HO CHI MINH NATIONAL UNIVERSITY - UNIVERSITY OF SCIENCE

DEPARTMENT OF MATHEMATICS AND COMPUTER SCIENCE

….□□✍….

TECHNICAL REPORT

Topic: Visualizing the airfare data set of airlines in India and developing some predictive models.

Instructor: Nguyễn Tiến Đạt

Class: 20TTH

Group members:

20110161 – Nguyễn Lê Công Duy

20110342 – Nguyễn Thanh Trúc

19110139 – Nguyễn Song Nhật

Ho Chi Minh City, May 2023

INDEX

PREFACE

Currently, airfare pricing is a priority concern worldwide as demand increases rapidly every year. There are many reasons for this phenomenon, but it is mainly due to tourism—especially domestic tourism—which leads to customers needing a dataset with highly reliable analysis and predictions. This is a dataset extracted from the "Ease my trip" website of 6 famous airlines in India.

There are many studies surrounding this pricing issue, particularly regarding which factor plays the most important role. This is because a flight's price is determined by a combination of factors, including brand, flight duration, number of stops, and the time interval between booking and flying.

In this report, the team will attempt to solve each small question by using a variety of charts to find the most general answers. At the end of the paper, there will be a section using machine learning models to predict prices and evaluate which algorithm is more optimal for the Indian airfare dataset.

PYTHON CODE & DETAILED ANALYSIS

Step 1: DATA UNDERSTANDING

Start with loading dataset

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
C:\Users\PC\anaconda3\lib\site-packages\scipy\__init__.py:146: UserWarning: A NumPy version >=1.16.5 and <1.23.0 is require
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
```

In [2]:
```
df = pd.read_csv("archive/Clean_Dataset.csv")
```

In [3]:
```
df.head()
```

Out[3]:

	Unnamed: 0	airline	flight	source_city	departure_time	stops	arrival_time	destination_city	class	duration	days_left	price
0	0	SpiceJet	SG-8709	Delhi	Evening	zero	Night	Mumbai	Economy	2.17	1	5953
1	1	SpiceJet	SG-8157	Delhi	Early_Morning	zero	Morning	Mumbai	Economy	2.33	1	5953
2	2	AirAsia	I5-764	Delhi	Early_Morning	zero	Early_Morning	Mumbai	Economy	2.17	1	5956
3	3	Vistara	UK-995	Delhi	Morning	zero	Afternoon	Mumbai	Economy	2.25	1	5955
4	4	Vistara	UK-963	Delhi	Morning	zero	Morning	Mumbai	Economy	2.33	1	5955

In [4]:
```
df = df.drop('Unnamed: 0', axis=1) # Bỏ cột dư thừa
```

In [5]:
```
df.head()
```

Out[5]:

	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

In [6]:
```
# Print the type of each columns
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 300153 entries, 0 to 300152
Data columns (total 11 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   airline           300153 non-null  object
 1   flight            300153 non-null  object
 2   source_city       300153 non-null  object
 3   departure_time    300153 non-null  object
 4   stops             300153 non-null  object
 5   arrival_time      300153 non-null  object
 6   destination_city  300153 non-null  object
 7   class             300153 non-null  object
 8   duration          300153 non-null  float64
 9   days_left         300153 non-null  int64
 10  price             300153 non-null  int64
dtypes: float64(1), int64(2), object(8)
memory usage: 25.2+ MB
```

<u>Step 2</u>: Data Processing

In [7]:
```
# count the values of each airline
df1=df.groupby(['flight','airline'],as_index=False).count()
df1.airline.value_counts()
```

Out[7]:
```
airline
Indigo        704
Air_India     218
GO_FIRST      205
SpiceJet      186
Vistara       133
AirAsia       115
Name: count, dtype: int64
```

Generally, **Indigo** is the most popular airline among the six, while **AirAsia** is the least popular.

In [8]:
```
for column in df.columns:
    if column != 'duration' and column != 'price' and column != 'days_left' and column != 'flight':
        print(f"{column}: \n{df[column].unique()}\n\n")
# In ra các loại giá trị trong mỗi cột
```

