UMBC

FLIGHT FARE PREDICTION

CAPSTONE PROJECT

FLIGHT FARE PREDICTION

UMBC

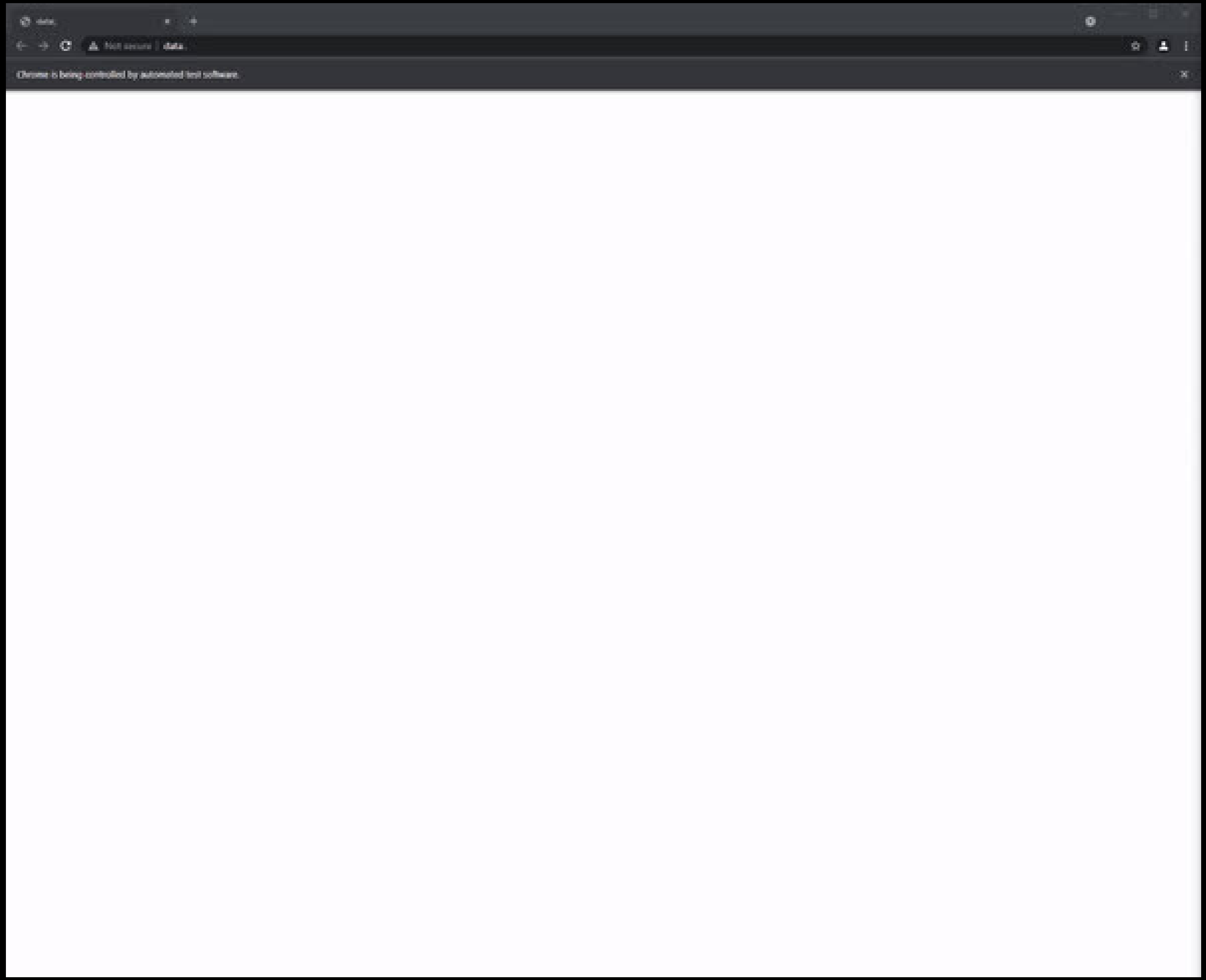
- A tool that estimates Flight Prices to help users look for best prices when booking flight tickets.
- Engineered features from the Departure Time, Date of Journey, to quantify the data and make it more understandable.
- Optimized multiple Regression models using GridsearchCV to reach the best model.
- Built a client facing UI using Streamlit

KAYAK SCRAPPER

Creating Datasets via scrapping data from Kayak

- The dataset will be created by scraping web content from different travel websites.

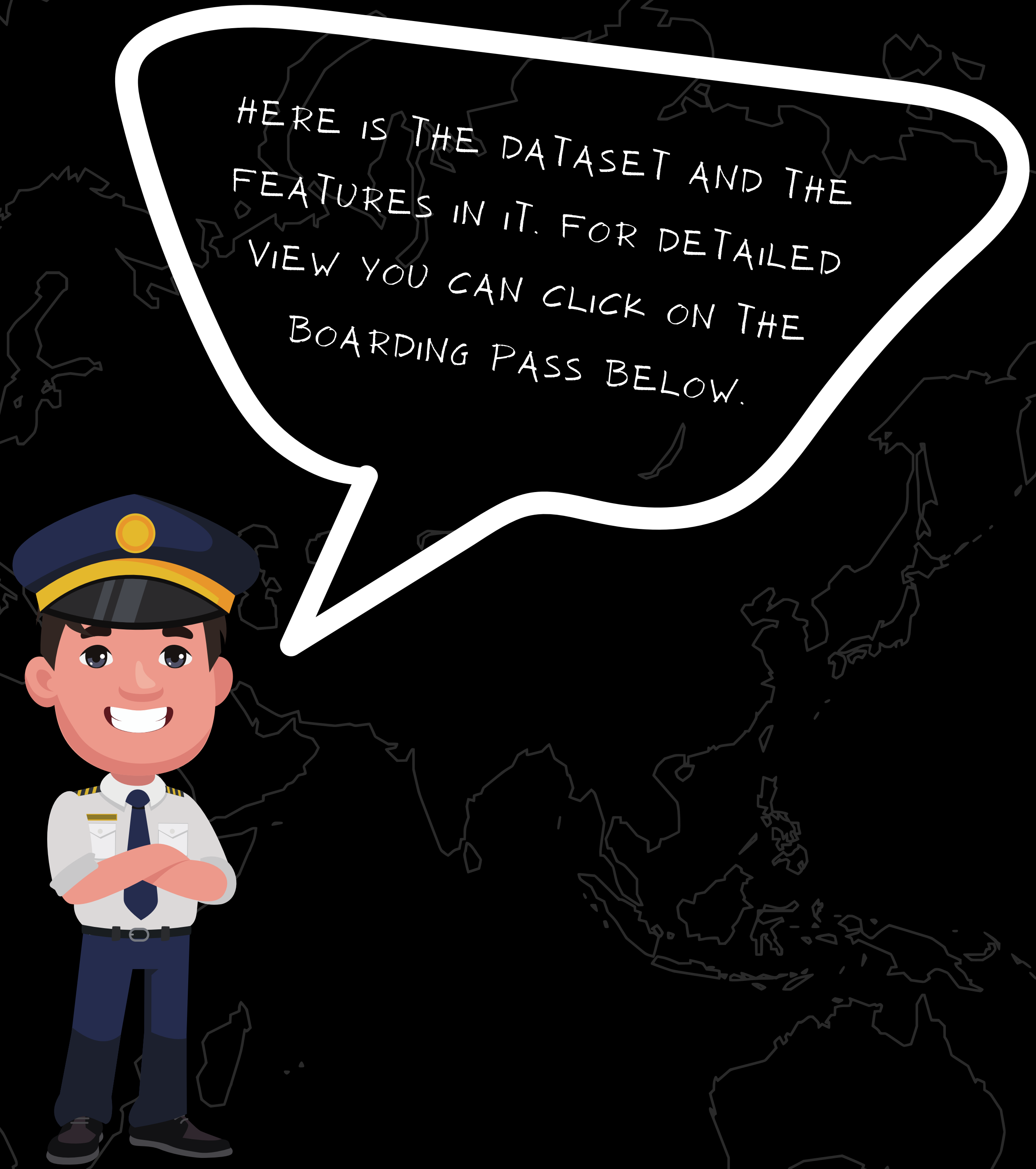
- The Search Engine Results - Flights & Tickets Keywords Dataset will also be used as it provides Rankings for world top destinations on Google.

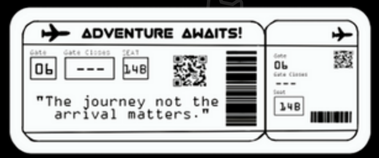


Chrome is being controlled by automated test software.

DATASET

- Size of test set: 2671 records
- FEATURES: Airline: The name of the airline.
- Date_of_Journey: The date of the journey
- Source: The source from which the service begins.
- Destination: The destination where the service ends.
- Route: The route taken by the flight to reach the destination.
- Dep_Time: The time when the journey starts from the source.
- Arrival_Time: Time of arrival at the destination.
- Duration: Total duration of the flight.
- Total_Stops: Total stops between the source and destination.
- Additional_Info: Additional information about the flight
- Price: The price of the ticket

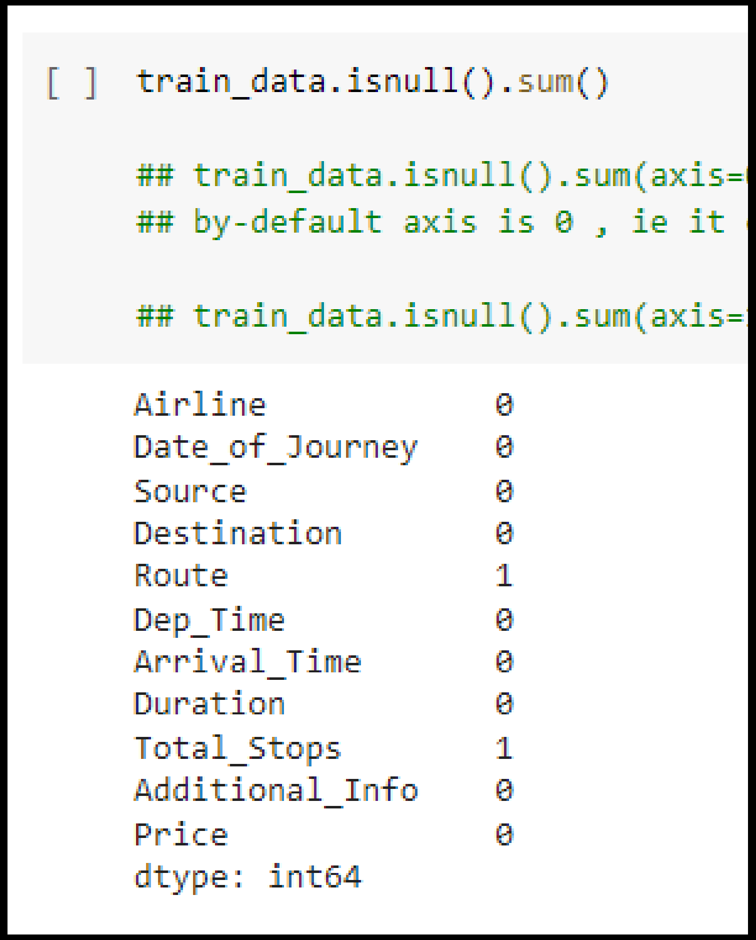

HERE IS THE DATASET AND THE FEATURES IN IT. FOR DETAILED VIEW YOU CAN CLICK ON THE BOARDING PASS BELOW.



