atípicos e identificar posibles correlaciones entre las variables.

El segundo objetivo se centra en el preprocesamiento de datos. Aquí, me enfoqué 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 fue crear un modelo de predicción utilizando un grupo específico de columnas de datos. Para hacer esto, usé algoritmos de aprendizaje automático para entrenar y adaptar un modelo de intentar predecir el precio de los vuelos.

El cuarto objetivo fue probar que dicho modelo de predicción funcionase correctamente.

En resumen, el trabajo se basa en el desarrollo de un sistema de predicción de precio de los vuelos aéreos, que se puede extrapolar a otro tipo de viajes, donde los objetivos secundarios son realizar análisis exploratorios, preprocesar los datos, crear modelos de predicción, elegir el más óptimo y realizar pruebas oportunas o test.

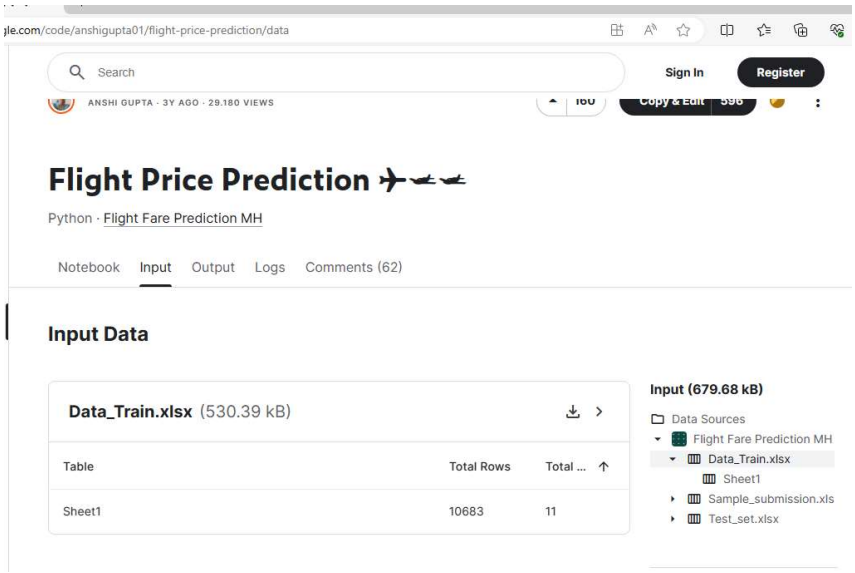
Metodología

1. Preprocesamiento del dataset

1.1. Dataset global

Los modelos de predicción de precios de vuelos que se desarrolla en el proyecto, parte de un conjunto de datos común y público obtenido de Kaggle en el siguiente enlace : Link: https://www.kaggle.com/code/anshigupta01/flight-price-prediction/data.

En este repositorio, se puede encontrar los datos en formato .xlsx desde marzo 2019 hasta junio del mismo año. El dataset consta de 11 columnas y 10.683 filas.



Una vez descargado el dataset voy a leer el dataset para entender las propiedades de los datos con ayuda de algunas técnicas de visualización.

1.2. Exploratory Data Analysis (EDA)

Lo primero importaré las librerías de Python necesarias para este proyecto.

Importar las librerias

```
# Manejo de datos
import pandas as pd

# Numéricos
import numpy as np

# Gráficos
import seaborn as sns
import missingno as msno
import matplotlib.pyplot as plt

# Análisis de datos
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.preprocessing import MinMaxScaler

# Modelado
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import accuracy_score, confusion_matrix
```

Después importaremos los datos y mostraré los 5 primeros datos o líneas

Observo que tengo 11 columnas que son:

Importar el dataset

```
df=pd.read_excel('dataset/Data_Train.xlsx')

#Mostramos los 5 primeros datos
df.head()
```

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price
0	IndiGo	24/03/2019	Banglore	New Delhi	BLR → DEL	22:20	01:10 22 Mar	2h 50m	non-stop	No info	3897
1	Air India	1/05/2019	Kolkata	Banglore	CCU → IXR → BBI → BLR	05:50	13:15	7h 25m	2 stops	No info	7662
2	Jet Airways	9/06/2019	Delhi	Cochin	DEL → LKO → BOM → COK	09:25	04:25 10 Jun	19h	2 stops	No info	13882
3	IndiGo	12/05/2019	Kolkata	Banglore	CCU → NAG → BLR	18:05	23:30	5h 25m	1 stop	No info	6218
4	IndiGo	01/03/2019	Banglore	New Delhi	BLR → NAG → DEL	16:50	21:35	4h 45m	1 stop	No info	13302

“Airline”: Muestra en texto el nombre de la aerolínea

“Date_of_Journey”:La fecha en la que se realiza el vuelo

“Source”: El origen del vuelo

“Destination”: El destino del vuelo

“Route”: Abreviación de la ruta del vuelo

“Dep_Time”: hora de inicio del vuelo

“Arrival_Time”: Hora de llegada del vuelo al destino

“Duration”: Duración del vuelo en horas y minutos

“Total_Stops”: Número de paradas del vuelo

“Aditional_Info”: Información adicional

“Price”: Precio total del vuelo, que es la variable a predecir

Veamos a ver cuántos datos tenemos:

```
# mostramos el tamaño del dataframe
df.shape
```

```
(10683, 11)
```

Tenemos 10.683 filas y 11 columnas

Voy a ver qué información de columna tenemos reflejada en el dataset para poder saber el tipo de dato y si habrá que convertirlo o no.

```
#Información acerca de los datos
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10683 entries, 0 to 10682
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   Airline          10683 non-null  object
 1   Date_of_Journey  10683 non-null  object
 2   Source           10683 non-null  object
 3   Destination      10683 non-null  object
 4   Route            10682 non-null  object
 5   Dep_Time         10683 non-null  object
 6   Arrival_Time     10683 non-null  object
 7   Duration         10683 non-null  object
 8   Total_Stops      10682 non-null  object
 9   Additional_Info  10683 non-null  object
 10  Price            10683 non-null  int64 
dtypes: int64(1), object(10)
memory usage: 918.2+ KB
```