```
airline:
['SpiceJet' 'AirAsia' 'Vistara' 'GO_FIRST' 'Indigo' 'Air_India']


source_city:
['Delhi' 'Mumbai' 'Bangalore' 'Kolkata' 'Hyderabad' 'Chennai']


departure_time:
['Evening' 'Early_Morning' 'Morning' 'Afternoon' 'Night' 'Late_Night']


stops:
['zero' 'one' 'two_or_more']


arrival_time:
['Night' 'Morning' 'Early_Morning' 'Afternoon' 'Evening' 'Late_Night']


destination_city:
['Mumbai' 'Bangalore' 'Kolkata' 'Hyderabad' 'Chennai' 'Delhi']


class:
['Economy' 'Business']
```

Step 3: Data Analysis

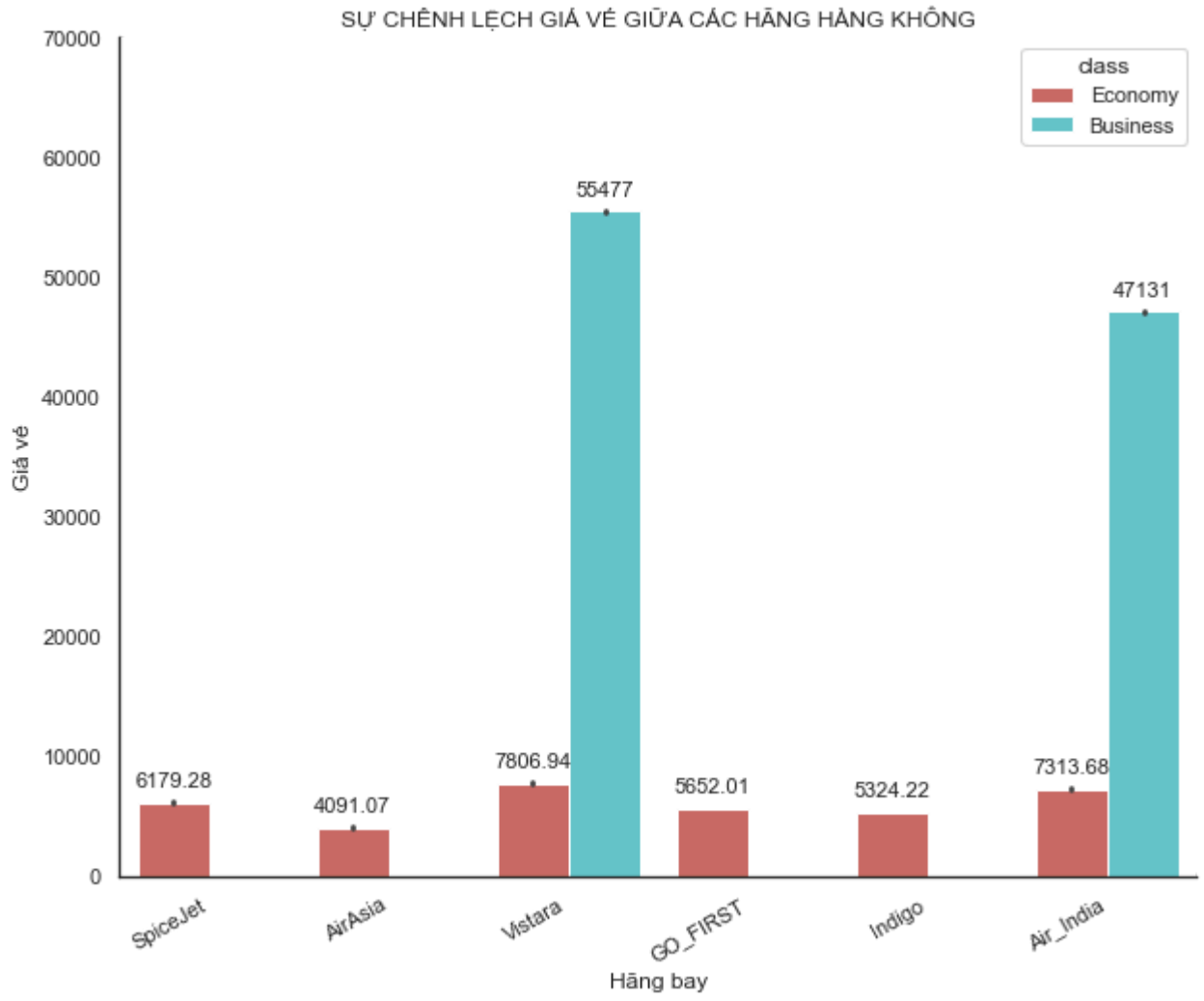
In [9]:
```
import matplotlib.pyplot as plt
import seaborn as sns