PREPROCESSING DATA

Dealing with missing values

- For data manipulation, numerical computation, and visualization we imported pandas, NumPy, seaborn, and matplotlib library
- Reading data and saving it into the train_data variable.
- Previewing data by calling head function.



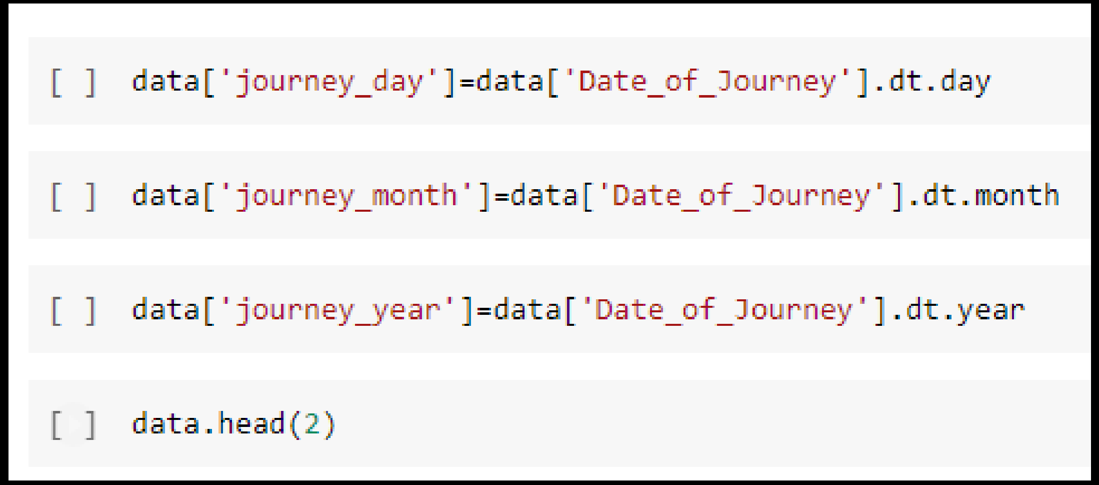
```
[ ]  train_data.isnull().sum()

     ## train_data.isnull().sum(axis=
     ## by-default axis is 0 , ie it

     ## train_data.isnull().sum(axis=

     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
```

PREPROCESSING DATA

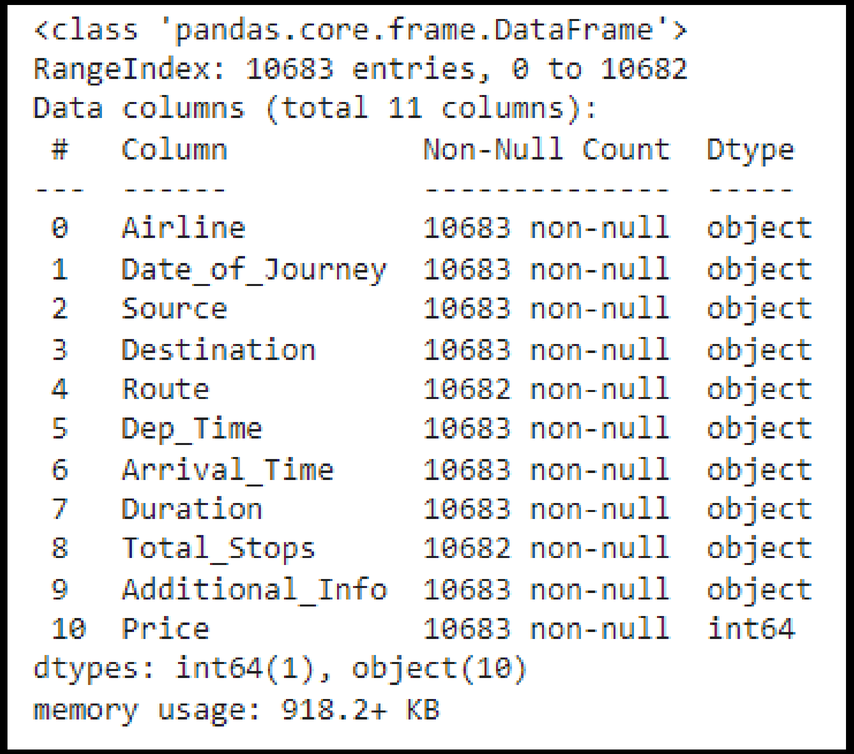
Cleaning data for analysis and modeling purpose

- We can see that Date_of_Journey is a object data type, Therefore, we have to convert this datatype into timestamp because our model will not be able to understand these string values,it just understand Time-stamp.

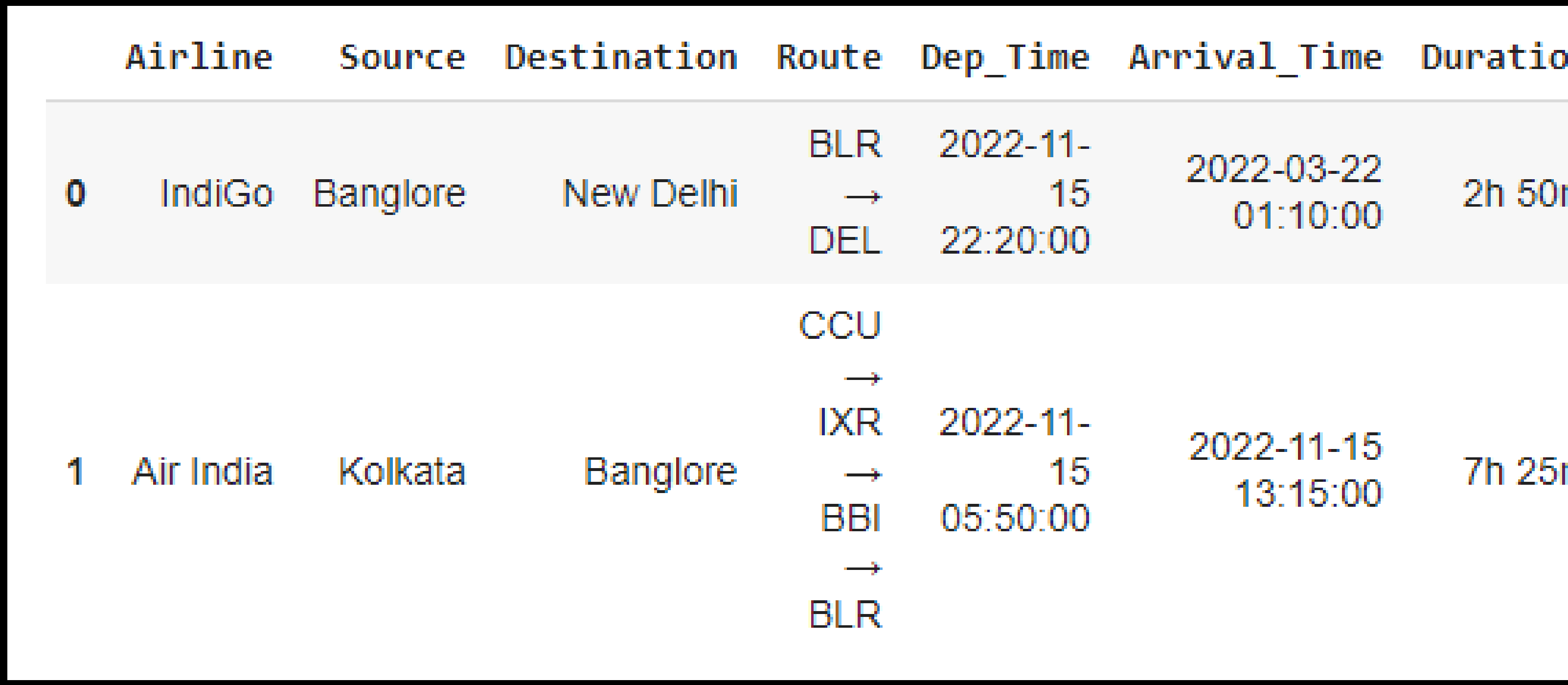
- For this we require pandas to_datetime to convert object data type to datetime dtype.



```
[ ]  data['journey_day']=data['Date_of_Journey'].dt.day

[ ]  data['journey_month']=data['Date_of_Journey'].dt.month

[ ]  data['journey_year']=data['Date_of_Journey'].dt.year

[ ]  data.head(2)
```