Puedo observar que todas las columnas son del tipo object (datos cualitativos) salvo “Price” que es de tipo entero (datos cuantitativo nominal), y que faltan datos en las columnas “Route” y “ Total_Stops”, dado que tenemos 10.683 filas de datos enumeradas del 0 al 10.682. y en estas dos columnas aparecen 10.682 datos,lo verifico

```
# verificamos que tenemos datos faltantes
df.isnull().sum()
```

```
Airline          0
Date_of_Journey  0
Source           0
Destination      0
Route            1
Dep_Time         0
Arrival_Time     0
Duration         0
Total_Stops      1
Additional_Info  0
Price            0
dtype: int64
```

Y decido eliminar estos datos faltantes dado que es un porcentaje muy pequeño en comparación con los datos totales del dataset (son 1 fila de datos frente a 10.683) y verifico de nuevo que no hayan datos faltantes o vacíos.

```
#eliminamos estos valores faltantes dado el gran tamaño del dataset
df.dropna(inplace=True)
```

```
df.isnull().sum()
```

```
Airline            0
Date_of_Journey    0
Source             0
Destination        0
Route              0
Dep_Time           0
Arrival_Time       0
Duration           0
Total_Stops        0
Additional_Info    0
Price              0
dtype: int64
```

Saco los descriptivos, que como la única columna con datos enteros es “Price” son los únicos datos estadísticos que puedo sacar:

```
# estadísticas principales
df.describe()
```

	Price
count	10682.000000
mean	9087.214567
std	4611.548810
min	1759.000000
25%	5277.000000
50%	8372.000000
75%	12373.000000
max	79512.000000

Donde puedo ver que la media del precio de los vuelos es de 9087,21 $, con una desviación típica de 4611,54 $, el mínimo es de 1759,00 $, el máximo de 79512,00 y la mediana es de 8372,00 $.

1.3. Data preprocesing (DP)

Una vez se ha analizado el Dataset y se conoce las características principales de los datos, es necesario procesar los datos antes de crear un modelo de datos. En este punto, se va a corregir o realizar una “limpieza” de los datos que se tiene.

Data cleaning

```
df.dtypes
```

```
Airline            object
Date_of_Journey    object
Source             object
Destination        object
Route              object
Dep_Time           object
Arrival_Time       object
Duration           object
Total_Stops        object
Additional_Info    object
Price               int64
dtype: object
```

El tipo de datos de "Date_of_Journey", "Arrival_Time" y "Dep_Time" es de tipo objeto se tienen que convertir en fecha y tiempo para hacer las predicciones, es decir en datetime.

```
def change_into_datetime(col):
    df[col]=pd.to_datetime(df[col])
```

```
df.columns
```

```
Index(['Airline', 'Date_of_Journey', 'Source', 'Destination', 'Route',
       'Dep_Time', 'Arrival_Time', 'Duration', 'Total_Stops',
       'Additional_Info', 'Price'],
      dtype='object')
```

```
for i in ['Date_of_Journey','Dep_Time','Arrival_Time']:
    change_into_datetime(i)
```

```
df.dtypes
```

```
Airline                    object
Date_of_Journey    datetime64[ns]
Source                     object
Destination                object
Route                      object
Dep_Time           datetime64[ns]
Arrival_Time       datetime64[ns]
Duration                   object
Total_Stops                object
Additional_Info            object
Price                       int64
dtype: object
```

Voy a extraer el dia y mes de 'Date_of_Journey' y lo guardaré en dos columnas Luego eliminaré la columna 'Date_oj_Journey' dado que no la usaré.

```
df['journey_day']=df['Date_of_Journey'].dt.day
df['journey_month']=df['Date_of_Journey'].dt.month
df.drop('Date_of_Journey', axis=1, inplace=True)
```

```
df.head()
```

	Airline	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price	journey_day	journey_month
0	IndiGo	Banglore	New Delhi	BLR → DEL	2024-08-30 22:20:00	2024-03-22 01:10:00	2h 50m	non-stop	No info	3897	24	3
1	Air India	Kolkata	Banglore	CCU → IXR → BBI → BLR	2024-08-30 05:50:00	2024-08-30 13:15:00	7h 25m	2 stops	No info	7662	5	1
2	Jet Airways	Delhi	Cochin	DEL → LKO → BOM → COK	2024-08-30 09:25:00	2024-06-10 04:25:00	19h	2 stops	No info	13882	6	9
3	IndiGo	Kolkata	Banglore	CCU → NAG → BLR	2024-08-30 18:05:00	2024-08-30 23:30:00	5h 25m	1 stop	No info	6218	5	12
4	IndiGo	Banglore	New Delhi	BLR → NAG → DEL	2024-08-30 16:50:00	2024-08-30 21:35:00	4h 45m	1 stop	No info	13302	3	1

Ahora extraeré la hora y los minutos que están en las columnas 'Arrival_Time' y 'Dept_Time' y los guardaré en columnas nuevas y eliminaré estas columnas. Creo funciones para extraer hora y minutos y otra para eliminar las comunas. Posteriormente voy llamando a las funciones anteriormente definidas

```
#Creo funciones para extraer hora y minutos y otra para eliminar las columnas
def extract_hour(data, col):
    data[col+'_hour']=data[col].dt.hour
def extract_min(data, col):
    data[col+'_min']=data[col].dt.minute
def drop_col(data,col):
    data.drop(col, axis=1, inplace=True)
```

```
#voy llamando a las funciones anteriormente definidas
extract_hour(df,'Dep_Time')
extract_min(df,'Dep_Time')
drop_col(df, 'Dep_Time')

extract_hour(df,'Arrival_Time')
extract_min(df,'Arrival_Time')
drop_col(df, 'Arrival_Time')
```

```
df.head()
```

stination	Route	Duration	Total_Stops	Additional_Info	Price	journey_day	journey_month	Dep_Time_hour	Dep_Time_min	Arrival_Time_hour	Arrival_Time_min
ew Delhi	BLR → DEL	2h 50m	non-stop	No info	3897	24	3	22	20	1	10
Banglore	CCU → IXR → BBI → BLR	7h 25m	2 stops	No info	7662	5	1	5	50	13	15
Cochin	DEL → LKO → BOM → COK	19h	2 stops	No info	13882	6	9	9	25	4	25
Banglore	CCU → NAG → BLR	5h 25m	1 stop	No info	6218	5	12	18	5	23	30
ew Delhi	BLR → NAG → DEL	4h 45m	1 stop	No info	13302	3	1	16	50	21	35