custom_params = {"axes.spines.right": False, "axes.spines.top": False}
sns.set_theme(style="white", rc=custom_params)
```

Tìm hiểu các đặc trưng giữa giá vé và các yếu tố khác trong data

3.1 Price differences and Class comparison

In [10]:
```
# vẽ đồ thị so sánh giá vé giữa các hãng hàng không
plt.figure(figsize = (10,8))
ax = sns.barplot(data=df, x='airline', y='price', hue='class',palette='hls')
for i in ax.containers :
    ax.bar_label(i,padding= 6,
                 label_type = 'edge',
                 fontsize = 11)
plt.ylabel('Giá vé',fontsize=12)
plt.xticks(rotation=30, ha = 'right')
plt.xlabel('Hãng bay',fontsize=12)
plt.ylim(0,70000)
plt.title('SỰ CHÊNH LỆCH GIÁ VÉ GIỮA CÁC HÃNG HÀNG KHÔNG',fontsize=12)
```

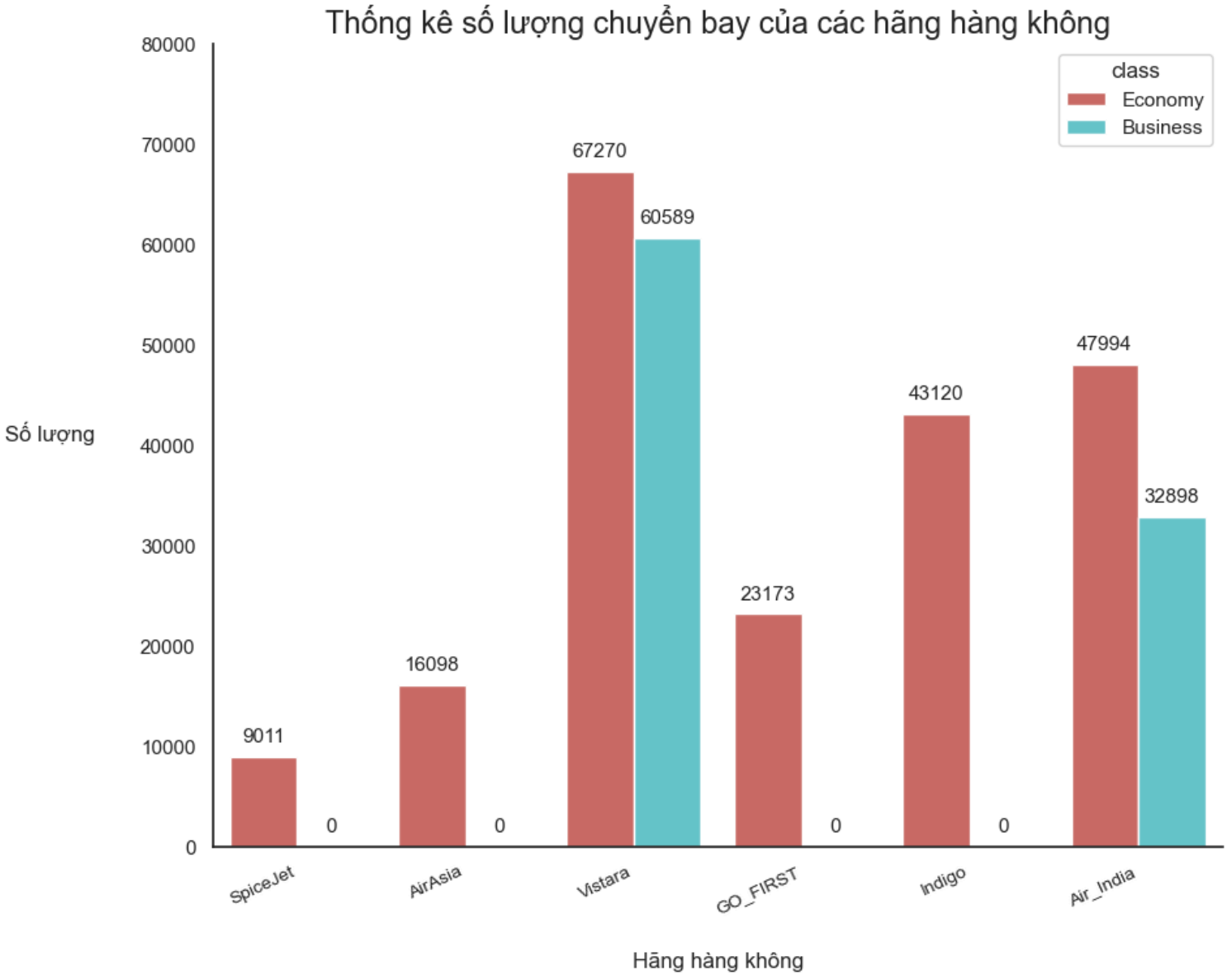
Out[10]:
```
Text(0.5, 1.0, 'SỰ CHÊNH LỆCH GIÁ VÉ GIỮA CÁC HÃNG HÀNG KHÔNG')
```



Prices vary by airline. In Economy, Vistara is most frequent and AirAsia is least. For Business class, only Vistara and Air India provide service.

