




BIL 401 – INTRODUCTION TO BIG DATA

STATUS REPORT

Sena Ezgi Anadollu 201401017
Beyza Altanlar 151101040



Flight Price Prediction Using Apache Spark and PySpark Framework

1st Sena Ezgi Anadolliu

*Department of Artificial Intelligence Engineering
TOBB University of Economics and Technology
Ankara, Turkey
sena@anadolliu.com*

2nd Beyza Altanlar

*Department of Computer Engineering
TOBB University of Economics and Technology
Ankara, Turkey
beyzaaltanlar@gmail.com*

Abstract—Airlines dynamically adjust ticket prices depending on various factors and use special algorithms to determine the optimal pricing policy.

This paper introduces methods for the analysis of airline flight data, focusing on flight price prediction within the Apache Spark environment. The factors influencing ticket pricing are considered using PySpark. Among the factors focused on are airline type, days left for departure, departure time and arrival time, source and destination city, class type. A novel dataset consisting of 300153 distinct flight booking options collected for 50 days, from February 11th to March 31st, 2022 is utilized.

It is envisaged that this study could be further elevated through the utilization of machine learning algorithms and intended to apply them as future work by using MLlib.

Index Terms—Feature Analysis, PySpark, Apache Spark, Flight Price Prediction, Big Data

I. INTRODUCTION

Apache Spark is an open-source computing framework designed for big data processing and analytics. It provides high-level abstractions and tools for efficiently processing large datasets in a distributed and parallel fashion.

PySpark is the Python API for Apache Spark allowing developers to write Spark applications using Python.

MLlib (Machine Learning Library) is a library in Apache Spark that is specifically built for machine learning tasks and offers a wide range of algorithms and tools for this purpose.

This study examines flight data to obtain insights into the factors influencing ticket pricing and acquire vital information about airline flights in the Apache Spark environment.

This paper's organization is as follows: Section II presents literature review containing studies related to flight price prediction. Our methods assembled for the objective are presented in Section III. Section IV provides the analysis of the data before the discussion part in Section V.

II. RELATED WORK

Due to the development of flight ticket price policies and global interest, there has been a surge in research and projects related to flight ticket price prediction. This section is dedicated to those projects that are similar to our own.

[1]This study is similar to our project in aspects of utilizing Spark and machine learning techniques to forecast flight ticket prices. It involved a substantial dataset of roughly 20 million

records (4.68 gigabytes). The researchers employed four regression machine learning algorithms - Random Forest, Gradient Boost Tree, Decision Tree, and Factorization Machines - in their analysis. To assess performance and generalization, they utilized Cross Validator and Training Validator functions. Based on the findings, the Gradient Boost Tree emerged as the most effective algorithm with the highest accuracy.

[2]In a study on predicting airline fares using machine learning, a dataset of 1814 Aegean Airlines flight data was utilized. The study employed various methods such as Multilayer Perceptron, Generalized Regression Neural Network, Extreme Learning Machine, Random Forest Regression Tree, Regression Tree, Bagging Regression Tree, Regression SVM, and Linear Regression, resulting in different outcomes. The Bagging Regression Tree method was found to yield the best results.

[3]In the analysis of flights between Delhi and Mumbai, K-nearest neighbors, linear regression, support vector machine, Multilayer Perceptron, Gradient Boosting, and Random Forest Algorithms were applied to a dataset. The Decision Tree model produced the most favorable results.

[4]The study investigated six targets in four different ways, utilizing eight machine learning models, six deep learning models, and two quantum machine learning models. The results showed that at least three models could achieve an accuracy percentage of 89% to 99%, with the highest success rate achieved through QML methods. The study also found that QML methods can provide effective solutions for airfare estimation. Comparing ML and DL, QML methods were found to yield the best results.

[3]Various machine learning algorithms have been assessed for their effectiveness in predicting flight prices. These include Linear Regression, Decision Tree, Random Forest, K-nearest neighbors, Multilayer Perceptron, Support Vector Machine (SVM) and Gradient Boosting. Among these, Random Forest and Multilayer Perceptron are the most successful. In an effort to enhance prediction accuracy, a "stacked" prediction model has been introduced, which combines the outcomes of Random Forest and Multilayer Perceptron by assigning weights. Although the "stacked" prediction model displays encouraging results, it implies that future research can further optimize the model by considering additional features such as

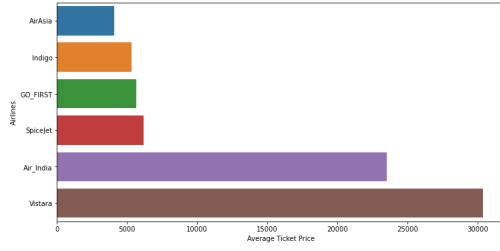


Fig. 1. Average Ticket Price by Airline

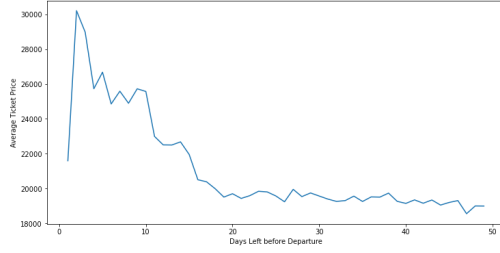


Fig. 2. Average Ticket Price by Days Left

the number of seats.

Although we found that the aforementioned projects employed various programming languages and datasets, they are similar to ours with regard to their machine-learning approaches. Our approach is unique in that it is designed to run on the Apache Spark environment.

III. PROPOSED IMPLEMENTATION

The data used in this study is obtained from the "Ease My Trip" website from Kaggle. It includes 300153 different flight reservation options. There is a cleaned dataset containing 11 features. These features include airline name, flight code, source city, departure time, number of stops, arrival time, destination city, class, duration, remaining time to departure, and ticket price.

We inspected the data on some aspects in order to gain insight into the data. Those aspects are price changes according to airline companies, the effect of last-minute ticket purchases on prices, varying based on departure and arrival times, price fluctuations based on source and destination cities, and the differences between Economy and Business class ticket prices. We plan to perform more advanced analysis with the help of the ML algorithms.

IV. RESULTS

The bar chart in Fig. 1, depicts the average ticket prices for different airlines. There is variation in ticket prices among the different airlines and Airline Vista stands out with the highest average ticket price.

The line plot in Fig. 2 illustrates how average ticket prices vary based on the number of days left before departure. Prices generally appear to increase as the departure date approaches, emphasizing the advantage of early booking. However, there

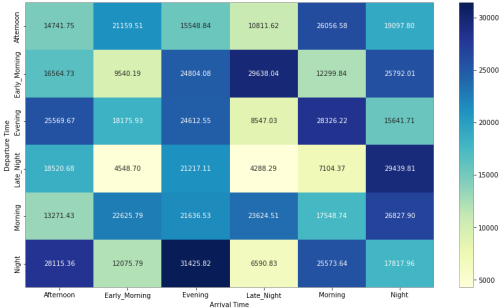


Fig. 3. Average Ticket Price by Departure and Arrival Time

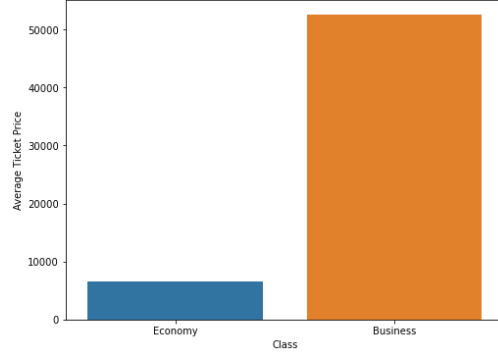


Fig. 4. Average Ticket Price by Class

is a slight dip in prices for tickets booked a day or two before departure.

The heatmap in Fig. 3 showcases the average ticket prices based on both departure and arrival times. Flights departing in the early morning or late evening, especially with midday arrivals, tend to have lower prices. In contrast, peak travel times during the day exhibit higher average prices.

This visualization in Fig. 4 provides insights into the cost differences between the two classes. Business class tickets are considerably more expensive than Economy class tickets.

The average ticket prices for various source and destination city pairs are shown in Fig. 5. Notably, routes involving Delhi consistently have lower prices, whereas those involving Chennai have higher prices than other cities.