Ahora aplicare pre-processing en la columna Duration, separando la duracion en horas y minutos

```
duration=list(df['Duration'])
for i in range(len(duration)):
    if len(duration[i].split(' '))==2:
        pass
    else:
        if 'h' in duration[i]:#verifico si duracion contiene solo horas
            duration[i]=duration[i] + ' 0m' # si no tiene minutos añado 0m
        else:
            duration[i]= '0h '+ duration[i] # si no tiene horas añado 0h
```

```
df['Duration']=duration
```

```
df.head()
```

stination	Route	Duration	Total_Stops	Additional_Info	Price	journey_day	journey_month	Dep_Time_hour	Dep_Time_min	Arrival_Time_hour	Arrival_Time_min
ew Delhi	BLR → DEL	2h 50m	non-stop	No info	3897	24	3	22	20	1	10
Banglore	CCU → IXR → BBI → BLR	7h 25m	2 stops	No info	7662	5	1	5	50	13	15
Cochin	DEL → LKO → BOM → COK	19h 0m	2 stops	No info	13882	6	9	9	25	4	25
Banglore	CCU → NAG → BLR	5h 25m	1 stop	No info	6218	5	12	18	5	23	30
ew Delhi	BLR → NAG → DEL	4h 45m	1 stop	No info	13302	3	1	16	50	21	35

Creo dos nuevas columnas con las duraciones en horas y otra de la duracion en minutos

```
def hour(x):
    return x.split(' ')[0][0:-1]
def minutes(x):
    return x.split(' ')[1][0:-1]
```

```
#Creo dos nuevas columnas con las duraciones en horas y otra de la duracion en minutos
df['dur_hour']=df['Duration'].apply(hour)
df['dur_min']=df['Duration'].apply(minutes)
```

```
df.head()
```

ation	Total_Stops	Additional_Info	Price	journey_day	journey_month	Dep_Time_hour	Dep_Time_min	Arrival_Time_hour	Arrival_Time_min	dur_hour	dur_min
1 50m	non-stop	No info	3897	24	3	22	20	1	10	2	50
1 25m	2 stops	No info	7662	5	1	5	50	13	15	7	25
9h 0m	2 stops	No info	13882	6	9	9	25	4	25	19	0
1 25m	1 stop	No info	6218	5	12	18	5	23	30	5	25
1 45m	1 stop	No info	13302	3	1	16	50	21	35	4	45

Y ahora borro la columna 'Duration'

```
#borro la columna 'Duration'
drop_col(df,'Duration')
```

```
df.dtypes
```

```
Airline              object
Source               object
Destination          object
Route                object
Total_Stops          object
Additional_Info      object
Price                 int64
journey_day           int64
journey_month         int64
Dep_Time_hour         int64
Dep_Time_min          int64
Arrival_Time_hour     int64
Arrival_Time_min      int64
dur_hour             object
dur_min              object
dtype: object
```

Cambio el tipo de dato para estas nuevas columnas que deben de ser entero

```
#cambio el tipo de dato para estas nuevas columnas que deben de ser entero
df['dur_hour'] = df['dur_hour'].astype(int)
df['dur_min'] = df['dur_min'].astype(int)
```

```
df.dtypes
```

```
Airline              object
Source               object
Destination          object
Route                object
Total_Stops          object
Additional_Info      object
Price                 int64
journey_day           int64
journey_month         int64
Dep_Time_hour         int64
Dep_Time_min          int64
Arrival_Time_hour     int64
Arrival_Time_min      int64
dur_hour              int32
dur_min               int32
dtype: object
```

Las variables categóricas son

```
column=[column for column in df.columns if df[column].dtype=='object']
column
```

```
['Airline', 'Source', 'Destination', 'Route', 'Total_Stops', 'Additional_Info']
```

Las variables cualitativas son

```
continuous_col =[column for column in df.columns if df[column].dtype!='object']
continuous_col
```

```
['Price',
 'journey_day',
 'journey_month',
 'Dep_Time_hour',
 'Dep_Time_min',
 'Arrival_Time_hour',
 'Arrival_Time_min',
 'dur_hour',
 'dur_min']
```

1.4. Featurea Engineering (FE)

En este punto, se realiza las diferentes operaciones necesarias para poder crear un modelo de predicción y que, además éste sea creado con el mejor rendimiento posible.

Para poder realizar un modelo de predicción, se necesita trabajar con datos numéricos. Para ello, los atributos categóricos deben ser transformados a numéricos mediante el uso de las diferentes técnicas existentes:

- OrdinalEncoder: Se emplea en 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 emplea en 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.

En base a los métodos existentes, se realiza los siguientes cambios en los diferentes atributos:

- Para la variable dependiente u objetivo, c1, así como para los atributos de entrada cuyos datos tienen un orden de relación, se emplea LabelEncoder.
- Para los atributos que poseen pocas clases y además tienen una mayor importancia en el significado del estudio, se emplea OneHotEncoder.

```
categorical =df[column]
categorical.head()
```

	Airline	Source	Destination	Route	Total_Stops	Additional_Info
0	IndiGo	Banglore	New Delhi	BLR → DEL	non-stop	No info
1	Air India	Kolkata	Banglore	CCU → IXR → BBI → BLR	2 stops	No info
2	Jet Airways	Delhi	Cochin	DEL → LKO → BOM → COK	2 stops	No info
3	IndiGo	Kolkata	Banglore	CCU → NAG → BLR	1 stop	No info
4	IndiGo	Banglore	New Delhi	BLR → NAG → DEL	1 stop	No info