3.2 Flight Statistics

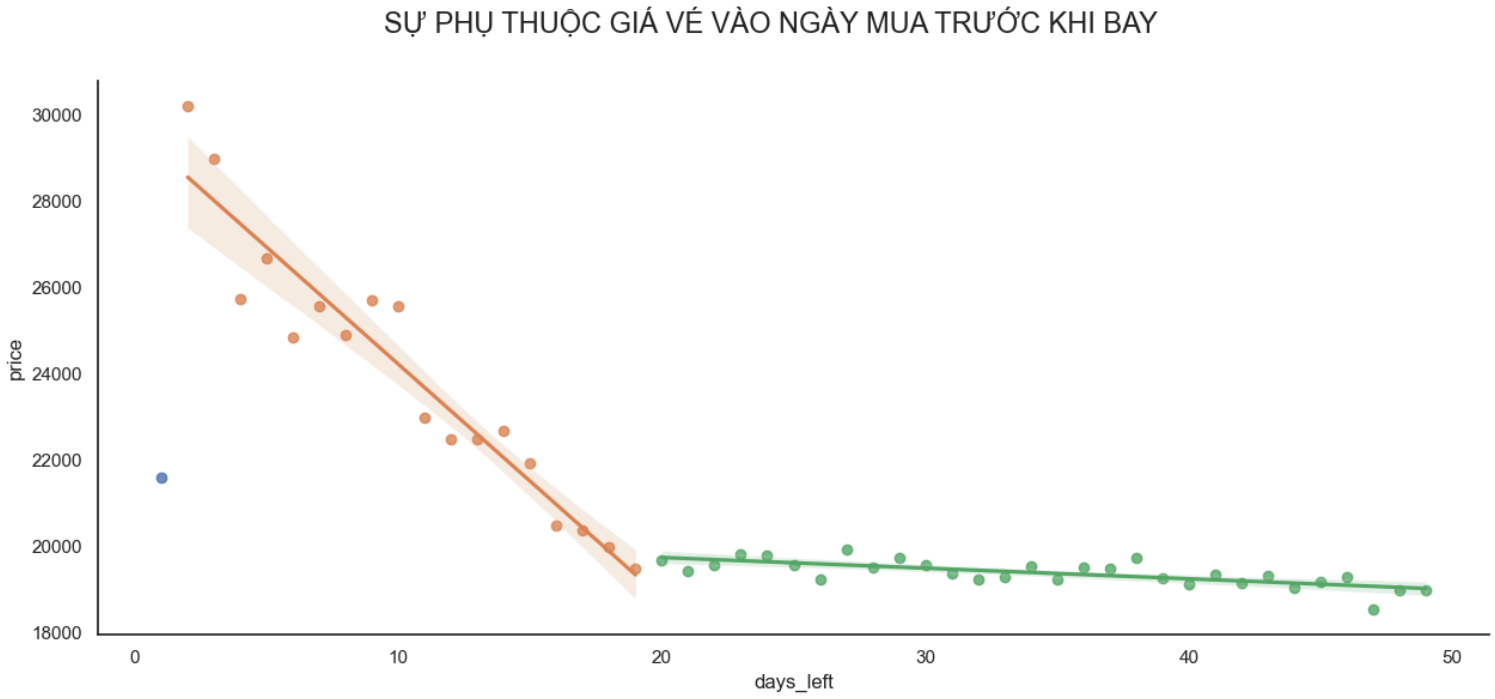
In [11]:
```
plt.figure(figsize=(10,8))
ax = sns.countplot(x = df['airline'],palette='hls', hue = df["class"])
plt.title('Thống kê số lượng chuyến bay của các hãng hàng không',fontsize=17)
for i in ax.containers :
    ax.bar_label(i,padding= 6,
                 label_type = 'edge',
                 fontsize = 11)
plt.ylim(0,80000)
plt.xlabel('Hãng hàng không',fontsize=12, labelpad= 20)
plt.ylabel('Số lượng',fontsize=12, rotation = 0, labelpad = 50, loc = 'center')
plt.xticks(rotation=25,
           horizontalalignment='right',
           fontweight='light',
           fontsize=10)
plt.show()
```



Vistara leads significantly in both categories, especially in Business class (84% higher than the runner-up). This confirms **Vistara** is the largest and most prominent airline in the dataset.

3.3 Price dependency on Days Left before departure