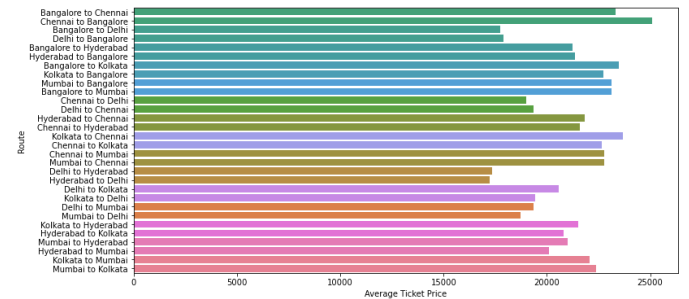


Fig. 5. Average Ticket Price by Source and Destination

V. DISCUSSION

Through the application of machine learning algorithms, this study is anticipated to be further refined. Further works can be discussed with the trial of different ML algorithms.

REFERENCES

- [1] Philip Wong, Phue Thant, Pratiksha Yadav, Ruta Antaliya, and Jongwook Woo. Using spark machine learning models to perform predictive analysis on flight ticket pricing data. *arXiv preprint arXiv:2310.07787*, 2023.
- [2] Konstantinos Tziridis, Th Kalampokas, George A Papakostas, and Kostas I Diamantaras. Airfare prices prediction using machine learning techniques. In *2017 25th European Signal Processing Conference (EUSIPCO)*, pages 1036–1039. IEEE, 2017.
- [3] Supriya Rajankar and Neha Sakharkar. A survey on flight pricing prediction using machine learning. *International Journal Of Engineering Research & Technology (Ijert)*, 8(6):1281–1284, 2019.
- [4] Theofanis Kalampokas, Konstantinos Tziridis, Nikolaos Kalampokas, Alexandros Nikolaou, Eleni Vrochidou, and George A Papakostas. A holistic approach on airfare price prediction using machine learning techniques. *IEEE Access*, 2023.

```
import pyspark as spark
from pyspark.sql import SparkSession

spark=SparkSession.builder.appName('FlightPricePrediction').getOrCreate()
```

Setting default log level to "WARN".

To adjust logging level use `sc.setLogLevel(newLevel)`. For SparkR, use `setLogLevel(newLevel)`.

23/11/13 07:03:14 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

```
df_pyspark=spark.read.csv('Clean_Dataset.csv',header=True,inferSchema=True)
df_pyspark.show()
```

```
+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+
|index|  airline| flight|source_city|departure_time|stops|
arrival_time|destination_city|  class|duration|days_left|price|
+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+
|    0| SpiceJet|SG-8709|    Delhi|    Evening| zero|
Night|    Mumbai|Economy|    2.17|    1| 5953|
|    1| SpiceJet|SG-8157|    Delhi| Early_Morning| zero|
Morning|    Mumbai|Economy|    2.33|    1| 5953|
|    2| AirAsia| I5-764|    Delhi| Early_Morning| zero|
Early_Morning|    Mumbai|Economy|    2.17|    1| 5956|
|    3| Vistara| UK-995|    Delhi|    Morning| zero|
Afternoon|    Mumbai|Economy|    2.25|    1| 5955|
|    4| Vistara| UK-963|    Delhi|    Morning| zero|
Morning|    Mumbai|Economy|    2.33|    1| 5955|
|    5| Vistara| UK-945|    Delhi|    Morning| zero|
Afternoon|    Mumbai|Economy|    2.33|    1| 5955|
|    6| Vistara| UK-927|    Delhi|    Morning| zero|
Morning|    Mumbai|Economy|    2.08|    1| 6060|
|    7| Vistara| UK-951|    Delhi|   Afternoon| zero|
Evening|    Mumbai|Economy|    2.17|    1| 6060|
|    8| GO_FIRST| G8-334|    Delhi| Early_Morning| zero|
Morning|    Mumbai|Economy|    2.17|    1| 5954|
|    9| GO_FIRST| G8-336|    Delhi|   Afternoon| zero|
Evening|    Mumbai|Economy|    2.25|    1| 5954|
|   10| GO_FIRST| G8-392|    Delhi|   Afternoon| zero|
Evening|    Mumbai|Economy|    2.25|    1| 5954|
|   11| GO_FIRST| G8-338|    Delhi|    Morning| zero|
Afternoon|    Mumbai|Economy|    2.33|    1| 5954|
|   12|  Indigo|6E-5001|    Delhi| Early_Morning| zero|
```

```

Morning|          Mumbai|Economy|    2.17|          1| 5955|
| 13|    Indigo|6E-6202|    Delhi|    Morning| zero|
Afternoon|          Mumbai|Economy|    2.17|          1| 5955|
| 14|    Indigo| 6E-549|    Delhi|    Afternoon| zero|
Evening|          Mumbai|Economy|    2.25|          1| 5955|
| 15|    Indigo|6E-6278|    Delhi|    Morning| zero|
Morning|          Mumbai|Economy|    2.33|          1| 5955|
| 16|Air_India| AI-887|    Delhi| Early_Morning| zero|
Morning|          Mumbai|Economy|    2.08|          1| 5955|
| 17|Air_India| AI-665|    Delhi| Early_Morning| zero|
Morning|          Mumbai|Economy|    2.17|          1| 5955|
| 18|  AirAsia| I5-747|    Delhi|    Evening| one|
Early_Morning|          Mumbai|Economy|    12.25|          1| 5949|
| 19|  AirAsia| I5-747|    Delhi|    Evening| one|
Morning|          Mumbai|Economy|    16.33|          1| 5949|
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
only showing top 20 rows

```

```

continuous_features = ['duration', 'days_Left', 'price']
categorical_features = ['airline', 'flight', 'source_city',
'departure_time', 'stops', 'arrival_time', 'destination_city',
'class']

```

```
df_pyspark.select(continuous_features).describe().toPandas()
```

	summary	duration	days_Left	price
0	count	300153	300153	300153
1	mean	12.221020812719066	26.004750910369044	20889.660523133203
2	stddev	7.191997238119004	13.56100368709358	22697.767366074422
3	min	0.83	1	1105
4	max	49.83	49	123071

```
df_pyspark.printSchema()
```

```

root
|-- index: integer (nullable = true)
|-- airline: string (nullable = true)
|-- flight: string (nullable = true)
|-- source_city: string (nullable = true)
|-- departure_time: string (nullable = true)
|-- stops: string (nullable = true)
|-- arrival_time: string (nullable = true)
|-- destination_city: string (nullable = true)
|-- class: string (nullable = true)
|-- duration: double (nullable = true)
|-- days_left: integer (nullable = true)
|-- price: integer (nullable = true)

```

```
df_pyspark.count()
```

```
300153
```

The dataset includes 300153 flight reservation options

```
df_pyspark.distinct().count()
```

```
300153
```