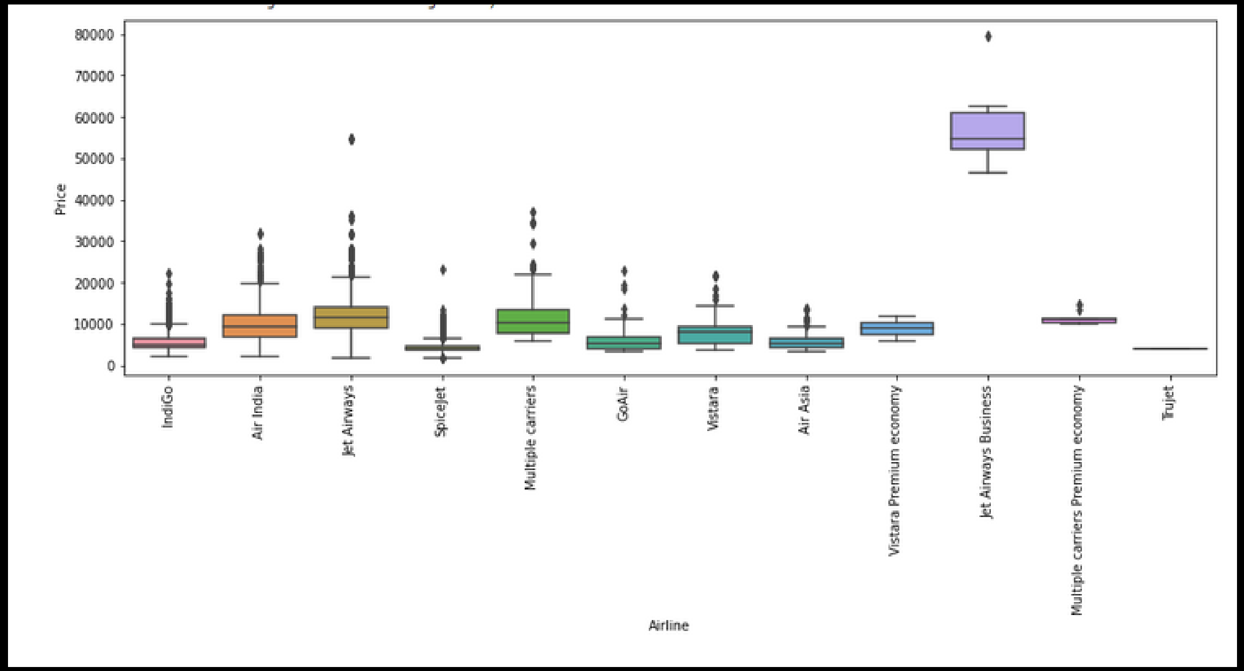


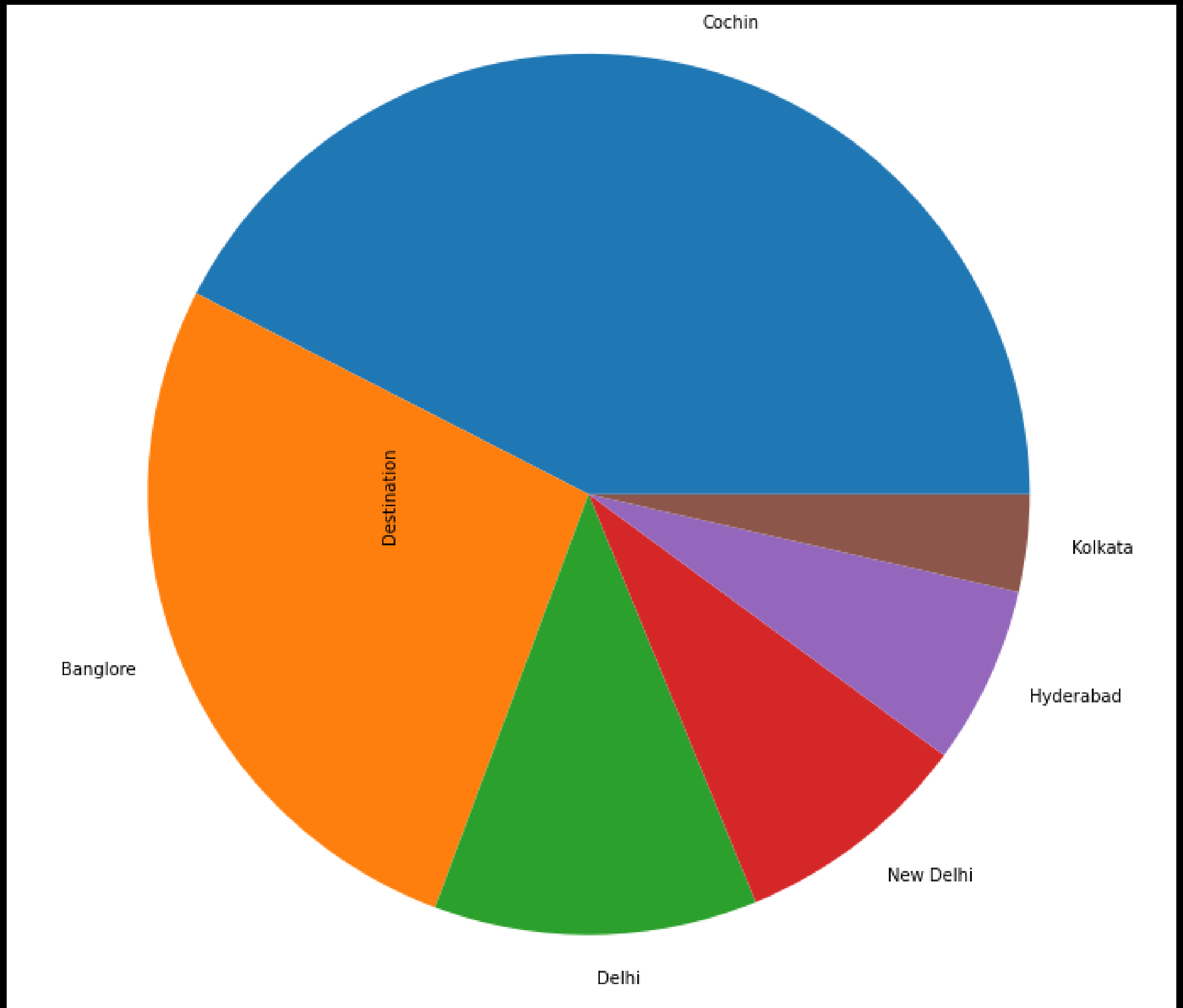
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10683 entries, 0 to 10682
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   Airline          10683 non-null  object
 1   Date_of_Journey  10683 non-null  object
 2   Source           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
```