```
In [12]: df_temp = df.groupby(['days_left'])['price'].mean().reset_index()
         plt.figure(figsize=(15,6)).suptitle('SỰ PHỤ THUỘC GIÁ VÉ VÀO NGÀY MUA TRƯỚC KHI BAY', fontsize=17)
         ax = plt.axes()
         sns.regplot(x=df_temp.loc[df_temp["days_left"]==1].days_left, y=df_temp.loc[df_temp["days_left"]==1].price, fit_reg=False, ax=ax)
         sns.regplot(x=df_temp.loc[(df_temp["days_left"]>1)&(df_temp["days_left"]<20)].days_left, y=df_temp.loc[(df_temp["days_left"]>1)&(df_temp["days_left"]<20)].price, fit_reg=True, ax=ax)
         sns.regplot(x=df_temp.loc[df_temp["days_left"]>=20].days_left, y=df_temp.loc[df_temp["days_left"]>=20].price, fit_reg=True, ax=ax)

Out[12]: <AxesSubplot:xlabel='days_left', ylabel='price'>
```

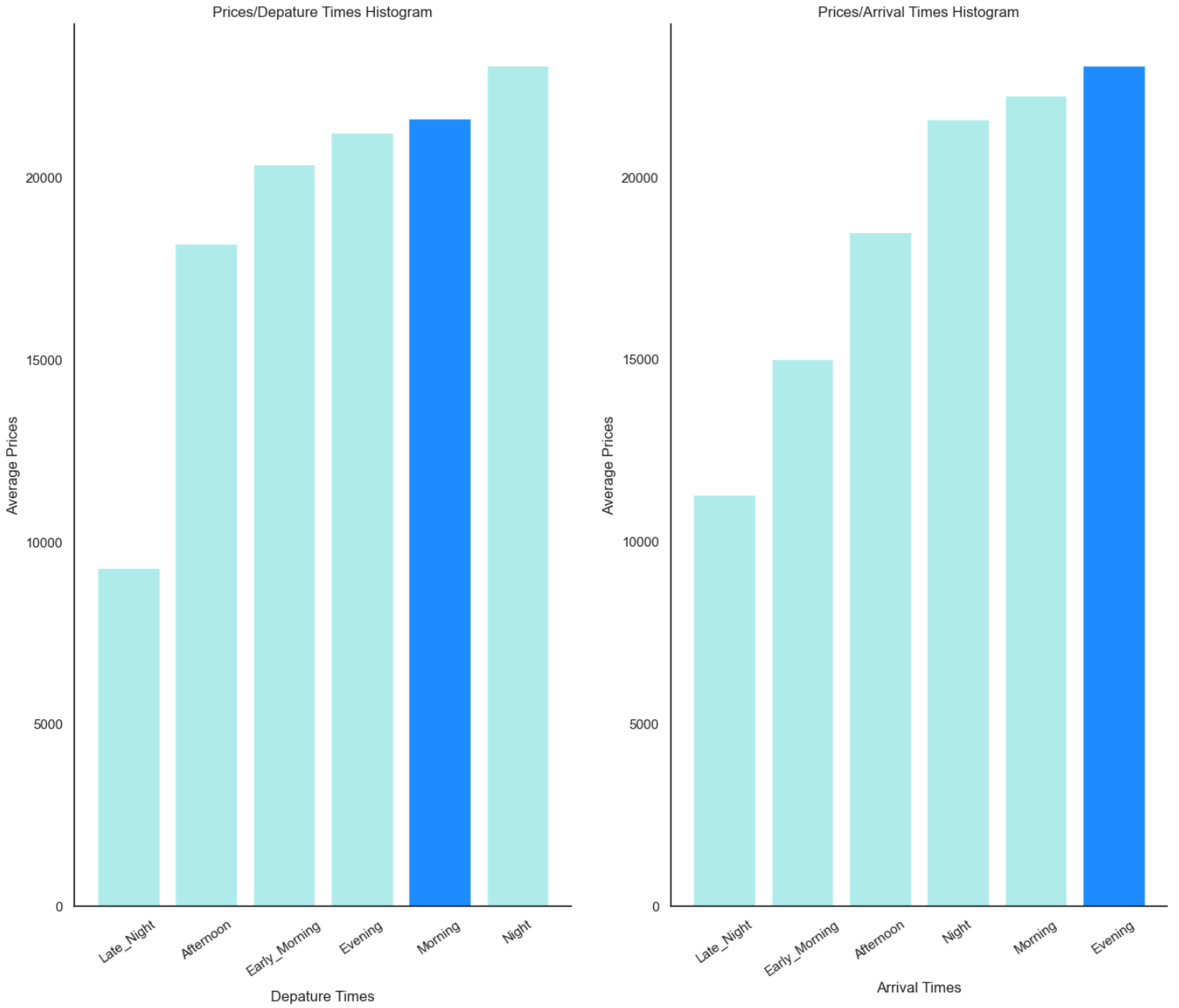


Based on data analysis, it's clear that airfares increase significantly when purchased within two weeks of the scheduled departure date, with Business Class tickets potentially experiencing higher price increases compared to Economy Class tickets.

To ensure the most cost-effective option, it's recommended to purchase Economy Class tickets at least 20 days (2 weeks) before departure, while for Business Class tickets, it's advisable to purchase them at least ten days before the scheduled departure date.

3.4 Do ticket prices change depending on departure and arrival times?

In [14]:
```
depature_time_price = df.groupby('departure_time')['price'].mean().round(0).sort_values(ascending = True)
arrival_time_price = df.groupby('arrival_time')['price'].mean().round(0).sort_values(ascending = True)