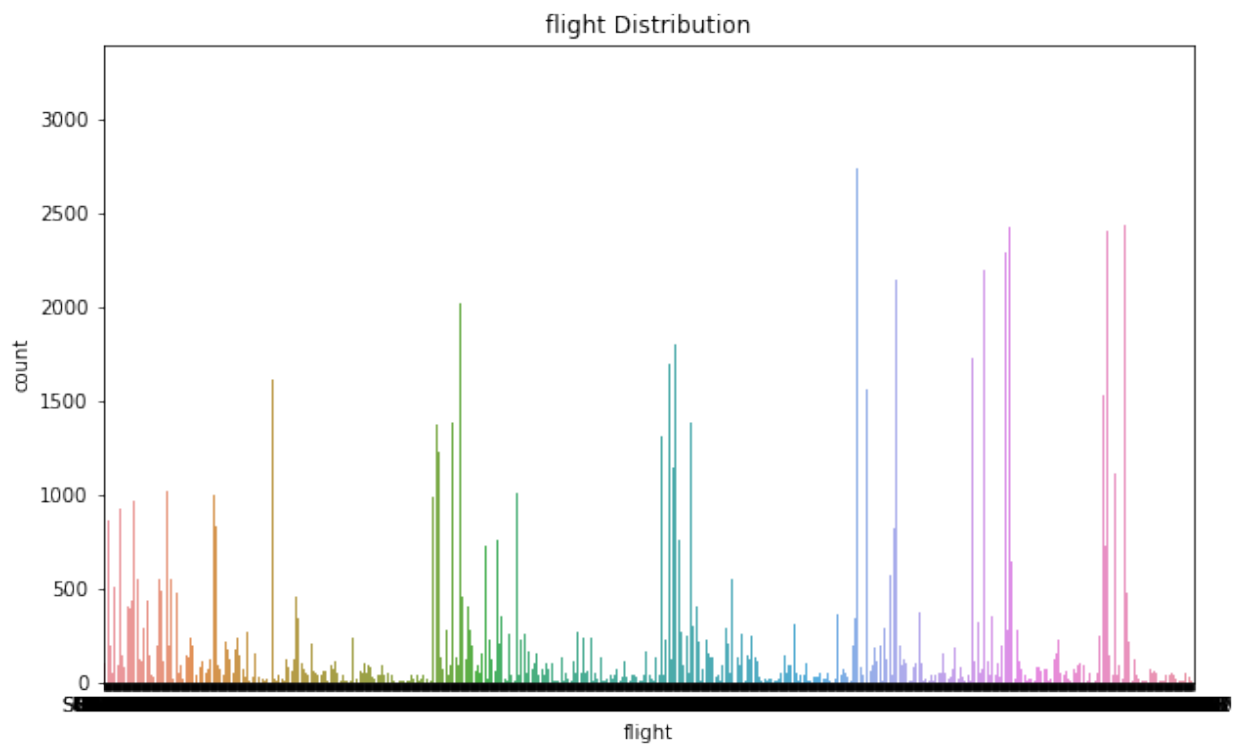
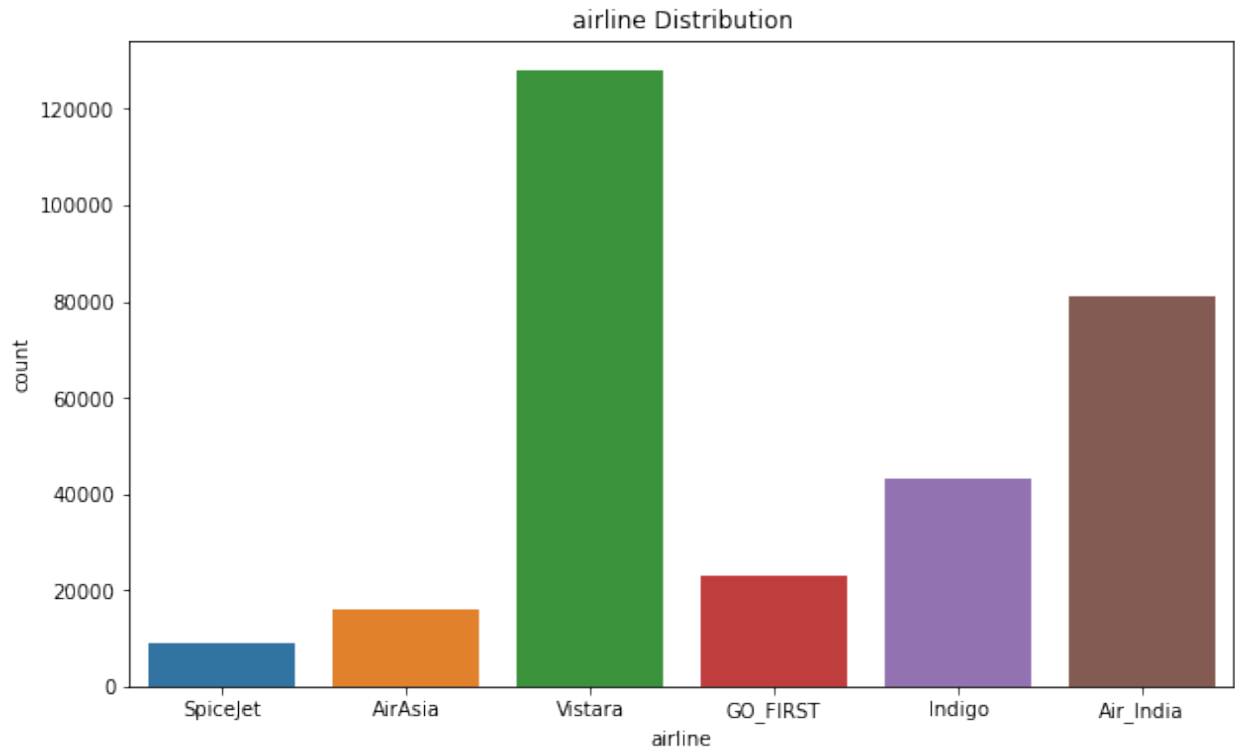
All values are distinct

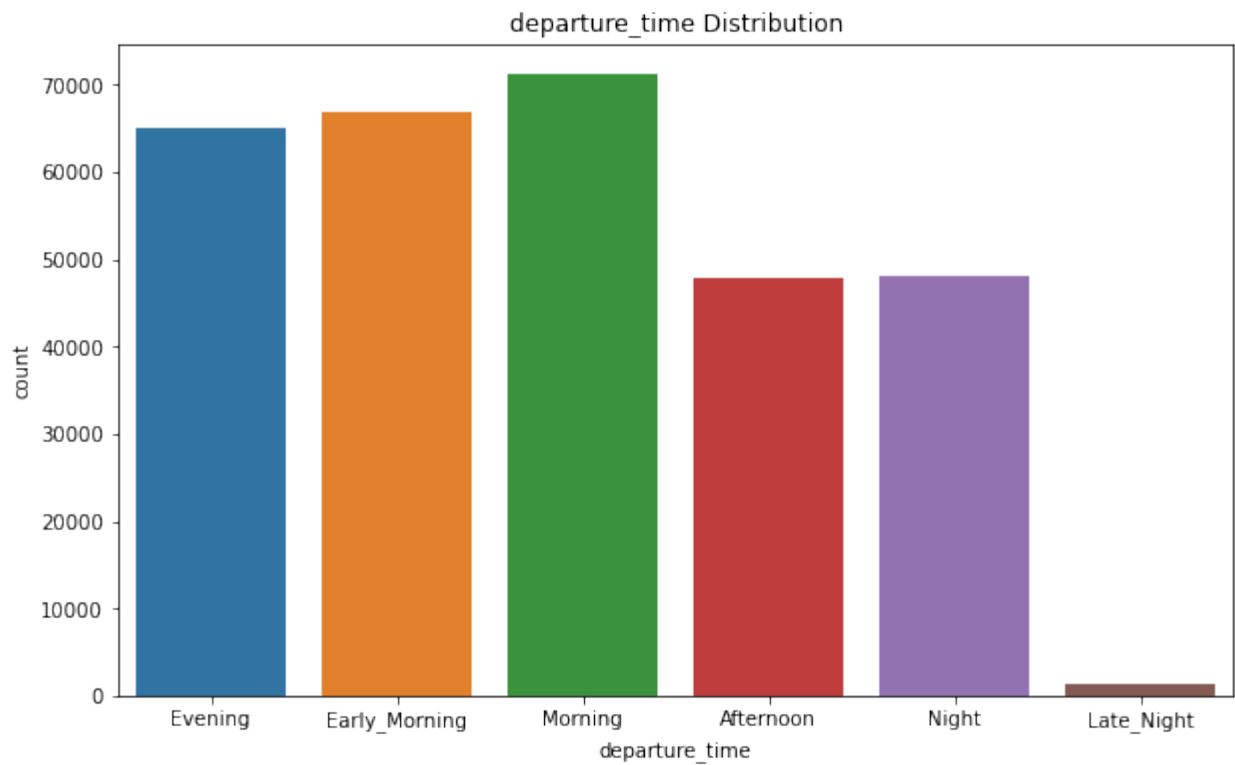
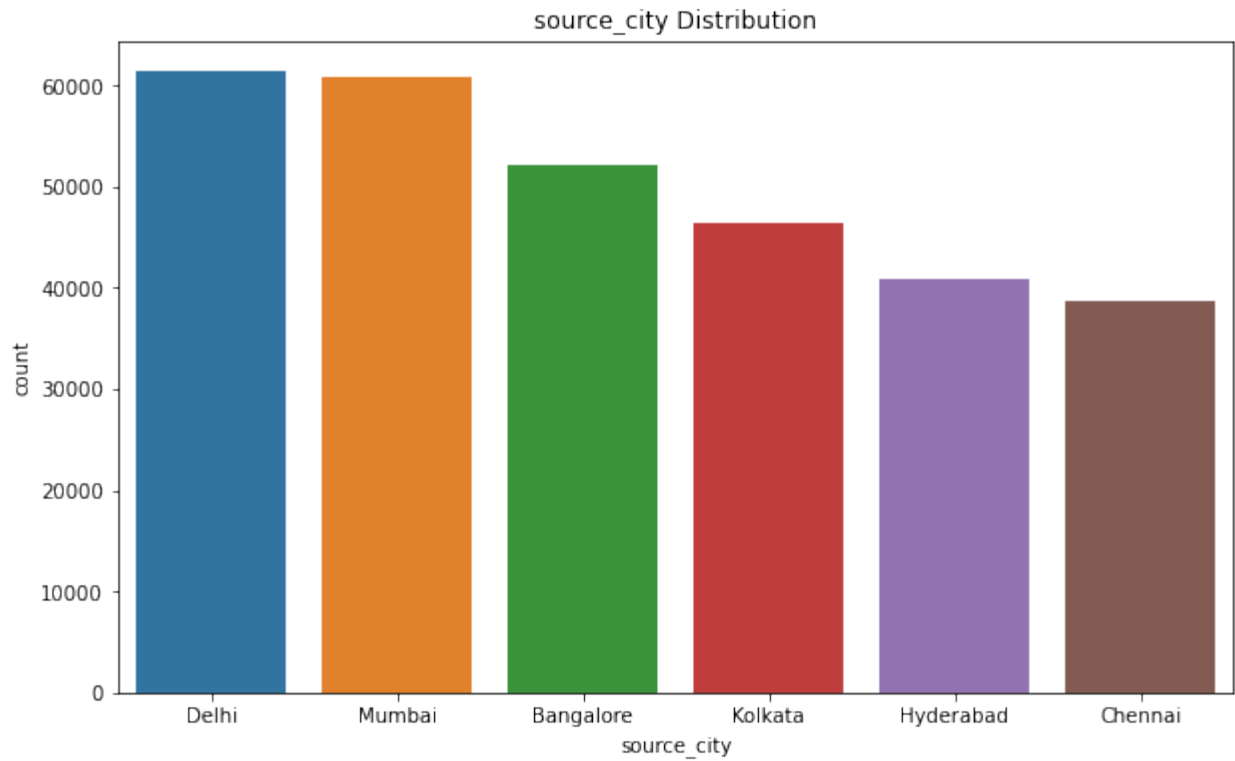
```
df_pyspark_dropped = df_pyspark.na.drop()  
df_pyspark_dropped.count()
```