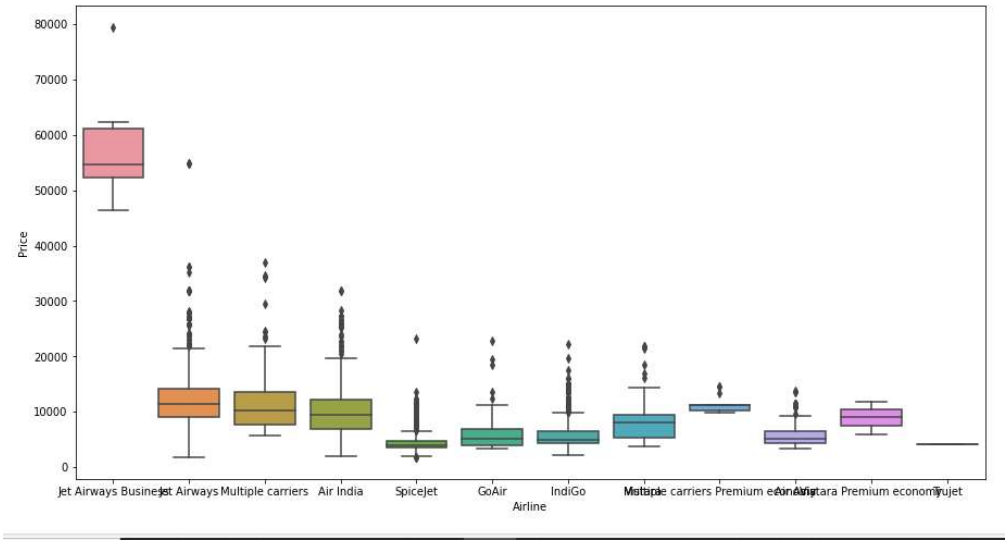
```
categorical['Airline'].value_counts()
```

```
Jet Airways                          3849
IndiGo                               2053
Air India                            1751
Multiple carriers                    1196
SpiceJet                              818
Vistara                               479
Air Asia                              319
GoAir                                 194
Multiple carriers Premium economy      13
Jet Airways Business                    6
Vistara Premium economy                 3
Trujet                                  1
Name: Airline, dtype: int64
```

Vamos a ver como se comporta ‘Airline ‘ con ‘Price’

```
plt.figure(figsize=(15,8))
sns.boxplot(x='Airline',y='Price',data=df.sort_values('Price',ascending=False))
```

```
<AxesSubplot:xlabel='Airline', ylabel='Price'>
```

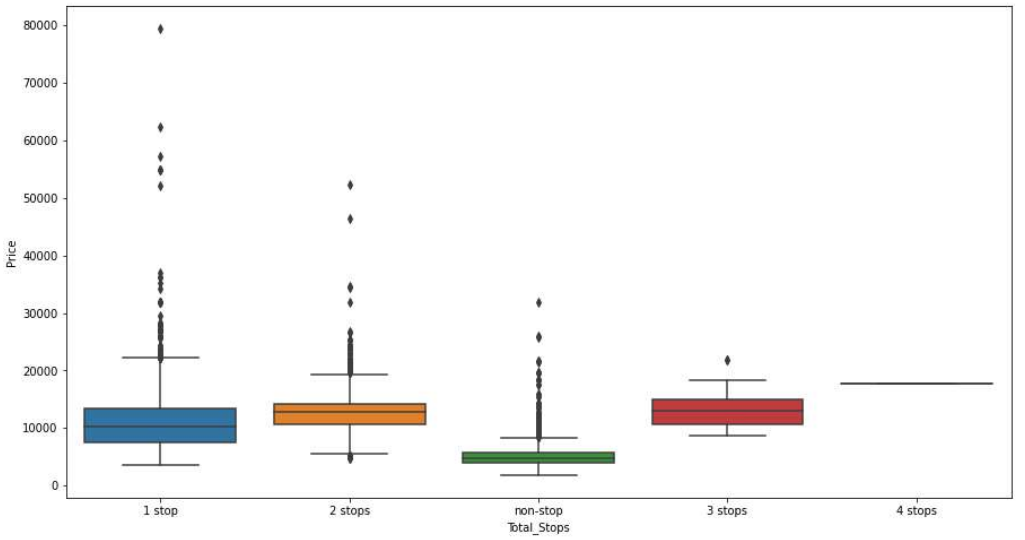


Podemos observar que Jet Airways Busines tiene los precios más altos, las demás aerolíneas tienen precios similares de media.

Vamos a ver como se comporta 'Total_Stops' con 'Price'

```
plt.figure(figsize=(15,8))
sns.boxplot(x='Total_Stops',y='Price',data=df.sort_values('Price',ascending=False))
```

```
<AxesSubplot:xlabel='Total_Stops', ylabel='Price'>
```



Los precios más bajos en media son de los que son vuelos directos aunque tiene valores atípicos

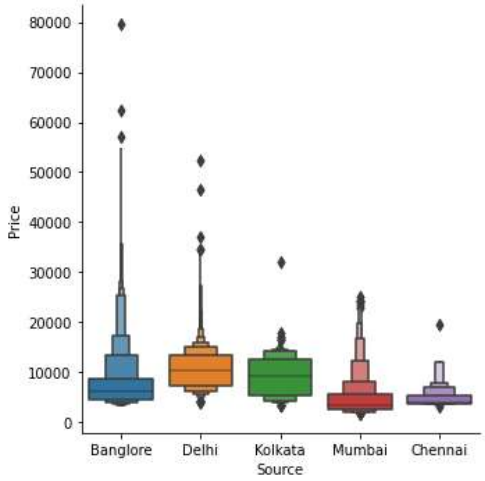
Como Airline son datos categóricos nominales los transformaré con OneHotEncoding

Comparo 'Source' con 'Price'

```
plt.figure(figsize=(15,15))
sns.catplot(x='Source',y='Price',data=df.sort_values('Price',ascending=Fa
```

```
<seaborn.axisgrid.FacetGrid at 0x211aec27fa0>

<Figure size 1080x1080 with 0 Axes>
```

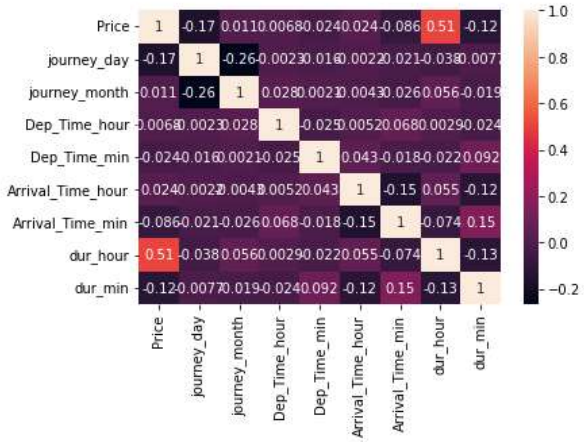


Voy a mirar las correlaciones

```
sns.heatmap(df.corr(),annot=True)
```

```
C:\Users\elena\AppData\Local\Temp\ipykernel_12488\4277794465.py
e.corr is deprecated. In a future version, it will default to F
only to silence this warning.
  sns.heatmap(df.corr(),annot=True)
```

```
<AxesSubplot:>
```