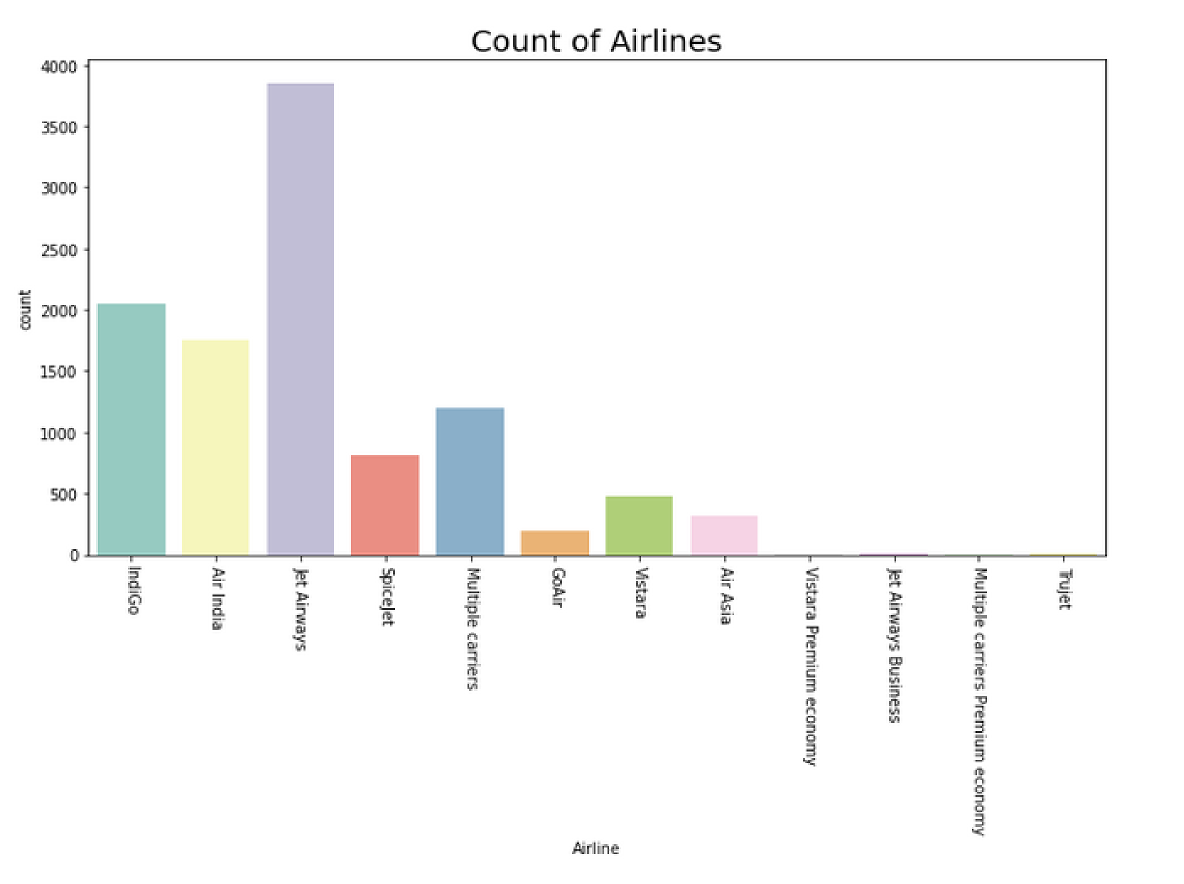


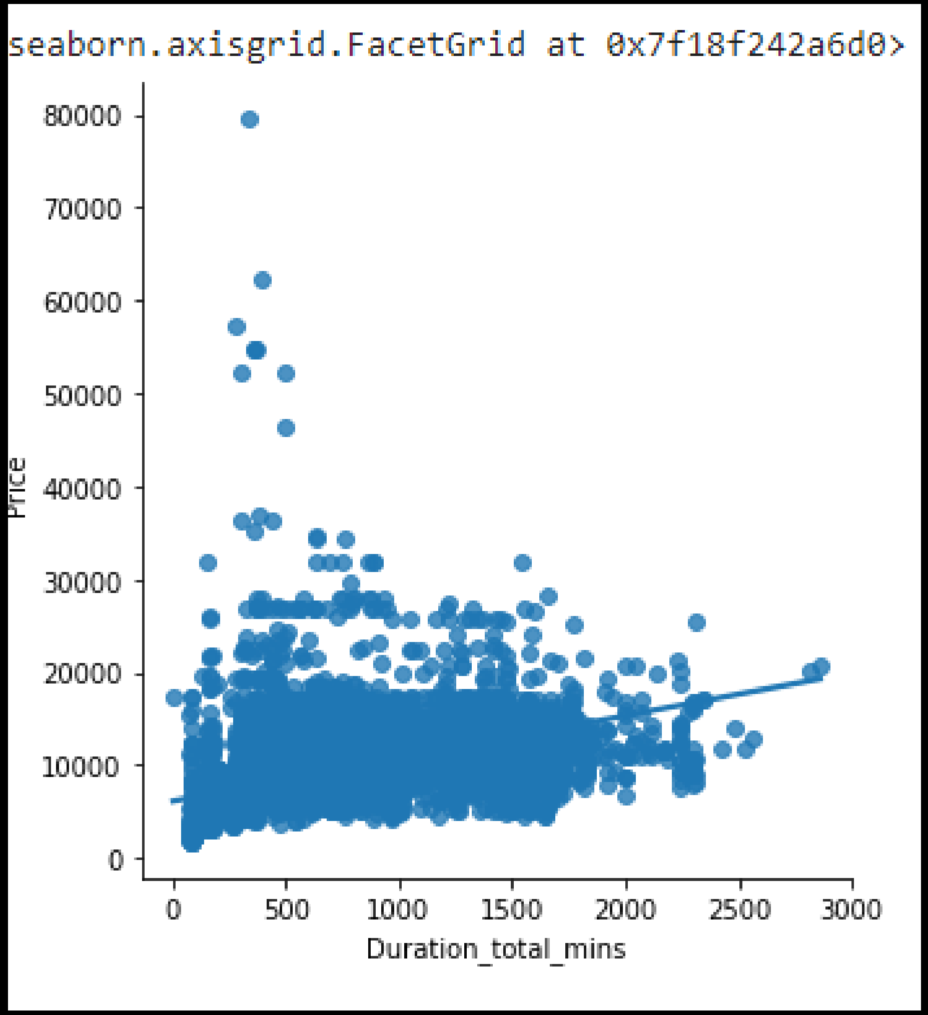
	Airline	Source	Destination	Route	Dep_Time	Arrival_Time	Duratio
0	IndiGo	Banglore	New Delhi	BLR → DEL	2022-11-15 22:20:00	2022-03-22 01:10:00	2h 50
1	Air India	Kolkata	Banglore	CCU → IXR → BBI → BLR	2022-11-15 05:50:00	2022-11-15 13:15:00	7h 25

EXPLORATORY DATA ANALYSIS







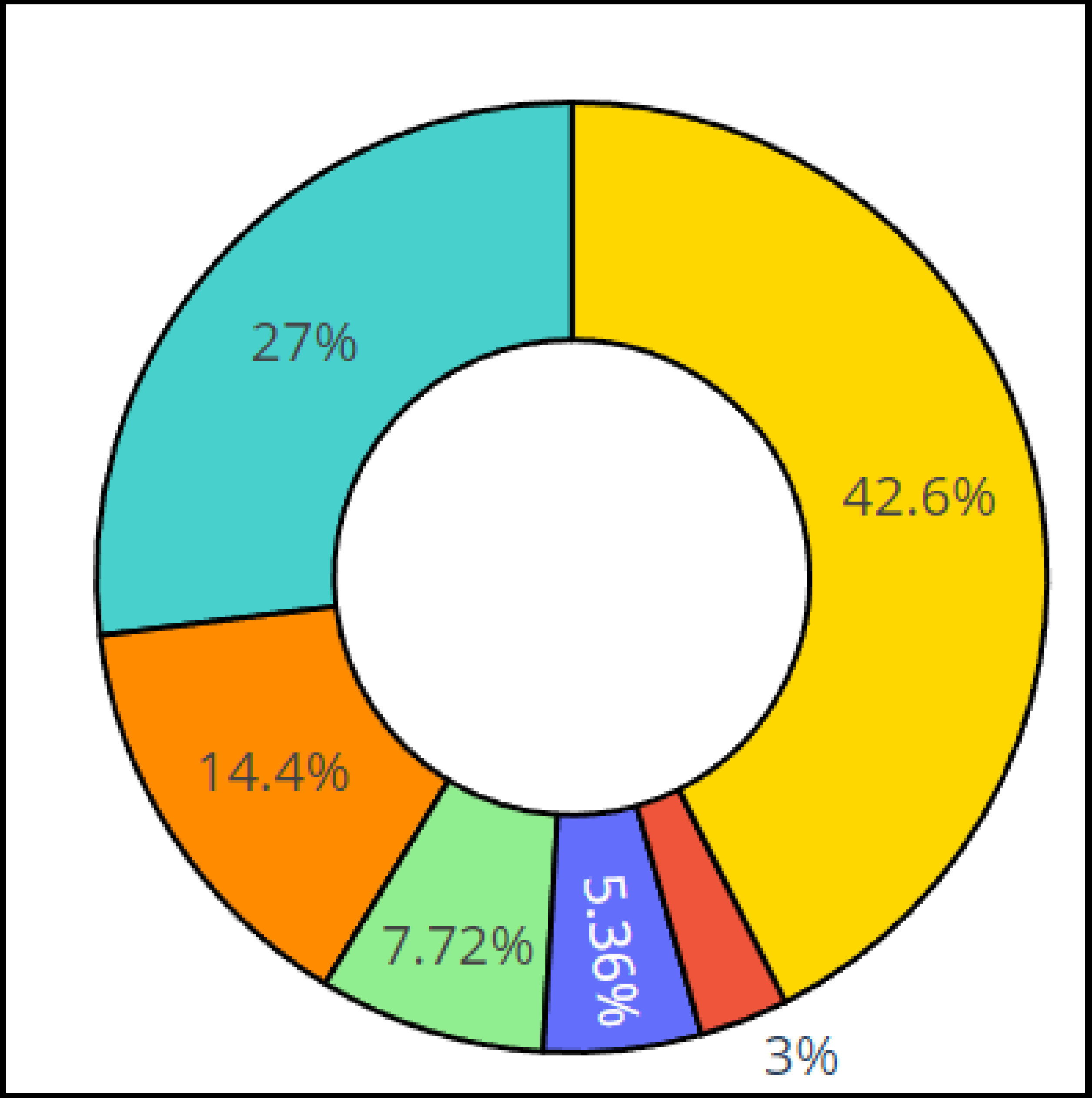