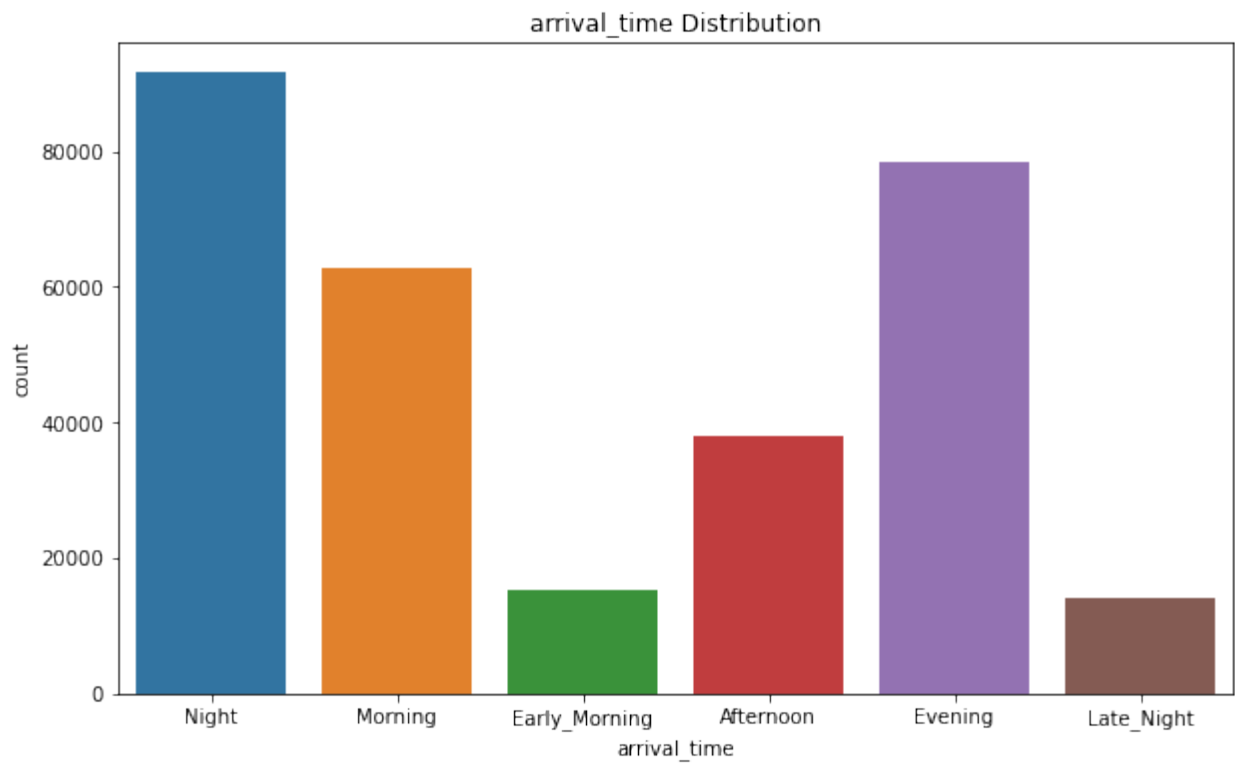
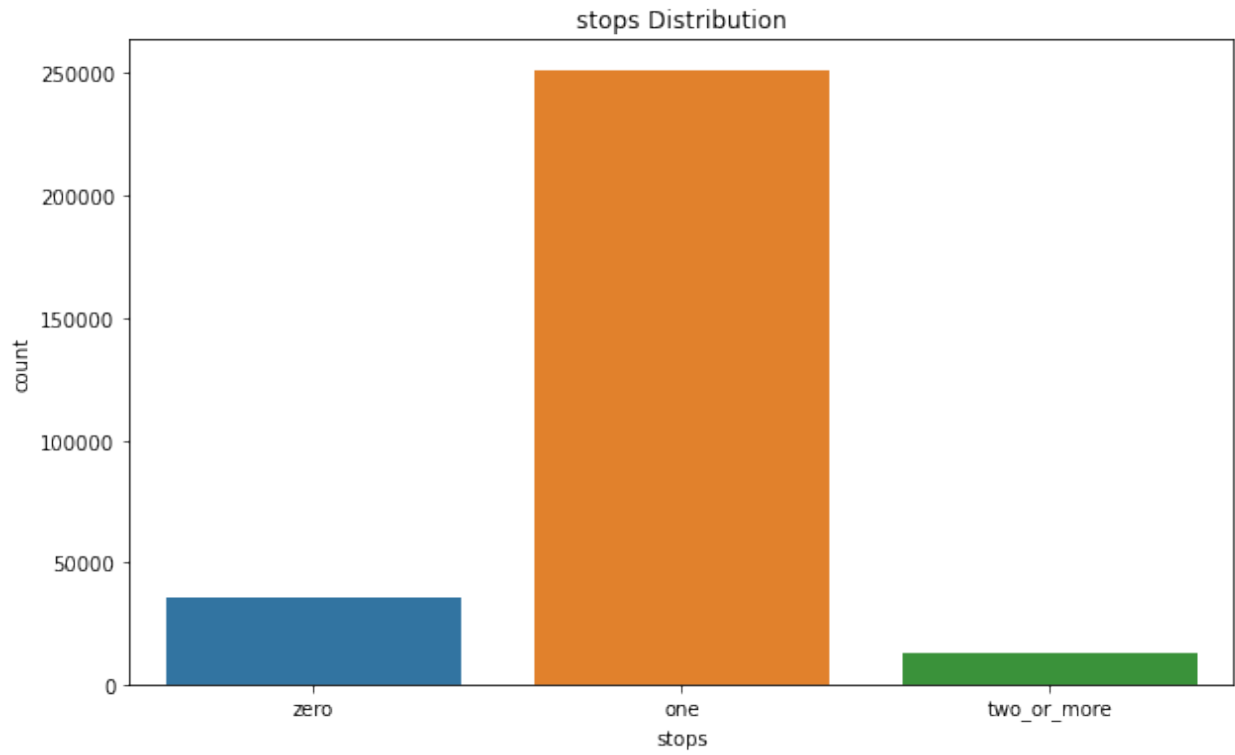
```
300153
```