No hay correlaciones que sean significativas de las variables

Codifico la columna Source que también es categórico nominal

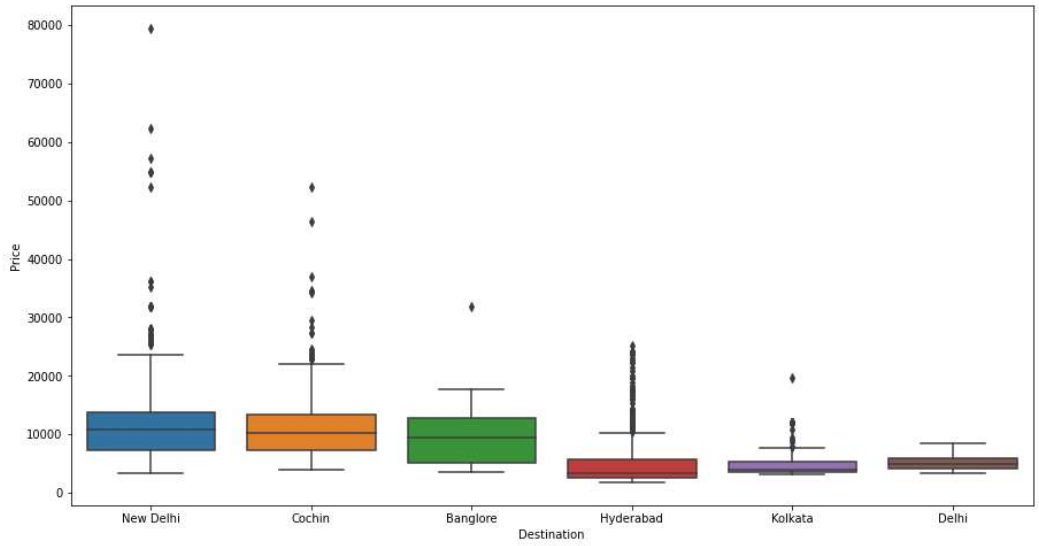
```
#codifico la columna Source que también es categórico nomninal
source=pd.get_dummies(categorical['Source'],drop_first=True)
source.head()
```

	Chennai	Delhi	Kolkata	Mumbai
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	0

Ahora quiero ver cómo se comportan los destinos y precio ,'Destination' y 'Price'

```
plt.figure(figsize=(15,8))
sns.boxplot(x='Destination',y='Price',data=df.sort_values('Price',ascending=False))
```

```
<AxesSubplot:xlabel='Destination', ylabel='Price'>
```



Codifico la columna Destination que es categórica nominal

```
#codifico la columna Destination que es categórica nominal
destination=pd.get_dummies(categorical['Destination'],drop_first=True)
destination.head()
```

	Cochin	Delhi	Hyderabad	Kolkata	New Delhi
0	0	0	0	0	1
1	0	0	0	0	0
2	1	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	1

Ahora separaré la columna Route en cinco columnas separando así las rutas (tenemos hasta stops)

```
categorical.head()
```

	Airline	Source	Destination	Route	Total_Stops	Additional_Info	Route1	Route2	Route3	Route4	Route5
0	IndiGo	Banglore	New Delhi	BLR → DEL	non-stop	No info	BLR	DEL	NaN	NaN	NaN
1	Air India	Kolkata	Banglore	CCU → IXR → BBI → BLR	2 stops	No info	CCU	IXR	BBI	BLR	NaN
2	Jet Airways	Delhi	Cochin	DEL → LKO → BOM → COK	2 stops	No info	DEL	LKO	BOM	COK	NaN
3	IndiGo	Kolkata	Banglore	CCU → NAG → BLR	1 stop	No info	CCU	NAG	BLR	NaN	NaN
4	IndiGo	Banglore	New Delhi	BLR → NAG → DEL	1 stop	No info	BLR	NAG	DEL	NaN	NaN

Elimino la variable 'Route'

Reviso si con los cambios hay datos vacíos

```
categorical.isnull().sum()
```

```
Airline              0
Source               0
Destination          0
Total_Stops          0
Additional_Info      0
Route1               0
Route2               0
Route3            3491
Route4            9116
Route5           10636
dtype: int64
```

Tenemos dastos vacíos en las columnas donde no hay tantas paradas por lo tanto cambiaremos esos valores nulos por None para poder trabajar estas columnas

```
categorical.columns
```

```
Index(['Airline', 'Source', 'Destination', 'Total_Stops', 'Additional_Info',
       'Route1', 'Route2', 'Route3', 'Route4', 'Route5'],
      dtype='object')
```

```
for i in ['Route3', 'Route4', 'Route5']:
    categorical[i].fillna('None',inplace=True)
```

```
categorical.isnull().sum()
```

```
Airline          0
Source           0
Destination      0
Total_Stops      0
Additional_Info  0
Route1           0
Route2           0
Route3           0
Route4           0
Route5           0
dtype: int64
```

```
: for i in categorical.columns:
    print('{} tiene un  total {} de categorias'.format(i,len(categorical[i].value_counts())))
```

```
Airline tiene un  total 12 de categorias
Source tiene un  total 5 de categorias
Destination tiene un  total 6 de categorias
Total_Stops tiene un  total 5 de categorias
Additional_Info tiene un  total 10 de categorias
Route1 tiene un  total 5 de categorias
Route2 tiene un  total 45 de categorias
Route3 tiene un  total 30 de categorias
Route4 tiene un  total 14 de categorias
Route5 tiene un  total 6 de categorias
```

Vamos a codificar estas variables de rutas nuevas creadas a partir de la columna Route inicial en variables cualitativas ordinales

```
# Vamos a codificar estas variables de rutas nuevas creadas a partir
#de la columna Route inicial en variables cualitativas ordinales

encoder = LabelEncoder()
```

```
for i in ['Route1', 'Route2', 'Route3', 'Route4', 'Route5']:
    categorical[i]=encoder.fit_transform(categorical[i])
```

```
categorical.head()
```