plt.figure(figsize =(16,13))
plt.subplot(1,2,1)
plt.bar(depature_time_price.index, depature_time_price.values,color=['paleturquoise', 'paleturquoise','paleturquoise', 'paleturquoise', 'dodgerblue'])
plt.title("Prices/Depature Times Histogram")
plt.xlabel("Depature Times", labelpad=10)
plt.ylabel("Average Prices")
plt.xticks(rotation = 35)

plt.subplot(1,2,2)
plt.bar(arrival_time_price.index, arrival_time_price.values,color=['paleturquoise', 'paleturquoise','paleturquoise', 'paleturquoise', 'paleturquoise', 'dodgerblue'])
plt.title("Prices/Arrival Times Histogram")
plt.xlabel("Arrival Times", labelpad=3)
plt.ylabel("Average Prices")
plt.xticks(rotation = 35)

plt.show()
```



Both departure and arrival times show the same result: late at night is when tickets are cheapest.

The most expensive departure times are evening, while arrival times are in the afternoon, so passengers should try to avoid these hours when booking tickets.

3.5 How does the price change with variations between different departure and arrival points?

In [13]:
```
#removing outliers
df1=df.drop(["flight","departure_time","arrival_time","stops"],axis=1)
df2=df1.copy()
#feature engineering destination and source in on column
df2["source_and_destination"] = df2[["source_city", "destination_city"]].apply(lambda x: "-".join(x), axis=1)
df2.drop(["source_city","destination_city"],axis=1,inplace=True)
#source and destination is made it into a single column
```

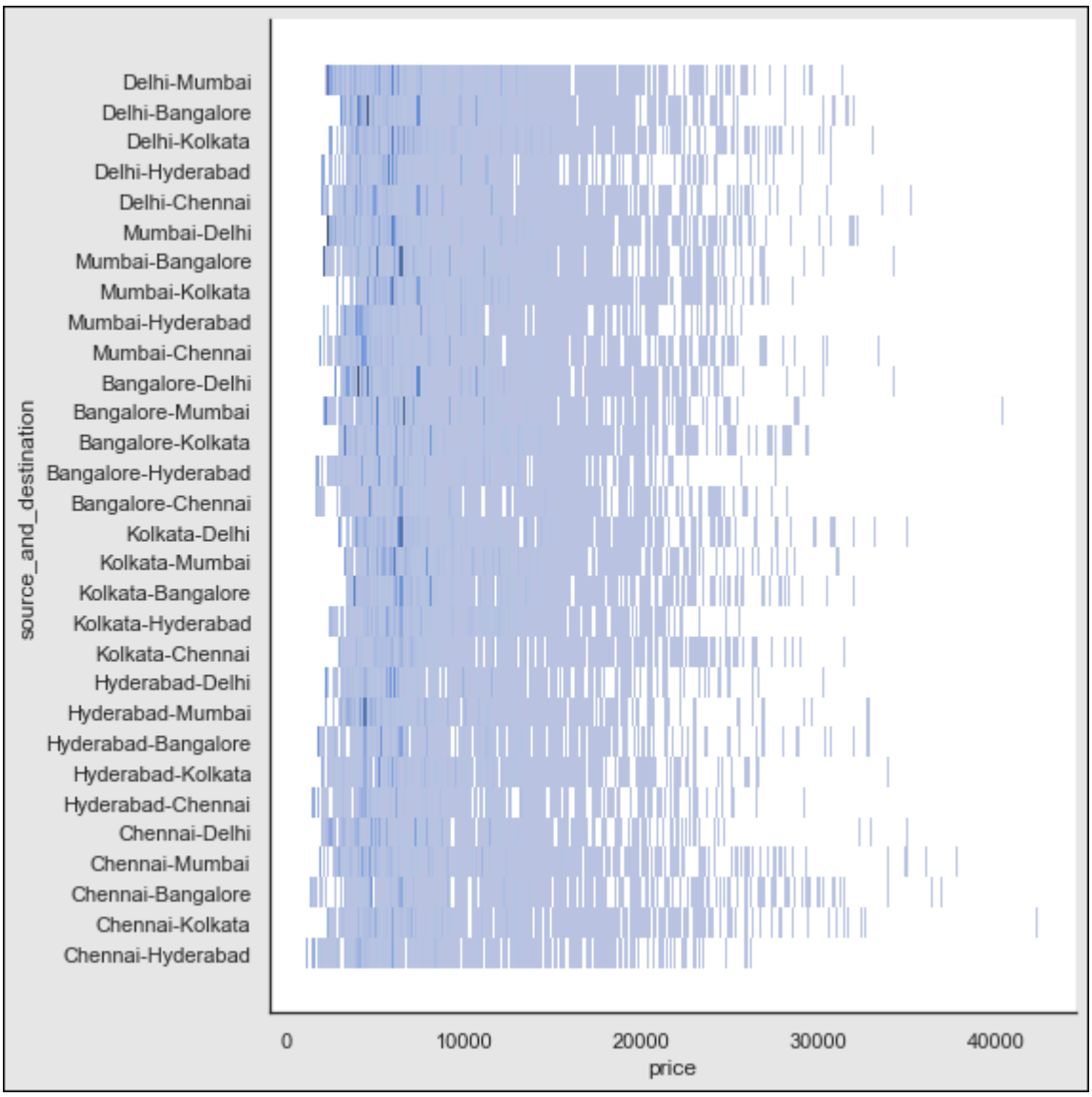
In [14]:
```
#seperating dataframes as business and economy for better analysis
df_economy=df2[df2["class"]=="Economy"]
df_business=df2[df2["class"]=="Business"]