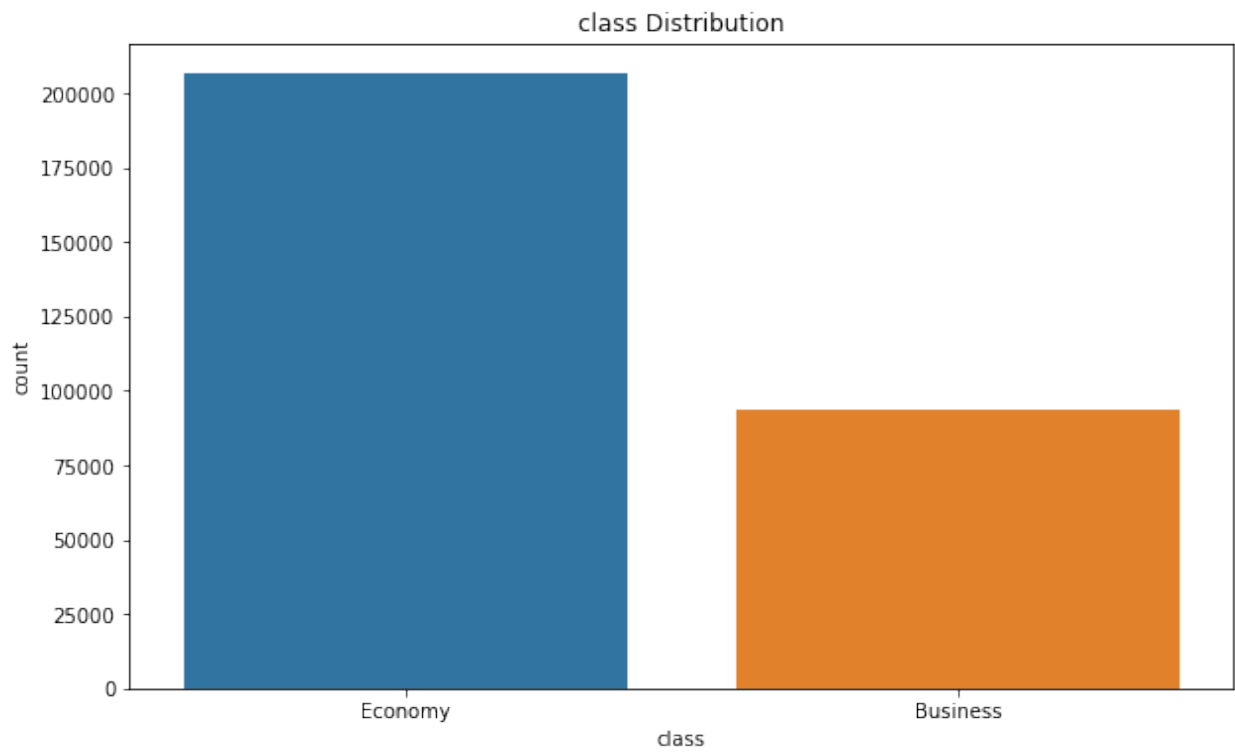
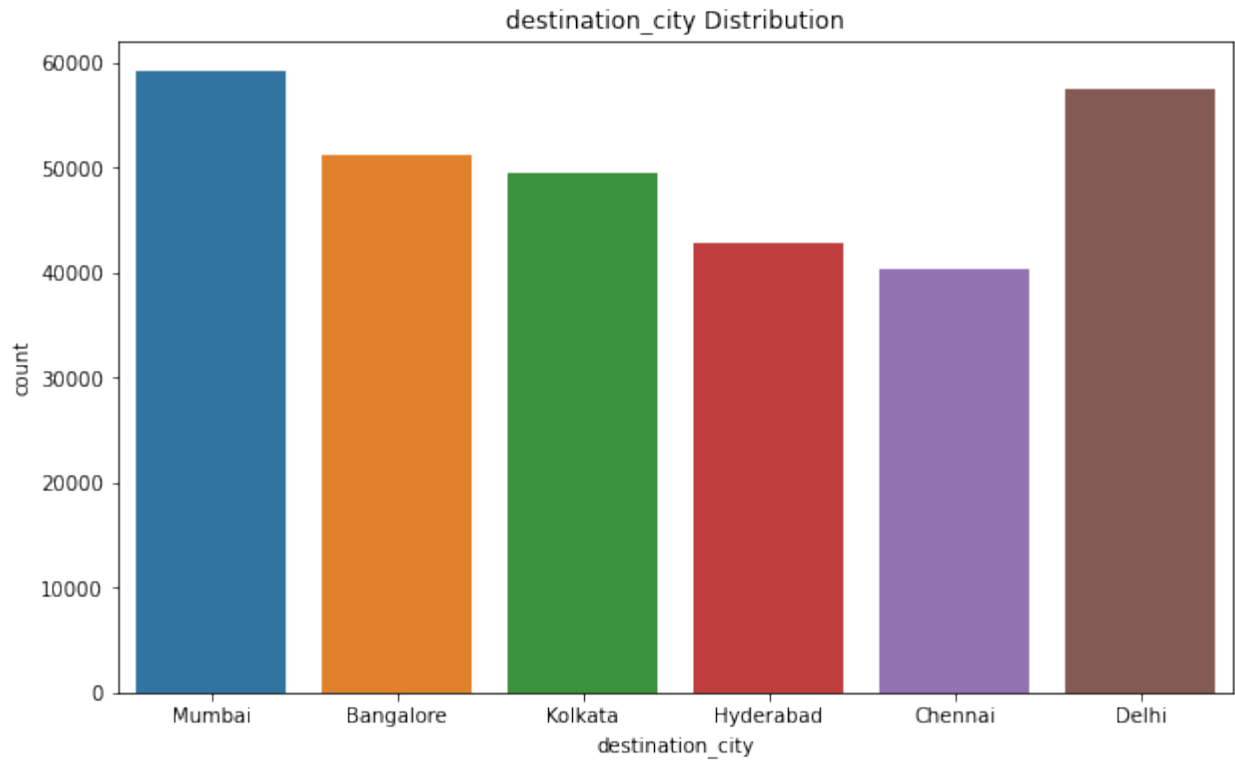
There is not any null values