	Airline	Source	Destination	Total_Stops	Additional_Info	Route1	Route2	Route3	Route4	Route5
0	IndiGo	Banglore	New Delhi	non-stop	No info	0	13	29	13	5
1	Air India	Kolkata	Banglore	2 stops	No info	2	25	1	3	5
2	Jet Airways	Delhi	Cochin	2 stops	No info	3	32	4	5	5
3	IndiGo	Kolkata	Banglore	1 stop	No info	2	34	3	13	5
4	IndiGo	Banglore	New Delhi	1 stop	No info	0	34	8	13	5

Elimino la columna Aditional_Info dado que no es necesaria para el análisis

Codificamos Total stops para poder trabajar con ella

```
categorical['Total_Stops'].unique()
```

```
array(['non-stop', '2 stops', '1 stop', '3 stops', '4 stops'],
      dtype=object)
```

```
# Codificamos Total stops para poder trabajar con ella
dict={'non-stop':0, '2 stops':2, '1 stop':1, '3 stops':3, '4 stops':4}
categorical['Total_Stops']=categorical['Total_Stops'].map(dict)
```

Elimino del dataframe categorical, las columnas Source ,Destination y Airline dado que ya las tenemos codificadas

Finalmente normalizo los valores

```
from sklearn.preprocessing import MinMaxScaler
```

```
#Inicializar el escalador
scaler = MinMaxScaler()

#Normalizar los datos
final_df_normalizado = scaler.fit_transform(final_df)

#Mostrar los registros normalizados
final_df=pd.DataFrame( final_df_normalizado, columns=columnas )
final_df
```

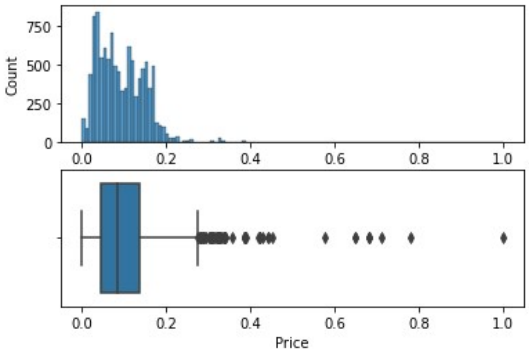
	Total_Stops	Route1	Route2	Route3	Route4	Route5	Air India	GoAir	IndiGo	Jet Airways	...	New Delhi	Price	journey_day	journey_month	Dep_Time_I
0	0.00	0.00	0.295455	1.000000	1.000000	1.0	0.0	0.0	1.0	0.0	...	1.0	0.027497	0.875000	0.181818	0.95
1	0.50	0.50	0.568182	0.034483	0.230769	1.0	1.0	0.0	0.0	0.0	...	0.0	0.075920	0.083333	0.000000	0.21
2	0.50	0.75	0.727273	0.137931	0.384615	1.0	0.0	0.0	0.0	1.0	...	0.0	0.155917	0.125000	0.727273	0.39
3	0.25	0.50	0.772727	0.103448	1.000000	1.0	0.0	0.0	1.0	0.0	...	0.0	0.057348	0.083333	1.000000	0.78
4	0.25	0.00	0.772727	0.275862	1.000000	1.0	0.0	0.0	1.0	0.0	...	1.0	0.148457	0.000000	0.000000	0.69
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
10677	0.00	0.50	0.113636	1.000000	1.000000	1.0	0.0	0.0	0.0	0.0	...	0.0	0.030198	0.041667	0.727273	0.82
10678	0.00	0.50	0.113636	1.000000	1.000000	1.0	1.0	0.0	0.0	0.0	...	0.0	0.030687	1.000000	0.272727	0.86
10679	0.00	0.00	0.295455	1.000000	1.000000	1.0	0.0	0.0	0.0	1.0	...	0.0	0.070351	1.000000	0.272727	0.34
10680	0.00	0.00	0.295455	1.000000	1.000000	1.0	0.0	0.0	0.0	0.0	...	1.0	0.140046	0.000000	0.000000	0.47
10681	0.50	0.75	0.363636	0.137931	0.384615	1.0	1.0	0.0	0.0	0.0	...	0.0	0.128535	0.083333	0.727273	0.43

10682 rows × 35 columns

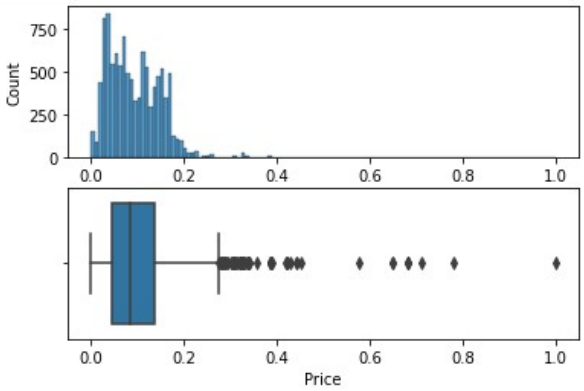
Revisamos si hay outliers

```
def plot(data,col):
    fig,(ax1,ax2)=plt.subplots(2,1)
    sns.histplot(data[col],ax=ax1)
    sns.boxplot(data[col],ax=ax2)
```

```
plot(final_df,'Price')
```



Hay outliers en 'Price' , así que los reemplazaremos con la mediana



1.5. Data Modeling

Para entrenar el modelo, es necesario realizar una serie de segmentaciones en los datos que se tiene. En concreto, se realiza un total de tres particiones de los datos, siendo las siguientes:

- Test:95% de los datos con los que se realiza el modelo. Sus valores son generalmente del 70 % para el entrenamiento del mismo y el 30 % para la fase de test
- Prueba 5% de los datos que se reserva para la validación del modelo. Estos son empleados para realizar predicciones con el modelo ya generado y testeado.