```
seaborn.axisgrid.FacetGrid at 0x7f18f242a6d0>
```

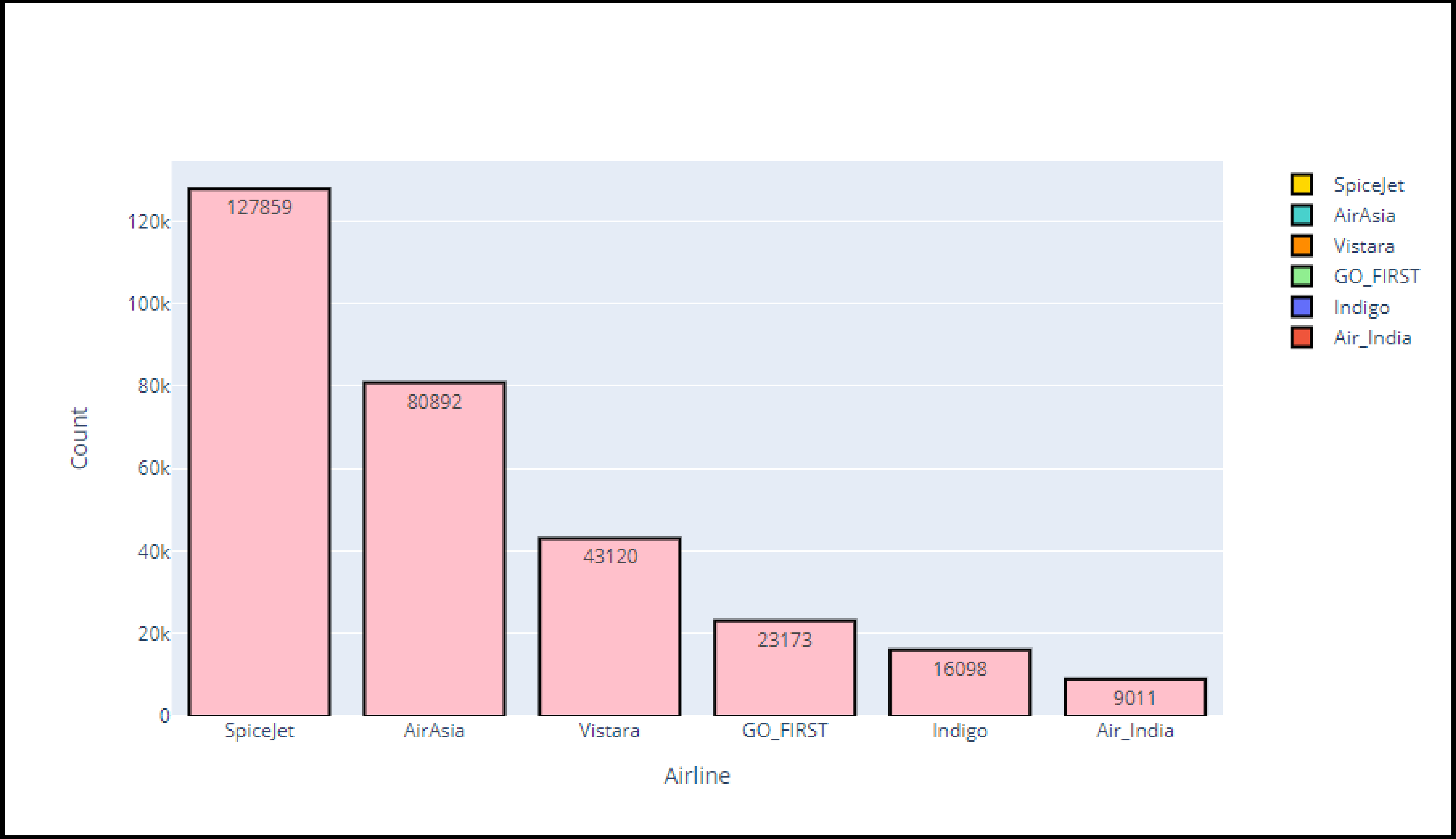
EXPLORATORY DATA ANALYSIS

- Does price vary with Airlines?
- How is the price affected when tickets are bought in just 1 or 2 days before departure?
- Does ticket price change based on the departure time and arrival time?
- How the price changes with change in Source and Destination?
- How does the ticket price vary between Economy and Business class?
- Which features have the most impact on predicting flight price?
- Which features have the most impact on predicting flight price?
- Which model is the best for predicting flight price?

EXPLORATORY DATA ANALYSIS

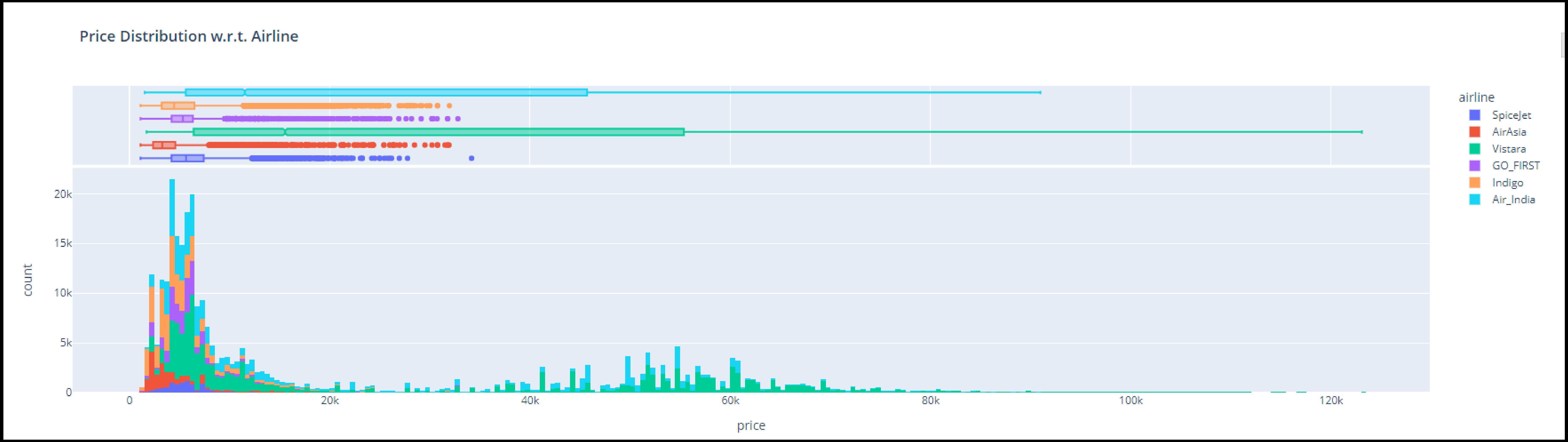
Airline Distribution





EXPLORATORY DATA ANALYSIS

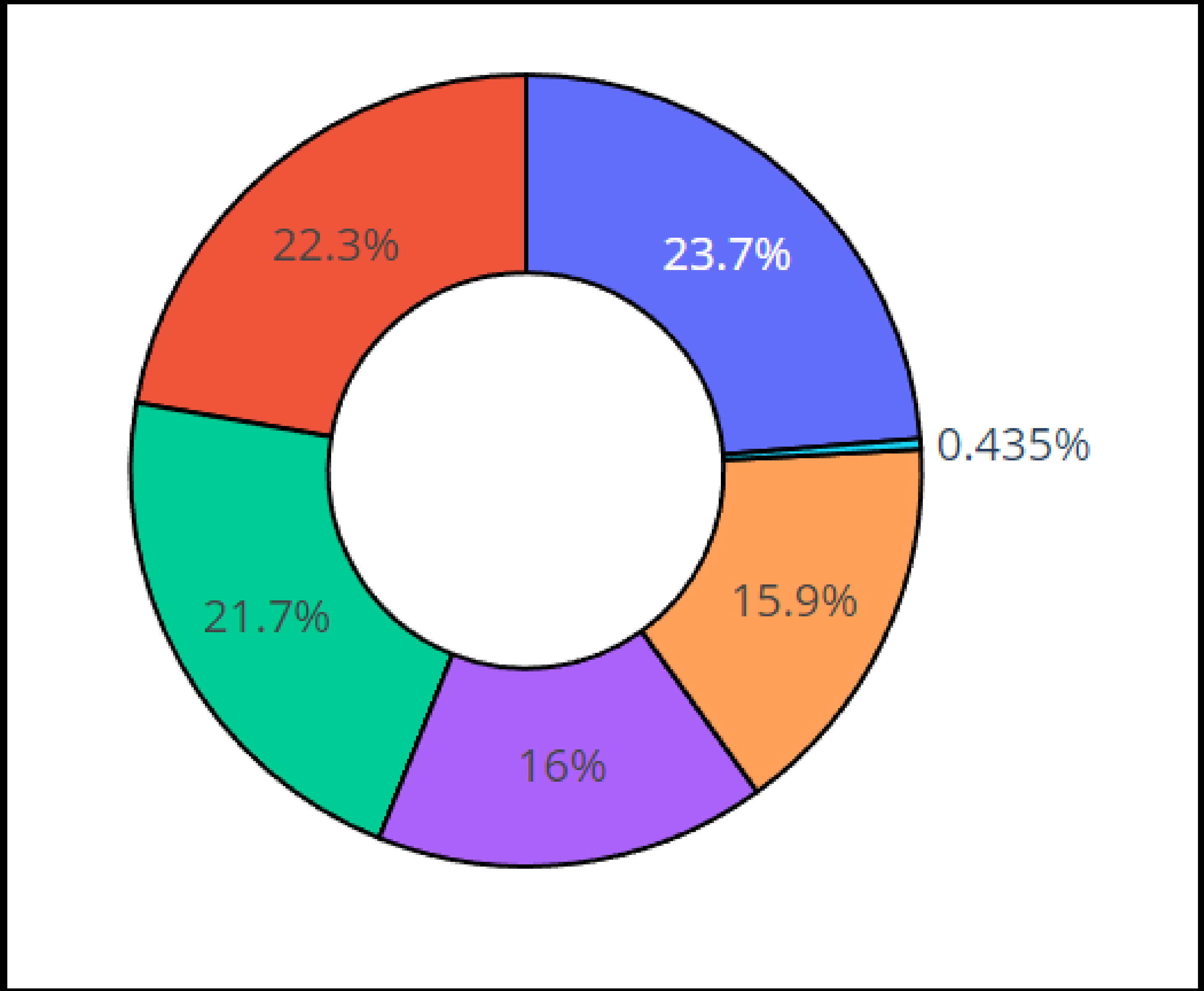
Does price vary with Airlines?

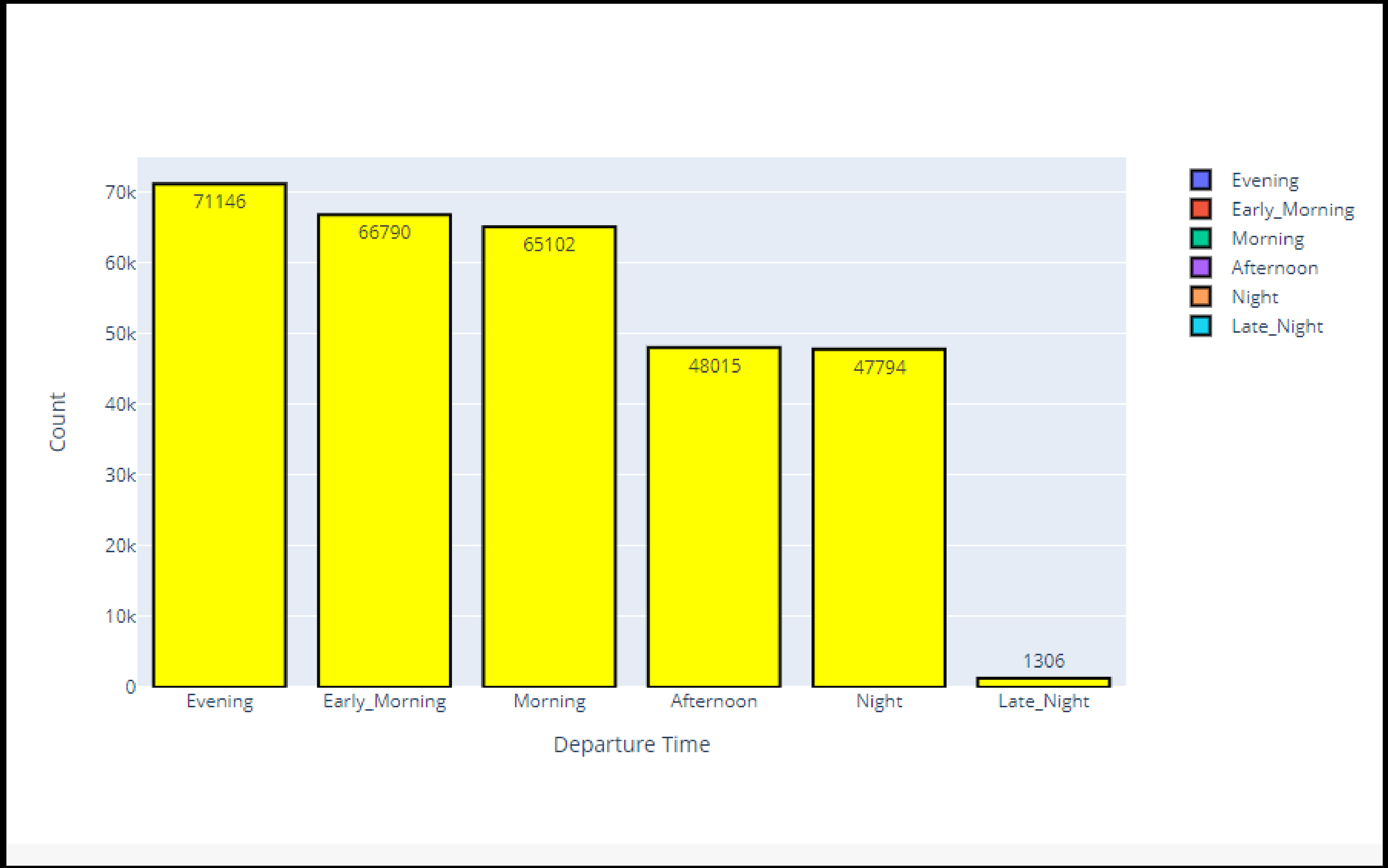