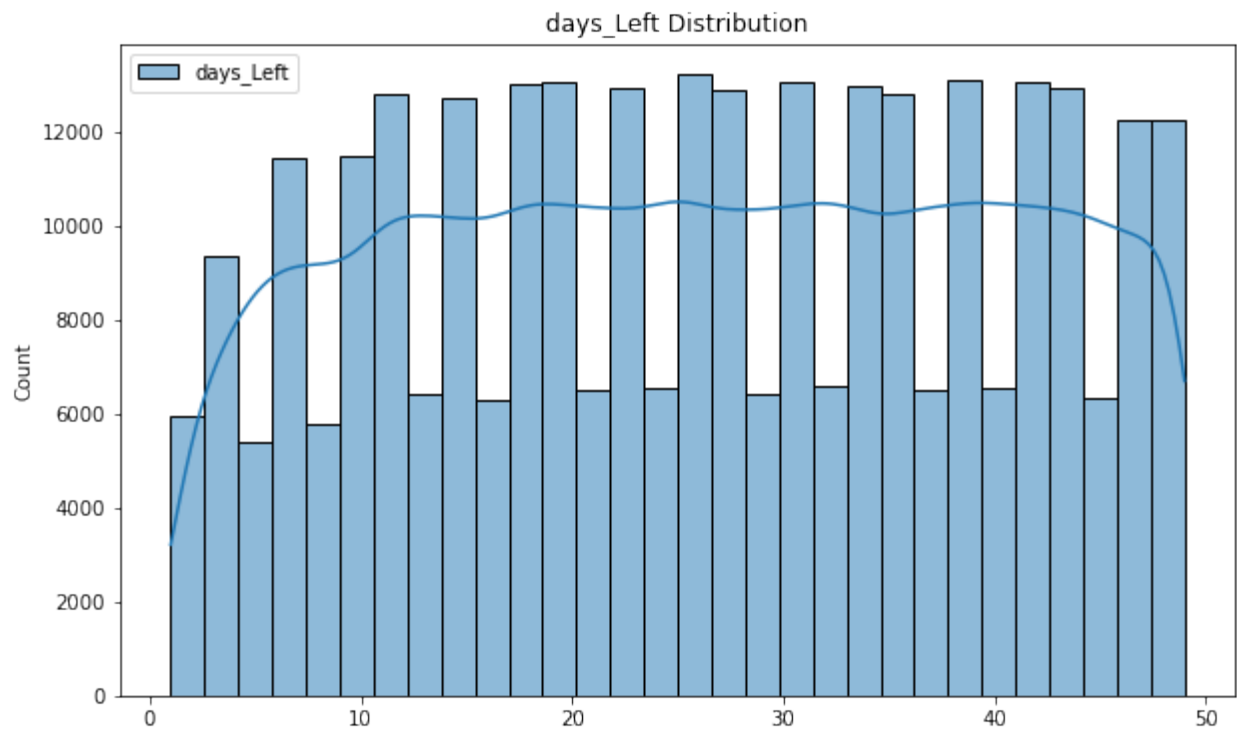
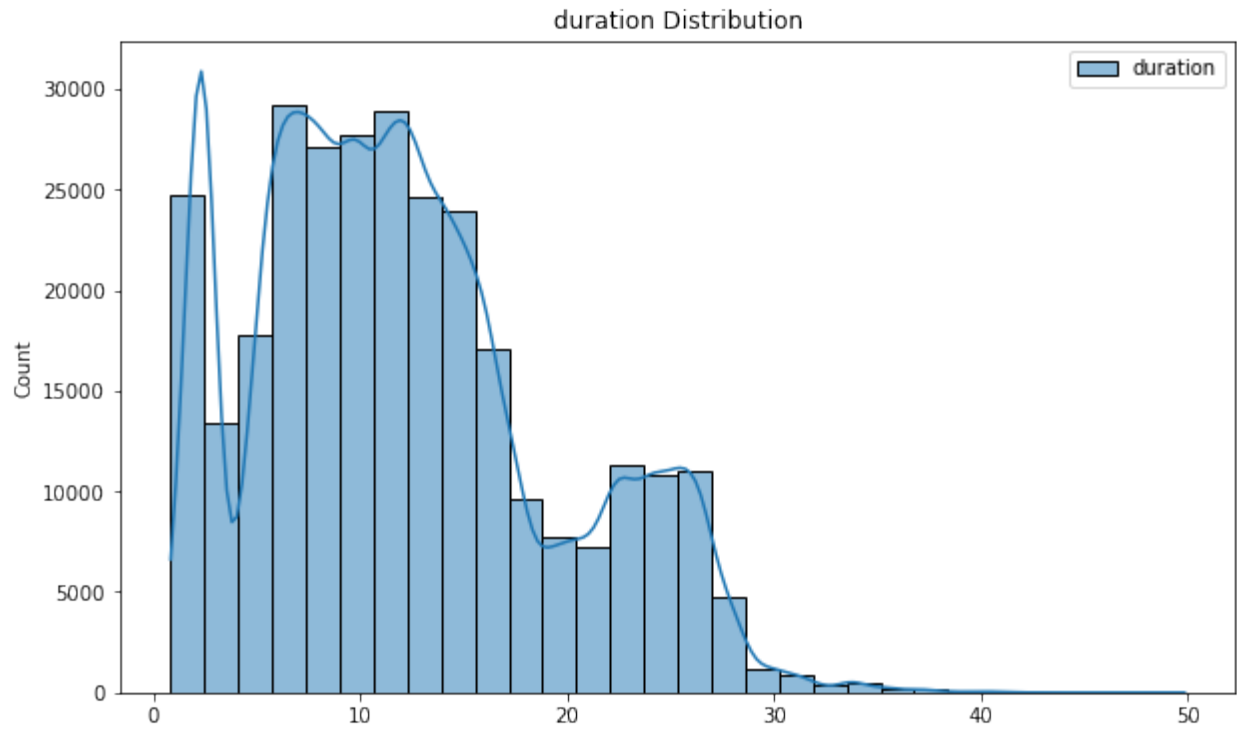
```
import matplotlib.pyplot as plt  
import seaborn as sns  
from pyspark.sql.functions import col  
  
# Distribution for categorical variables  
for feature in categorical_features:  
    plt.figure(figsize=(10, 6))  
    sns.countplot(x=feature, data=df_pyspark.toPandas())  
    plt.title(f"{feature} Distribution")  
    plt.show()  
  
# Distribution for continuous variables  
for feature in continuous_features:  
    plt.figure(figsize=(10, 6))  
    sns.histplot(df_pyspark.select(col(feature)).toPandas(), bins=30,  
kde=True)  
    plt.title(f"{feature} Distribution")  
    plt.show()
```

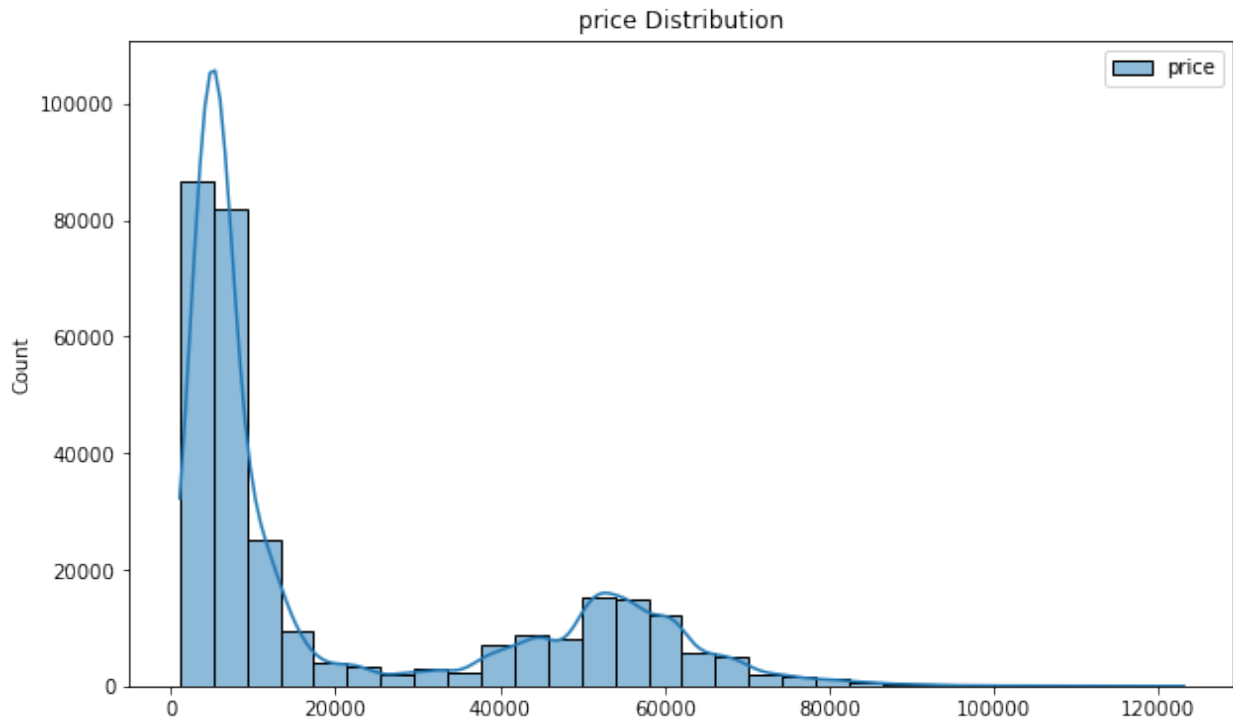








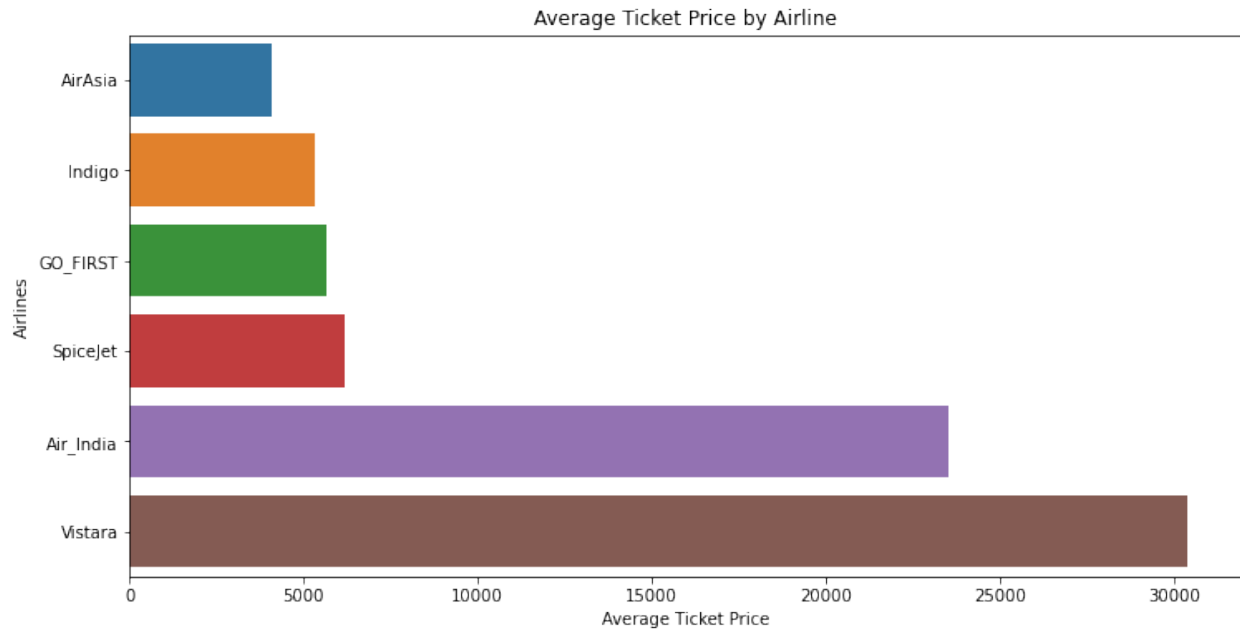




```
# Group by Airline and calculate average price
avg_price_by_airline = df_pyspark.groupBy("airline").agg({"price":
"avg"}).sort("avg(price)")
```

```
# Visualization
```