Separamos el dataset en X e Y columnas

```
#Como lo que queremos es predecir el precio, está será nuestra y
X=final_df.drop('Price',axis=1)
y=df['Price']
```

Sacamos los pesos de las columnas

```
from sklearn.feature_selection import mutual_info_classif
```

```
mutual_info_classif(X,y)
```

```
array([2.14821257e+00, 2.06098863e+00, 2.97906522e+00, 2.57974802e+00,
       1.97793543e+00, 1.79544416e+00, 7.44664527e-01, 8.79779559e-02,
       6.73472571e-01, 9.37407824e-01, 0.00000000e+00, 5.72913847e-01,
       3.05791764e-04, 3.32632390e-01, 1.04906185e-03, 2.20315801e-01,
       1.06184245e-02, 1.59870663e-01, 1.55156094e+00, 8.73808360e-01,
       3.04899410e-01, 1.56740949e+00, 4.07826304e-01, 2.91375483e-01,
       1.70709926e-01, 3.88107158e-01, 1.07291411e+00, 1.07979753e+00,
       1.51131727e+00, 1.30424019e+00, 1.89808872e+00, 1.64976835e+00,
       1.75291735e+00, 1.13183229e+00])
```

```
#Sacamos los "pesos" de las columnas
imp = pd.DataFrame(mutual_info_classif(X,y),index=X.columns)
imp
```

	importance
Route2	2.960606
Route3	2.620579
Total_Stops	2.162049
Route1	2.014494
Route4	1.962673
Arrival_Time_hour	1.893668
Route5	1.819370
dur_hour	1.748024
Arrival_Time_min	1.623404
Cochin	1.552030
Dep_Time_hour	1.541916
Delhi	1.510148
Dep_Time_min	1.271375
dur_min	1.134130
journey_month	1.093604
journey_day	1.054388
Jet Airways	0.909142
Kolkata	0.859325
Air India	0.752321
IndiGo	0.667049
Multiple carriers	0.578274
Delhi	0.403559
New Delhi	0.374641
SpiceJet	0.317984
Hyderabad	0.284147
Mumbai	0.283564
Vistara	0.220110
Chennai	0.177378
Kolkata	0.171551
GoAir	0.087877
Vistara Premium economy	0.004971
Jet Airways Business	0.000000
Multiple carriers Premium economy	0.000000
Trujet	0.000000

```
# spiliting the dataset
from sklearn.model_selection import train_test_split
#separo los datos para las pruebas
X_train1,X_prueba,y_train1,y_prueba = train_test_split(X,y,test_size=0.05,random_state=786)
#Separo los datos para realizar los modelos

X_train,X_test,y_train,y_test = train_test_split(X_train1,y_train1,test_size=0.20,random_state=123)
```

```
print("X_train", len(X_train))
print("X_test", len(X_test))
print("X_prueba", len(X_prueba))
```

```
X_train 8117
X_test 2030
X_prueba 535
```

2. Generar el modelo

2.1. Comparación de modelos

```
from sklearn.metrics import r2_score,mean_absolute_error,mean_squared_error
def predict(ml_model):
    print('Model is: {}'.format(ml_model))
    model= ml_model.fit(X_train,y_train)
    print("Training score: {}".format(model.score(X_train,y_train)))
    predictions = model.predict(X_test)
    print("Predictions are: {}".format(predictions))
    print('\n')
    r2score=r2_score(y_test,predictions)
    print("r2 score is: {}".format(r2score))

    print('MAE:{}'.format(mean_absolute_error(y_test,predictions)))
    print('MSE:{}'.format(mean_squared_error(y_test,predictions)))
    print('RMSE:{}'.format(np.sqrt(mean_squared_error(y_test,predictions))))

    sns.histplot(y_test-predictions)
```

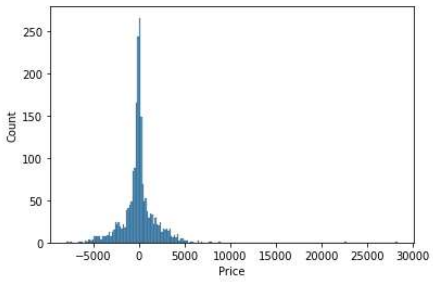
```
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import GradientBoostingRegressor,RandomForestRegressor
```

- RandomForestRegressor

```
predict(RandomForestRegressor())
```

```
Model is: RandomForestRegressor()
Training score: 0.9524370177970066
Predictions are: [ 5001.58        4305.02       10483.64       ... 12500.65266667
 11613.67266667  2018.11666667]


r2 score is: 0.8353885834710345
MAE:1146.9003437201006
MSE:3663272.4130027946
RMSE:1913.9677147232119
```



From the graph,it is clear that we predicted 83,32% correctly.

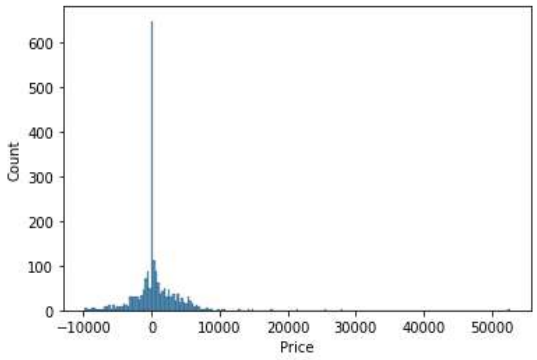
- LogisticRegression

```
predict(LogisticRegression())
```

```
Model is: LogisticRegression()

Training score: 0.3145250708389799
Predictions are: [ 4409  3943 10151 ... 13377 10262  2017]

r2 score is: 0.5375594642617759
MAE:1835.9418719211822
MSE:10291179.633497536
RMSE:3207.986850580522
```

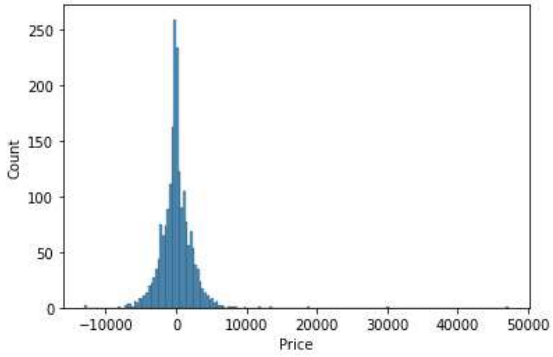


- KNeighborsRegressor

```
predict(KNeighborsRegressor())
```

```
Model is: KNeighborsRegressor()
Training score: 0.8074769774259778
Predictions are: [ 4564.   4344.4  9870.2 ...  9567.6 12042.8  2143.4]

r2 score is: 0.7261966614775397
MAE:1518.195369458128
MSE:6093236.044906404
RMSE:2468.448104560111
```