- Mean ticket price for *SpiceJet* airline is: 6179.28
- Mean ticket price for *AirAsia* airline is: 4091.07
- Mean ticket price for *Vistara* airline is: 30396.54
- Mean ticket price for *GO_FIRST* airline is: 5652.01
- Mean ticket price for *Indigo* airline is: 5324.22
- Mean ticket price for *Air_India* airline is: 23507.02

EXPLORATORY DATA ANALYSIS

Departure Time Distribution

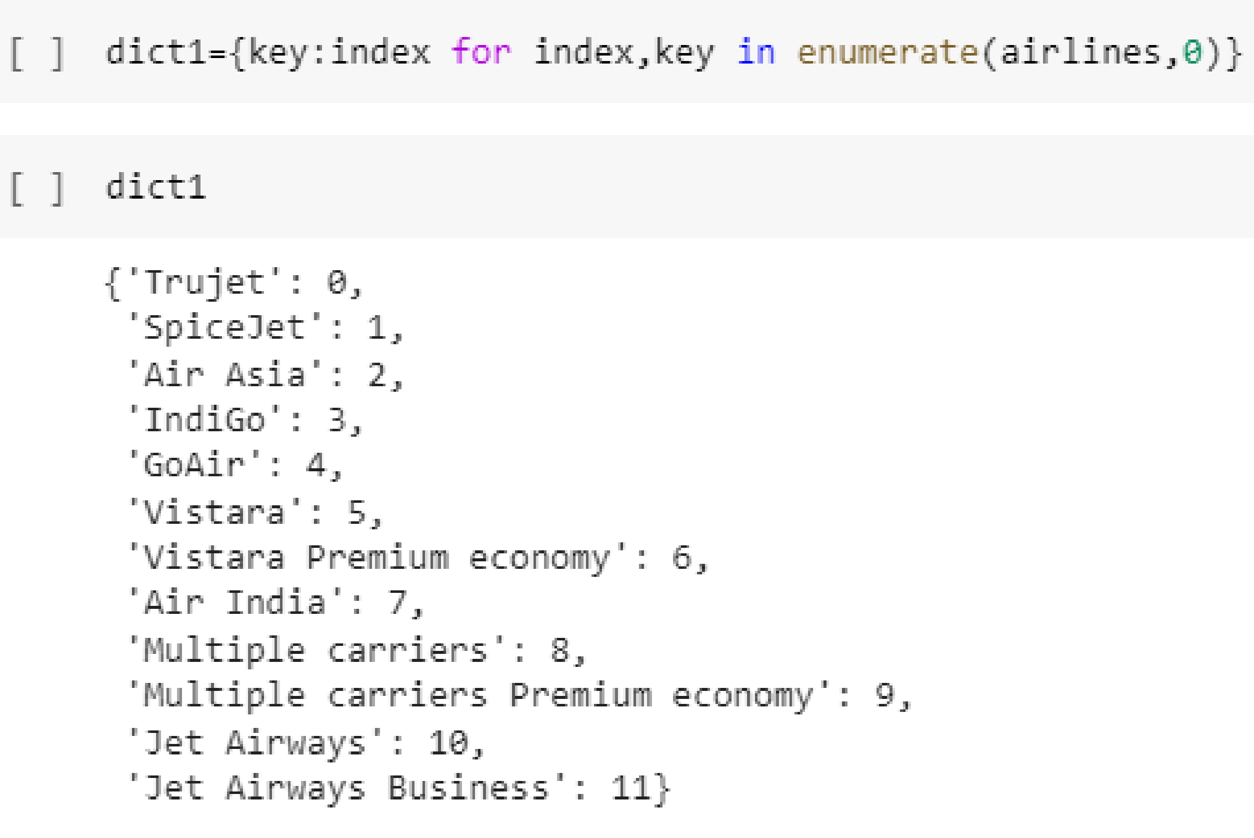




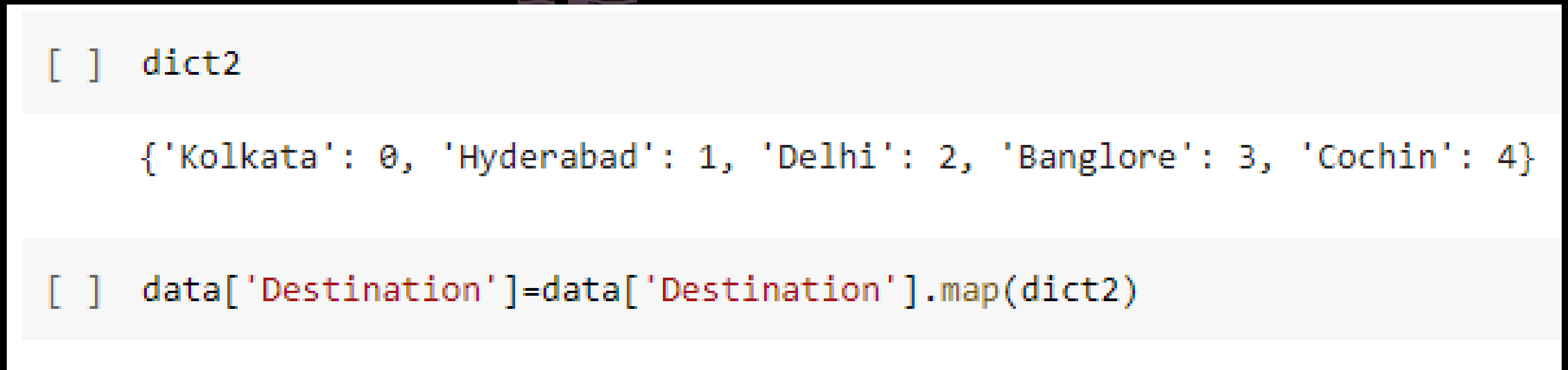
FEATURE ENCODING

Handling Categorical Data

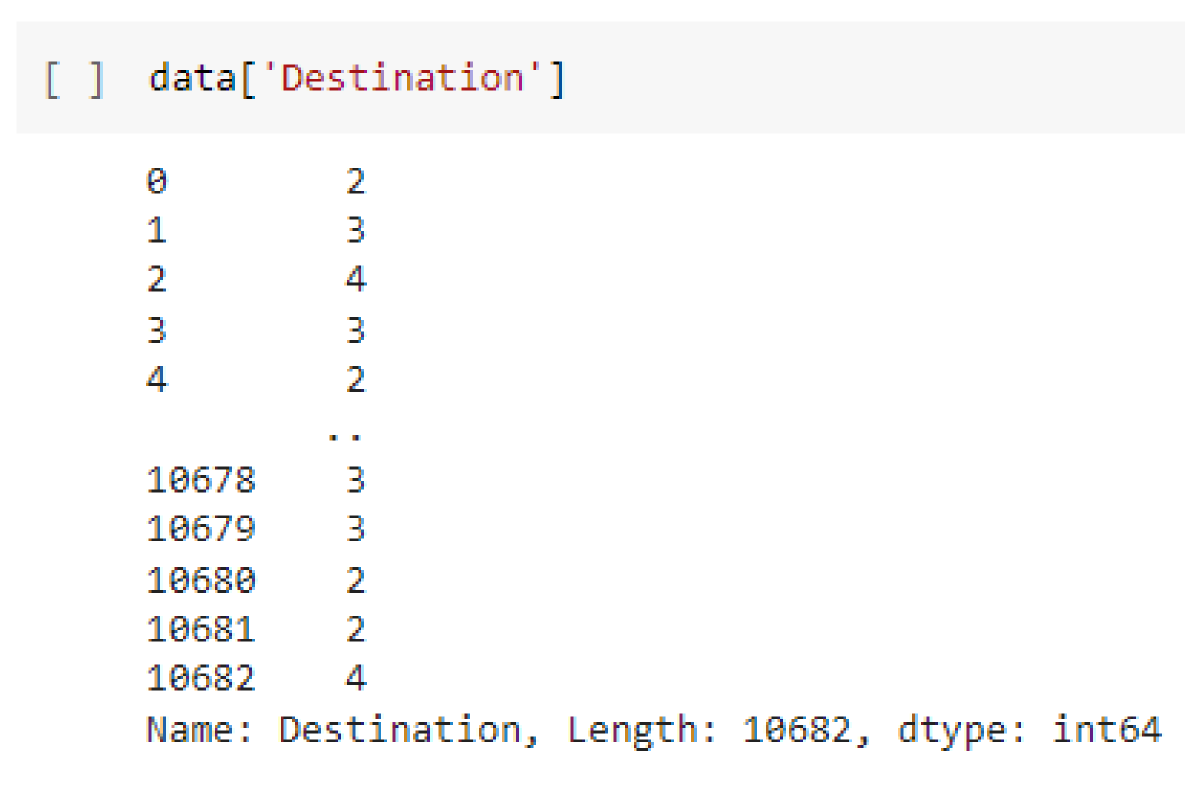
- We are using 2 basic Encoding Techniques to convert Categorical data into some numerical format
- if data belongs to Nominal data (ie data is not in any order) -->> OneHotEncoder is used in this case
- if data belongs to Ordinal data (ie data is in order) -->> LabelEncoder is used in this case