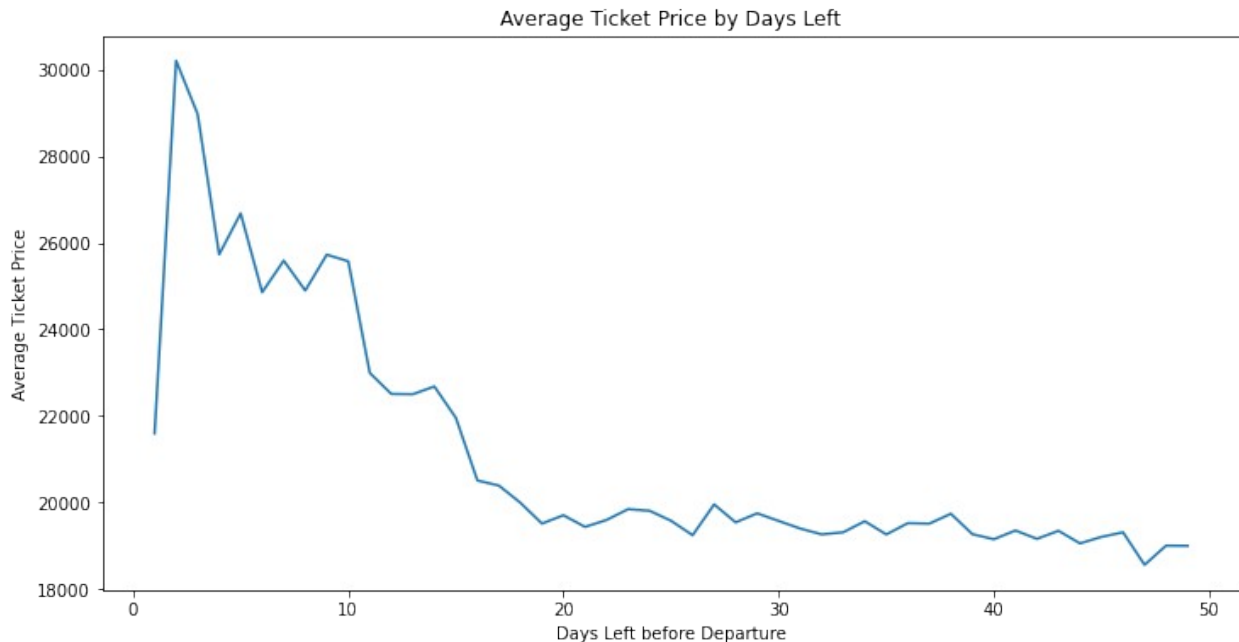
```
plt.figure(figsize=(12, 6))
sns.barplot(x="avg(price)", y="airline",
data=avg_price_by_airline.toPandas())
plt.title("Average Ticket Price by Airline")
plt.xlabel("Average Ticket Price")
plt.ylabel("Airlines")
plt.show()
```



Airline Vista has the highest average ticket price

```
# Group by Days Left and calculate average price
avg_price_by_days_left = df_pyspark.groupBy("days_left").agg({"price":
"avg"}).sort("days_left")

# Visualization
plt.figure(figsize=(12, 6))
sns.lineplot(x="days_left", y="avg(price)",
data=avg_price_by_days_left.toPandas())
plt.title("Average Ticket Price by Days Left")
plt.xlabel("Days Left before Departure")
plt.ylabel("Average Ticket Price")
plt.show()
```

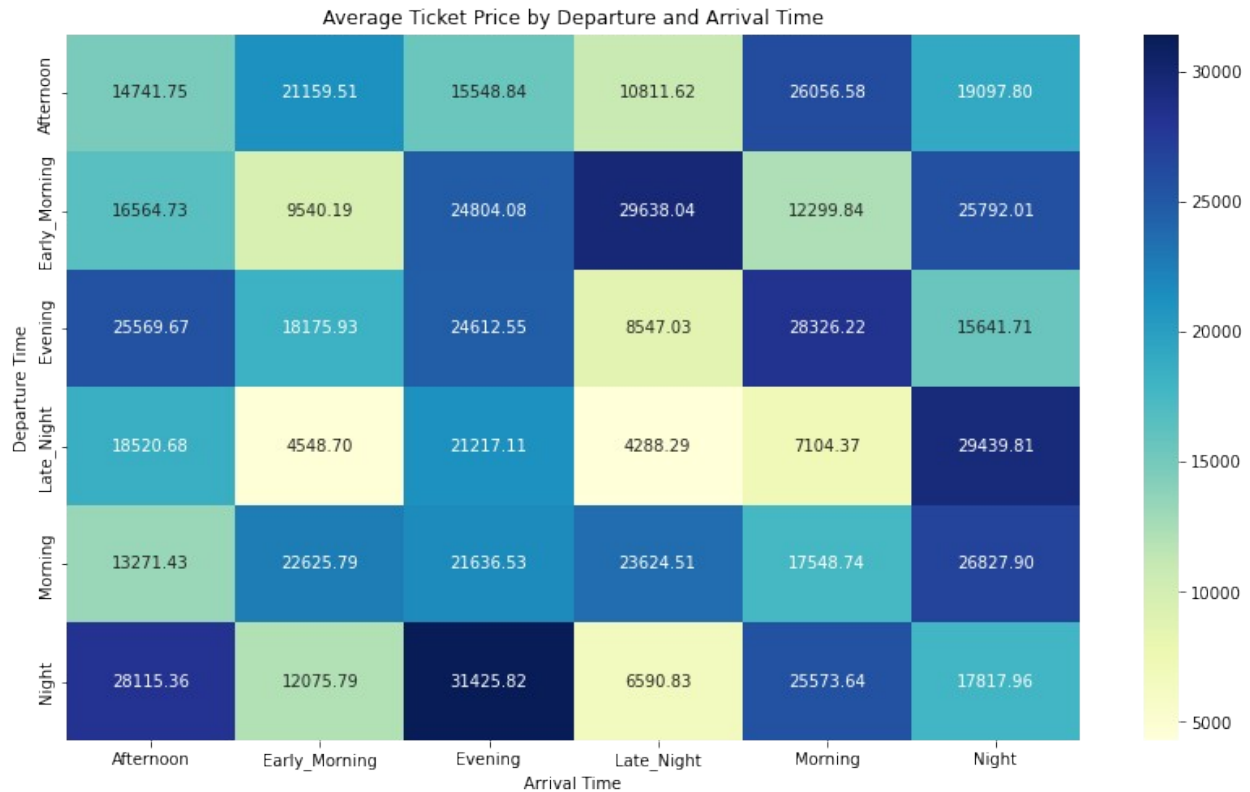


Prices generally increase as the departure date approaches. However, there is a slight dip in prices for tickets booked a day or two before departure

```
# Group by Departure Time and Arrival Time and calculate average price
avg_price_by_time = df_pyspark.groupBy("departure_time",
"arrival_time").agg({"price": "avg"})

# Convert PySpark DataFrame to Pandas DataFrame for visualization
avg_price_by_time_pd = avg_price_by_time.toPandas()

# Visualization
plt.figure(figsize=(14, 8))
sns.heatmap(
    avg_price_by_time_pd.pivot("departure_time", "arrival_time",
"avg(price)"),
    annot=True,
    cmap="YlGnBu",
    fmt=".2f"
)
plt.title("Average Ticket Price by Departure and Arrival Time")
plt.xlabel("Arrival Time")
plt.ylabel("Departure Time")
plt.show()
```