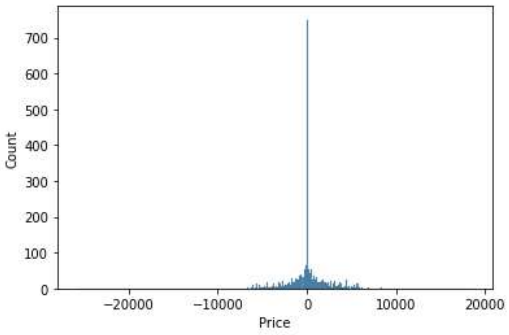


- DecisionTreeRegressor

```
predict(DecisionTreeRegressor())
```

```
Model is: DecisionTreeRegressor()
Training score: 0.9711544864808386
Predictions are: [ 5989.  4423. 12797. ... 13377. 10262.  2017.]


r2 score is: 0.7392778999163776
MAE:1331.1500328407224
MSE:5802125.374022442
RMSE:2408.7601321058187
```

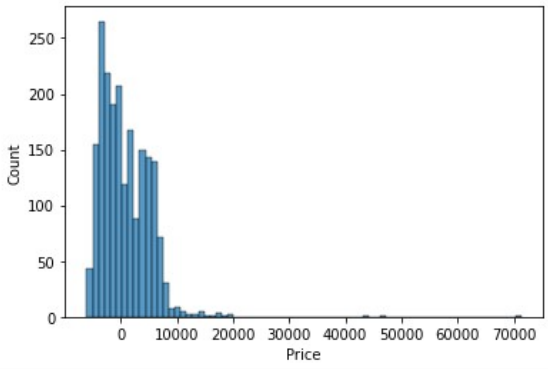


- SVR

```
from sklearn.svm import SVR
predict(SVR())
```

```
Model is: SVR()
Training score: 0.07037404612204323
Predictions are: [7995.36114404 7606.97327808 8393.48046418 ... 8474.3278232  8847.25680558
 7813.96671889]


r2 score is: 0.0728629267706592
MAE:3354.9573892899143
MSE:20632564.4663629
RMSE:4542.30827513533
```

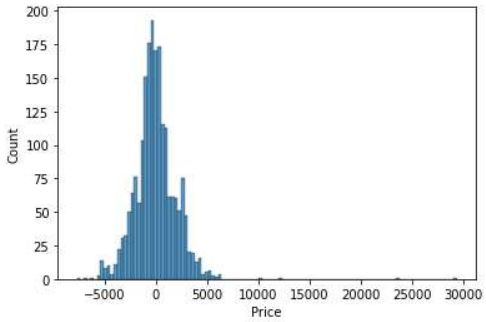


- GradientBoostingRegressor

```
: predict(GradientBoostingRegressor())

Model is: GradientBoostingRegressor()
Training score: 0.8006729534835295
Predictions are: [ 4624.46384359  4616.1600259  11930.85119909 ... 10324.5576977
 11595.4349471   2592.47265071]


r2 score is: 0.8023991108366071
MAE:1462.7657753820197
MSE:4397422.13098389
RMSE:2097.0031308951093
```



El mejor modelo es

Model is: RandomForestRegressor()

Training score: 0.9524370177970066

r2 score is: 0.8353885834710345

MAE:1146.9003437201006

MSE:3663272.4130027946

RMSE:1913.9677147232119

2.2. Ajustar el modelo

Para mejorar el rendimiento del modelo, es importante ajustar el mayor número de parámetros posibles.

Mejoramos el modelo

```
: from sklearn.model_selection import RandomizedSearchCV
```

```
: random_grid = {
    'n_estimators' : [100, 120, 150, 180, 200,220],
    'max_features':['auto','sqrt'],
    'max_depth':[5,10,15,20],
    }
```

```
rf=RandomForestRegressor()
rf_random=RandomizedSearchCV(estimator=rf,param_distributions=random_grid,cv=3,verbose=2,n_jobs=-1,)

rf_random.fit(X_train,y_train)

# best parameter
rf_random.best_params_
```

```
Fitting 3 folds for each of 10 candidates, totalling 30 fits
```

```
{'n_estimators': 180, 'max_features': 'auto', 'max_depth': 15}
```

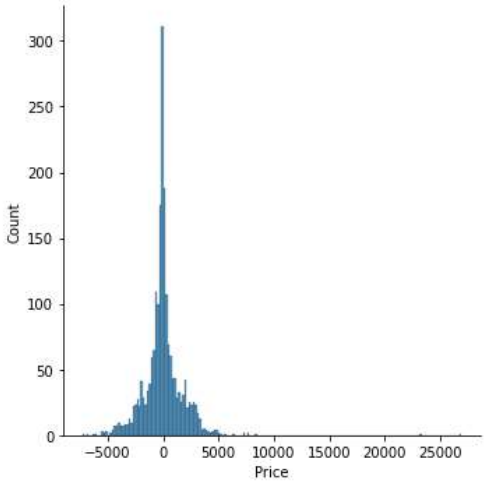
```
# best parameter
rf_random.best_params_
```

```
{'n_estimators': 180, 'max_features': 'auto', 'max_depth': 15}
```

```
#predicting the values
prediction = rf_random.predict(X_test)

#distribution plot between actual value and predicted value
sns.displot(y_test-prediction)
```

```
<seaborn.axisgrid.FacetGrid at 0x21606671ac0>
```




```
r2_score(y_test,prediction)
```

```
0.855009262717046
```

Despues de mejorar el modelo accuracy aumenta!

2.3. 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

```
#Guardamos el modelo

import joblib
nombre='modelos/RandomForestRegresor.pkl'
joblib.dump(rf,nombre)
```

```
['modelos/RandomForestRegresor.pkl']
```

3. Conclusiones

Durante la investigación, hemos desarrollado con éxito un sistema de predicción de precios de vuelos 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 precios de vuelos para identificar patrones y tendencias importantes. Esto ha llevado a una mejor comprensión de los factores que influyen en los precios de vuelos.

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 el modelo predictivo que construi para el conjuntode columnas de datos son efectivos para de precios de vuelos

Este enfoque permite una comprensión detallada de los factores específicos que pueden influir en de precios de vuelos,.

Con el TFM he desarrollado con éxito un sistema de predicción de de precios de vuelos que ayuda a ayudar a planificar un vuelo, eligiendo el mejor momento para volar para poder ahorrar en los costes además de tiempo. 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

4. Anexos

Anexo A .....https://github.com/ElenaArbiol/TFM