```
[ ]  dict1={key:index for index,key in enumerate(airlines,0)}

[ ]  dict1

     {'Trujet': 0,
      'SpiceJet': 1,
      'Air Asia': 2,
      'IndiGo': 3,
      'GoAir': 4,
      'Vistara': 5,
      'Vistara Premium economy': 6,
      'Air India': 7,
      'Multiple carriers': 8,
      'Multiple carriers Premium economy': 9,
      'Jet Airways': 10,
      'Jet Airways Business': 11}
```



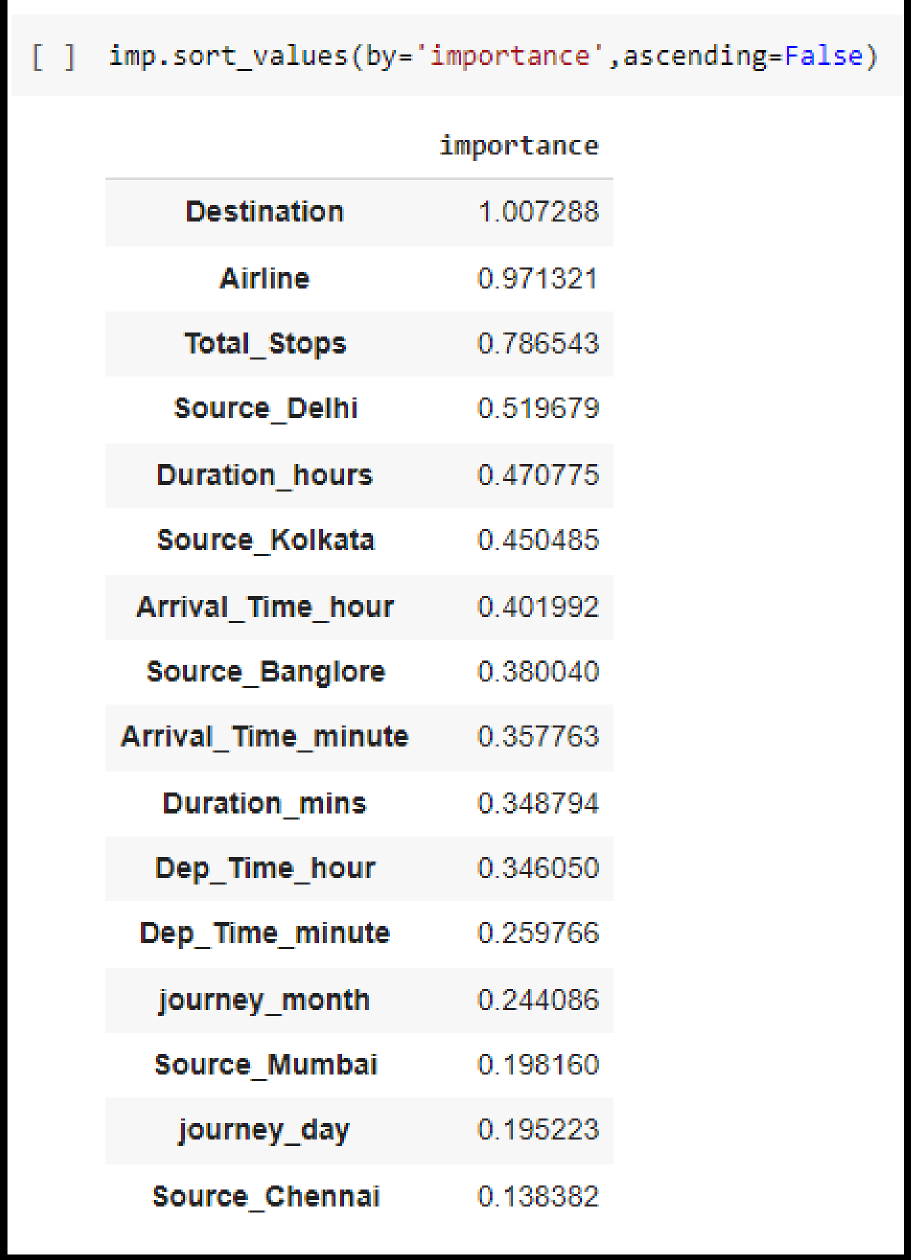
```
[ ]  dict2

     {'Kolkata': 0, 'Hyderabad': 1, 'Delhi': 2, 'Banglore': 3, 'Cochin': 4}

[ ]  data['Destination']=data['Destination'].map(dict2)
```



```
[ ]  data['Destination']

     0        2
     1        3
     2        4
     3        3
     4        2
             ..
     10678    3
     10679    3
     10680    2
     10681    2
     10682    4
     Name: Destination, Length: 10682, dtype: int64
```

FEATURE ENCODING

- Finding out the best feature which will contribute most to the target variable. So to get a high level overview of most of the frequently used feature selection technique.
- Why to apply Feature Selection? To select important features to get rid of curse of dimensionality ie..to get rid of duplicate features
- Ways or technqiues to do it if we have regression use-case a..SelectKBest Score function:



```
[ ]  imp.sort_values(by='importance',ascending=False)
```

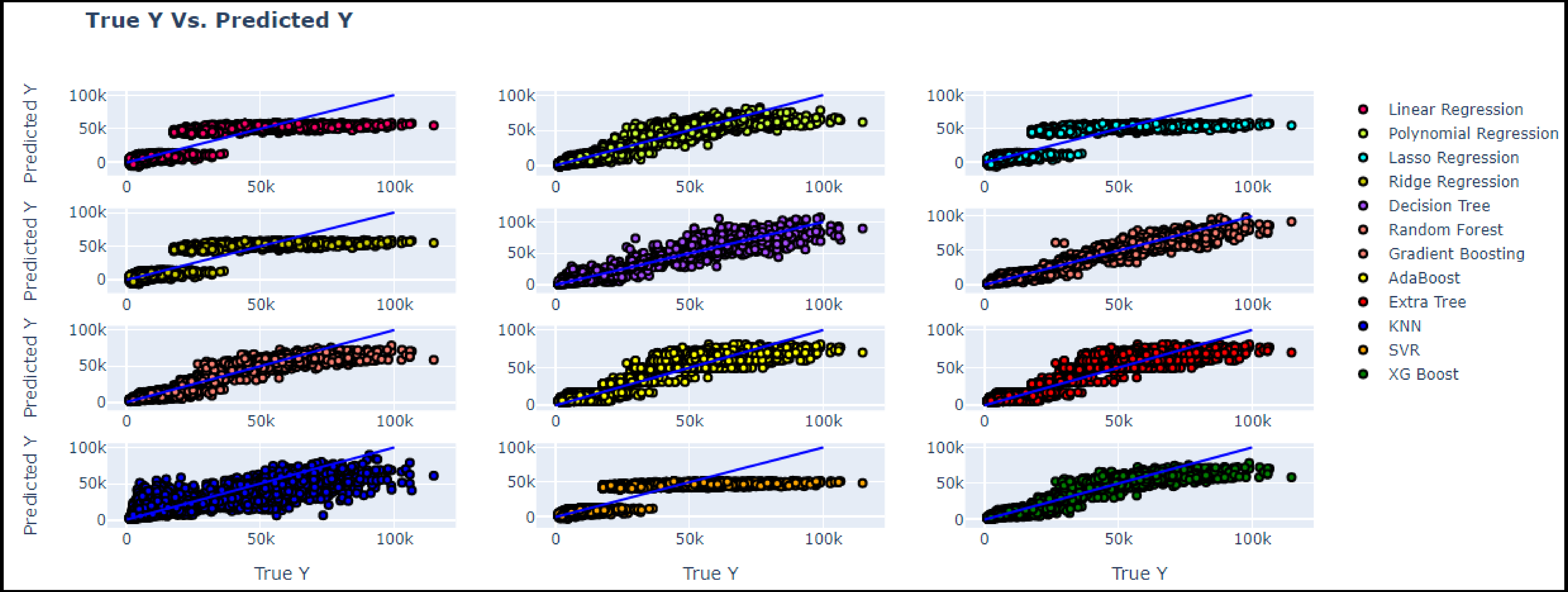
	importance
Destination	1.007288
Airline	0.971321
Total_Stops	0.786543
Source_Delhi	0.519679
Duration_hours	0.470775
Source_Kolkata	0.450485
Arrival_Time_hour	0.401992
Source_Banglore	0.380040
Arrival_Time_minute	0.357763
Duration_mins	0.348794
Dep_Time_hour	0.346050
Dep_Time_minute	0.259766
journey_month	0.244086
Source_Mumbai	0.198160
journey_day	0.195223
Source_Chennai	0.138382

MACHINE LEARNING MODEL USED

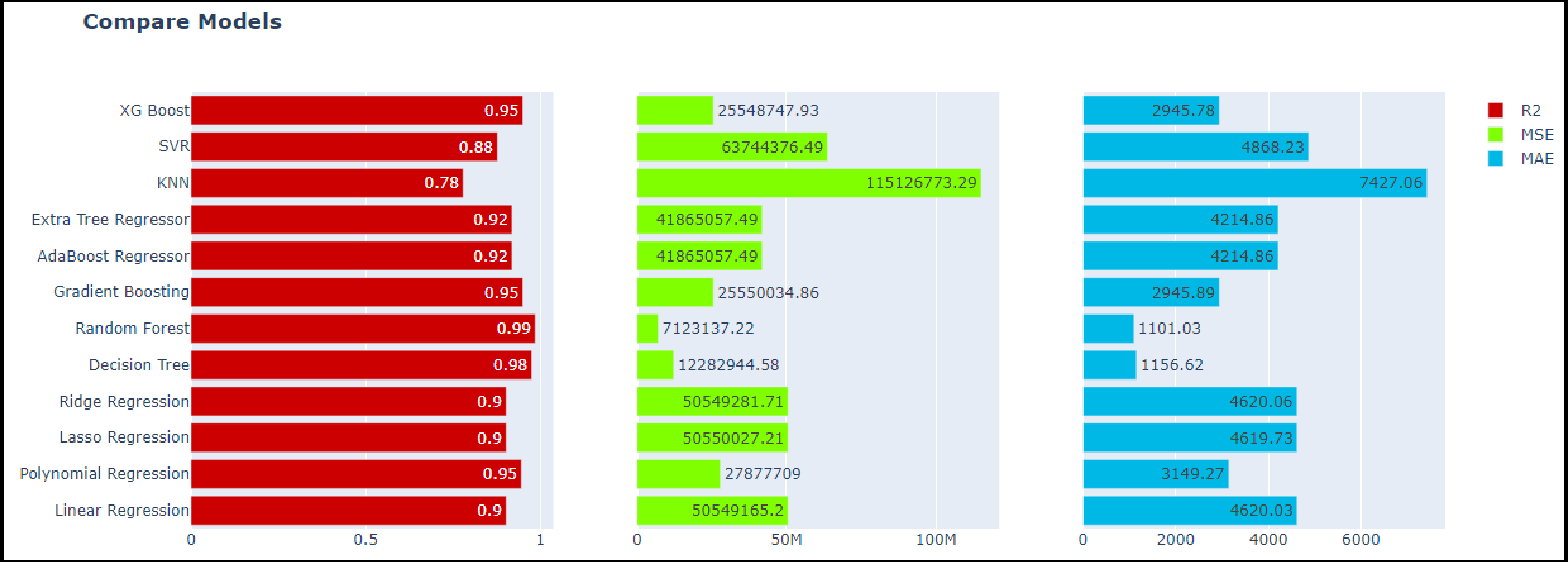
1. Linear regression
2. Polynomial regression
3. Lasso regression
4. Ridge regression
5. Decision tree
6. Gradient boosting
7. Random forest
8. Adaboost regression
9. Extra tree regression
10. KNN
11. SVR
12. XG Boost

MACHINE LEARNING MODEL

Which model is the best for predicting flight price?



MACHINE LEARNING MODEL



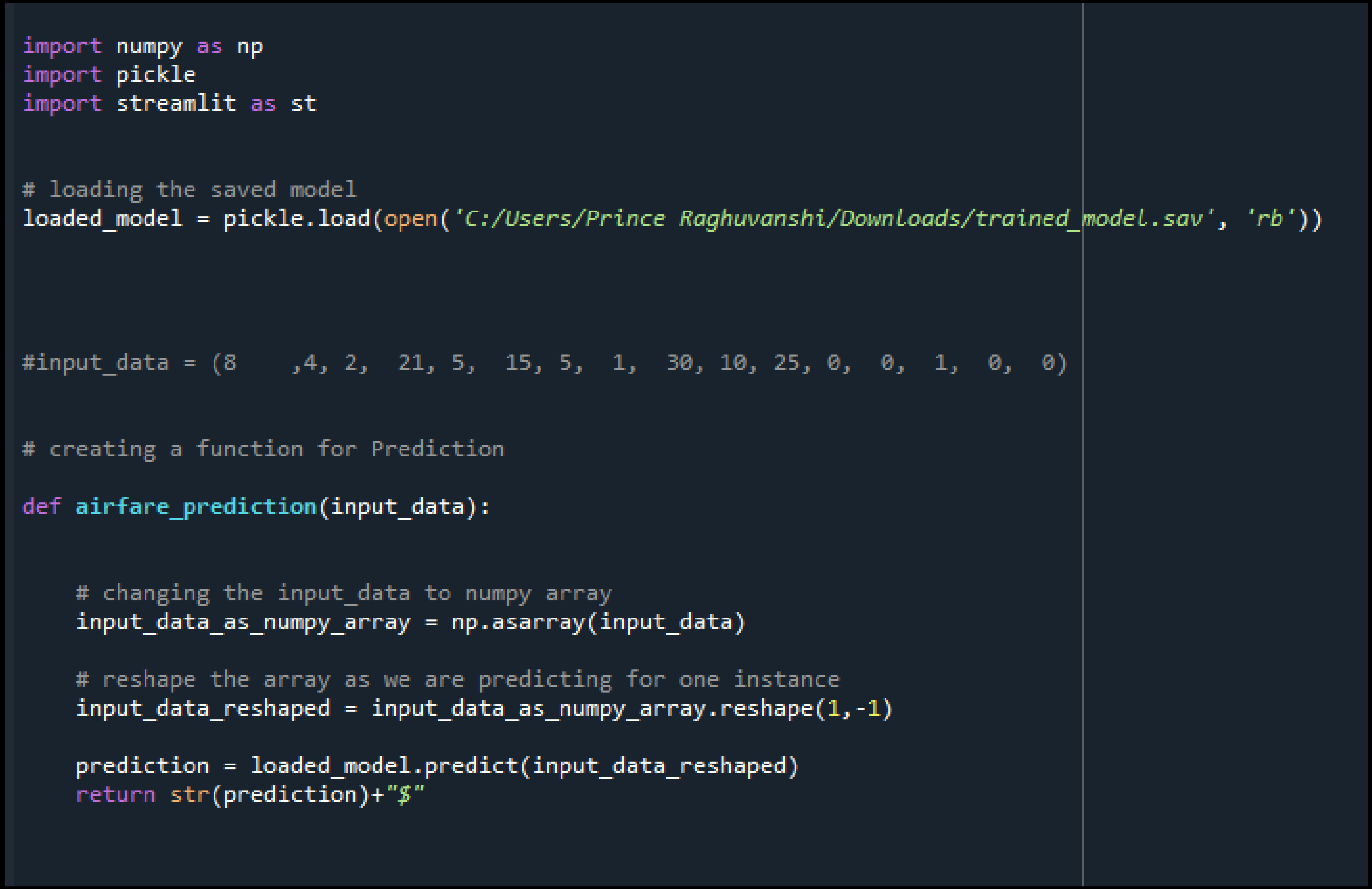
DEPLOYEMENT

Things to look at before deploying the model.