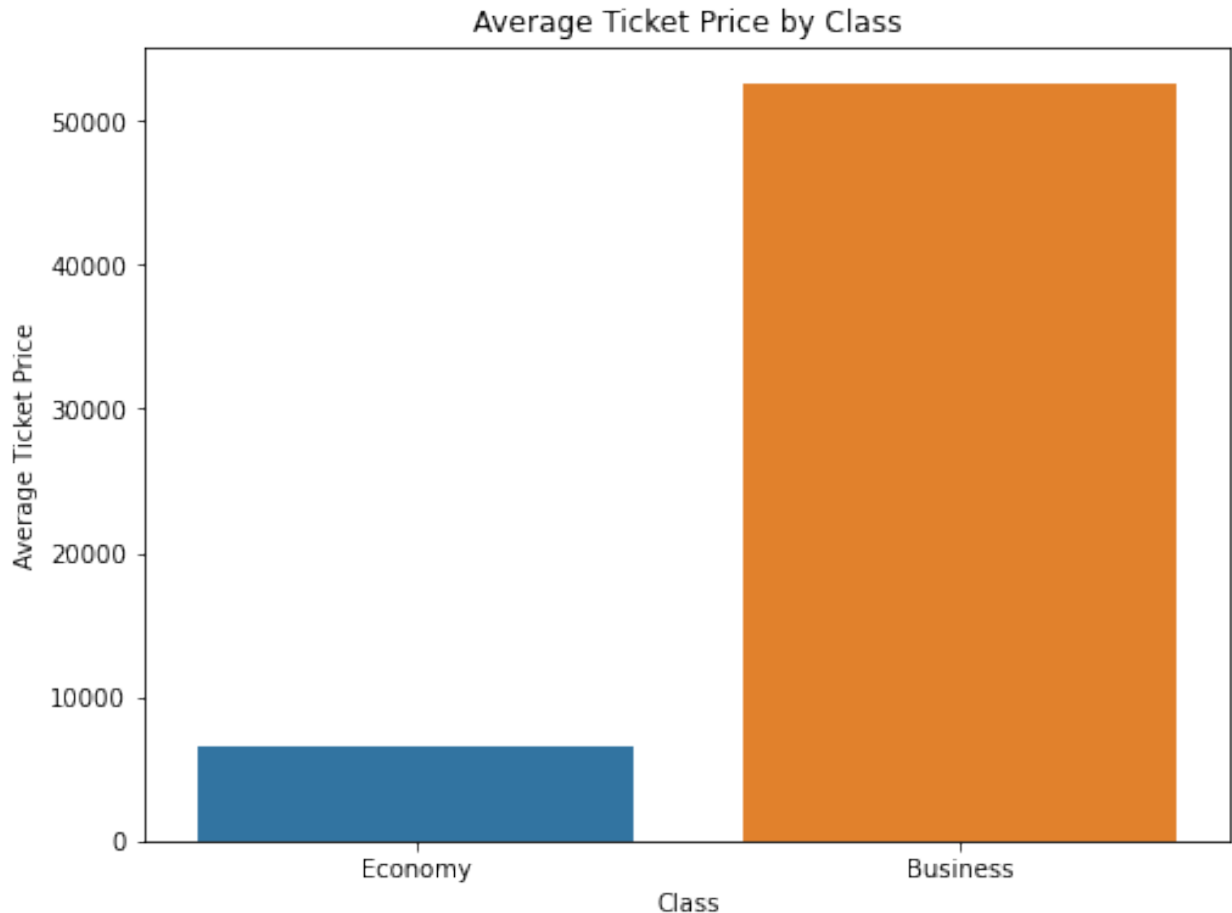


Flights departing in the early morning or late evening, especially with midday arrivals, tend to have lower prices. In contrast, peak travel times during the day exhibit higher average prices.

```
# Group by Class and calculate average price
avg_price_by_class = df_pyspark.groupBy("class").agg({"price": "avg"})

# Convert PySpark DataFrame to Pandas DataFrame for visualization
avg_price_by_class_pd = avg_price_by_class.toPandas()

# Visualization
plt.figure(figsize=(8, 6))
sns.barplot(x="class", y="avg(price)", data=avg_price_by_class_pd)
plt.title("Average Ticket Price by Class")
plt.xlabel("Class")
plt.ylabel("Average Ticket Price")
plt.show()
```

Business class tickets are considerably more expensive than Economy class tickets

```
# Create a function to use for creating new column

def identify_same_pairs(route):
    cities = route.split(" to ")
    return "-".join(sorted(cities))

# Group by Source City and Destination City and calculate average price
avg_price_by_route = df_pyspark.groupBy("source_city",
"destination_city").agg({"price": "avg"})

# Convert PySpark DataFrame to Pandas DataFrame for visualization
avg_price_by_route_pd = avg_price_by_route.toPandas()

# Add "Route" and "Sorted Route" columns to use for sorting
avg_price_by_route_pd["Route"] = avg_price_by_route_pd["source_city"]
+ " to " + avg_price_by_route_pd["destination_city"]
avg_price_by_route_pd["Sorted Route"] =
avg_price_by_route_pd["Route"].apply(identify_same_pairs)
```

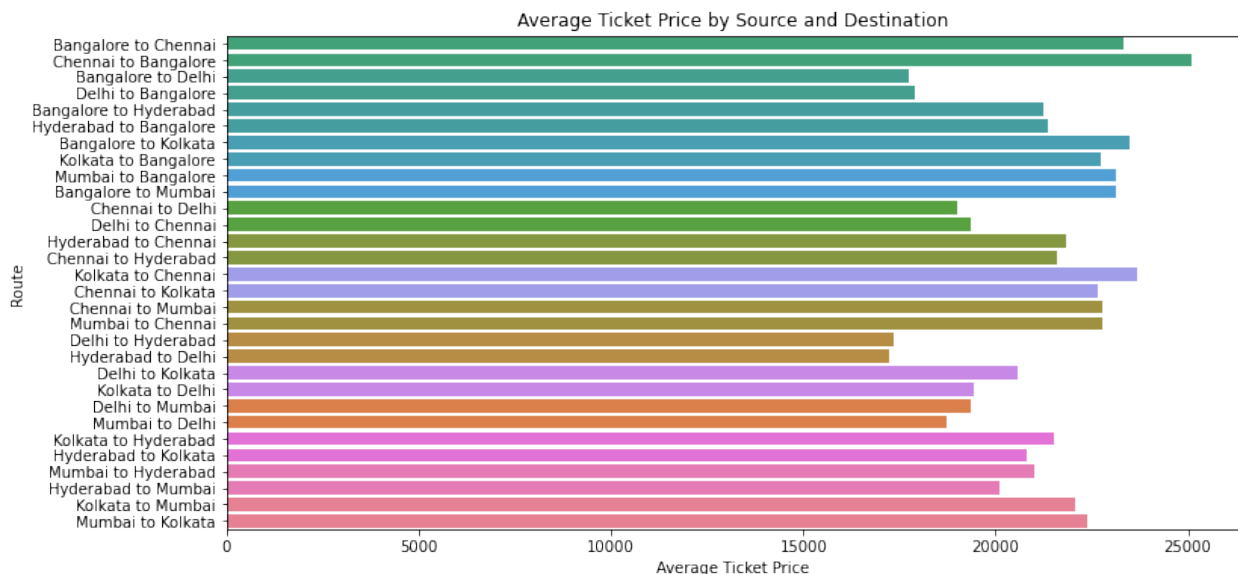
```

# Sort DataFrame based on the new "Route" column
avg_price_by_route_pd_sorted = (
    avg_price_by_route_pd.sort_values(by="Sorted Route")
    .reset_index(drop=True)
)

# Create a color palette
color_palette = sns.color_palette("husl",
n_colors=len(avg_price_by_route_pd_sorted) // 2)

# Visualization
plt.figure(figsize=(12, 6))
sns.barplot(
    x="avg(price)",
    y="Route",
    data=avg_price_by_route_pd_sorted,
    palette=sorted(color_palette * 2) # Repeat the color palette for
each pair
)
plt.title("Average Ticket Price by Source and Destination")
plt.xlabel("Average Ticket Price")
plt.ylabel("Route")
plt.show()

```



routes involving Delhi have lower prices while routes involving Chennai have higher prices than other cities.