```
{'KOLKATA': 0,
'HYDERABAD': 1,
'DELHI': 2,
'BANGLORE': 3,
'COCHIN': 4}
```

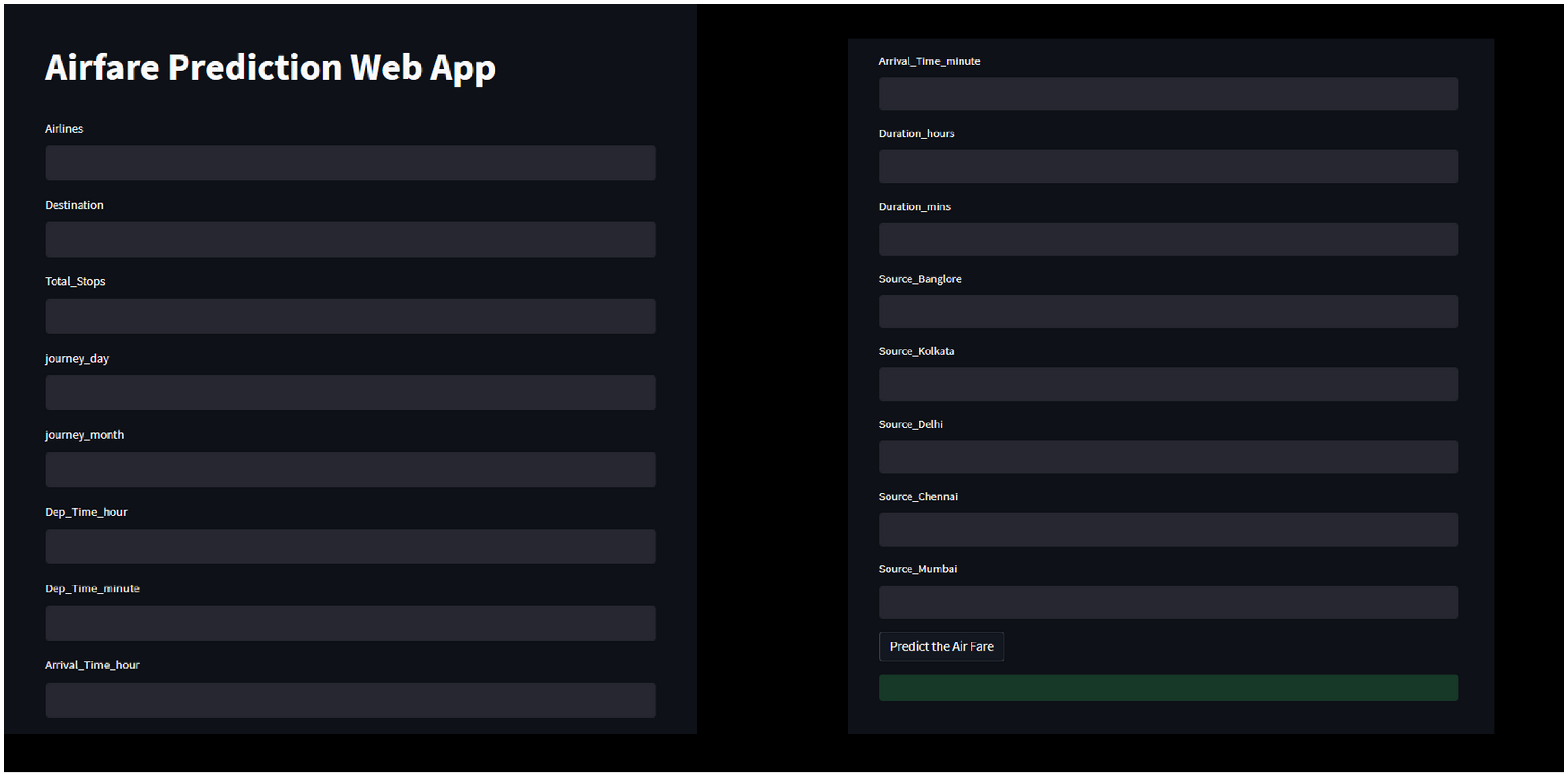
3	4	1	27	5	16	0	21	0	5	0	0	0	1	0	0

DEPLOYEMENT MODEL



```
import numpy as np
import pickle
import streamlit as st


# loading the saved model
loaded_model = pickle.load(open('C:/Users/Prince Raghuvanshi/Downloads/trained_model.sav', 'rb'))




#input_data = (8    ,4, 2,  21, 5,  15, 5,  1,  30, 10, 25, 0,  0,  1,  0,  0)


# creating a function for Prediction

def airfare_prediction(input_data):


    # changing the input_data to numpy array
    input_data_as_numpy_array = np.asarray(input_data)

    # reshape the array as we are predicting for one instance
    input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

    prediction = loaded_model.predict(input_data_reshaped)
    return str(prediction)+"$"
```

DEPLOYEMENT



Airfare Prediction Web App

Airlines

Destination

Total_Stops

journey_day

journey_month

Dep_Time_hour

Dep_Time_minute

Arrival_Time_hour

Arrival_Time_minute

Duration_hours

Duration_mins

Source_Banglore

Source_Kolkata

Source_Delhi

Source_Chennai

Source_Mumbai

REFERENCES

- https://www.altexsoft.com/blog/flight-price-predictor/
- https://medium.com/geekculture/flight-fare-prediction-93da3958eb95
- https://discuss.streamlit.io/t/drop-down-menu/3180/2

THANK YOU