dftemp=df1.reset_index() #temporary dataframe used for counts
df_economy.head()
#df_business.head()
```

Out[14]:

	airline	class	duration	days_left	price	source_and_destination
0	SpiceJet	Economy	2.17	1	5953	Delhi-Mumbai
1	SpiceJet	Economy	2.33	1	5953	Delhi-Mumbai
2	AirAsia	Economy	2.17	1	5956	Delhi-Mumbai
3	Vistara	Economy	2.25	1	5955	Delhi-Mumbai
4	Vistara	Economy	2.33	1	5955	Delhi-Mumbai

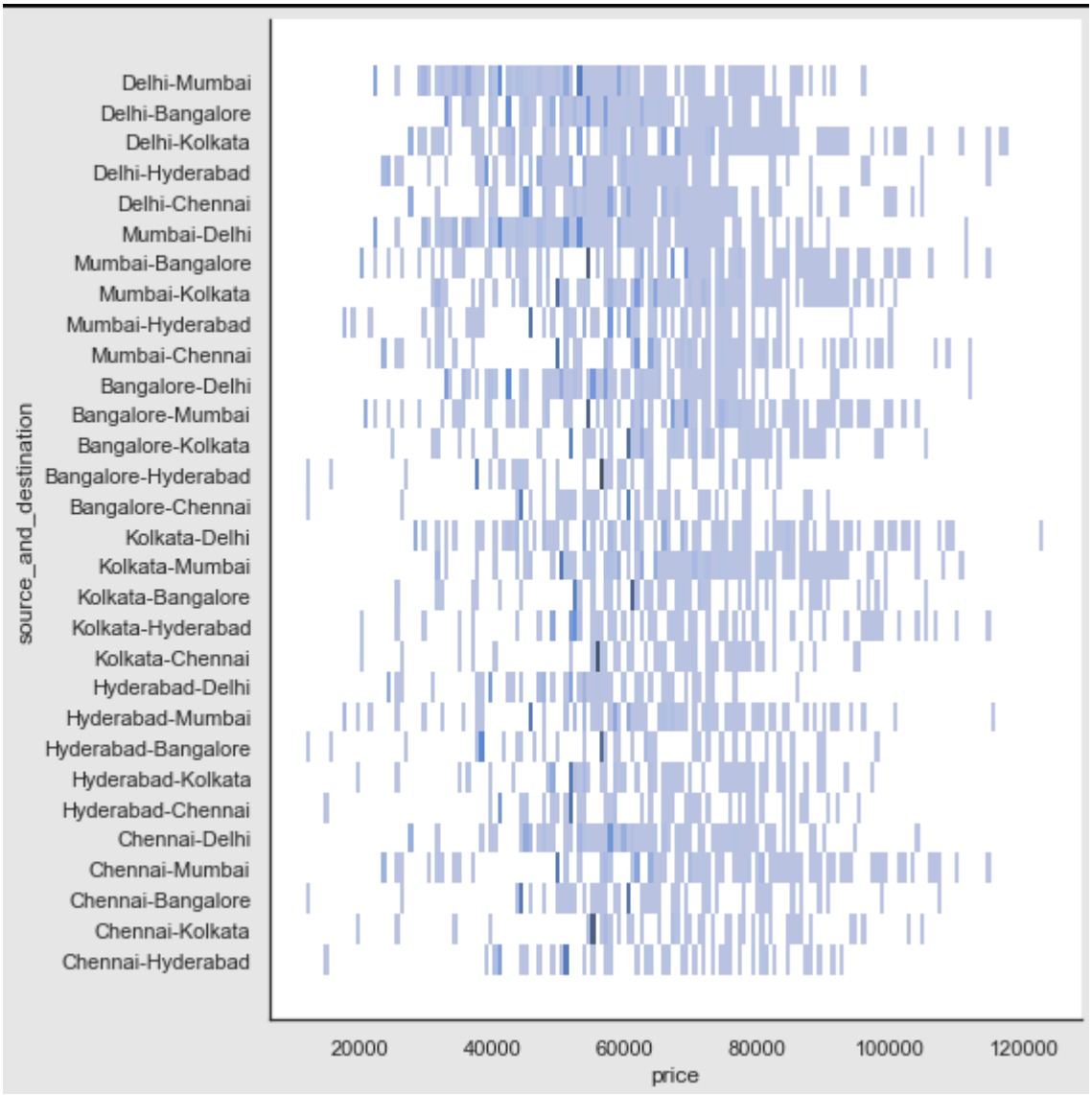
In [15]:
```
fig = plt.figure(figsize=(8,10))
sns.displot(data = df_economy,x='price',y='source_and_destination', height = 8)
```

Out[15]:
```
<seaborn.axisgrid.FacetGrid at 0x2059d2c3070>
```



In [16]:
```
fig = plt.figure(figsize=(8,10))
sns.displot(data=df_business,x='price',y='source_and_destination', height = 8)
```

Out[16]:
```
<seaborn.axisgrid.FacetGrid at 0x205ba812040>
```

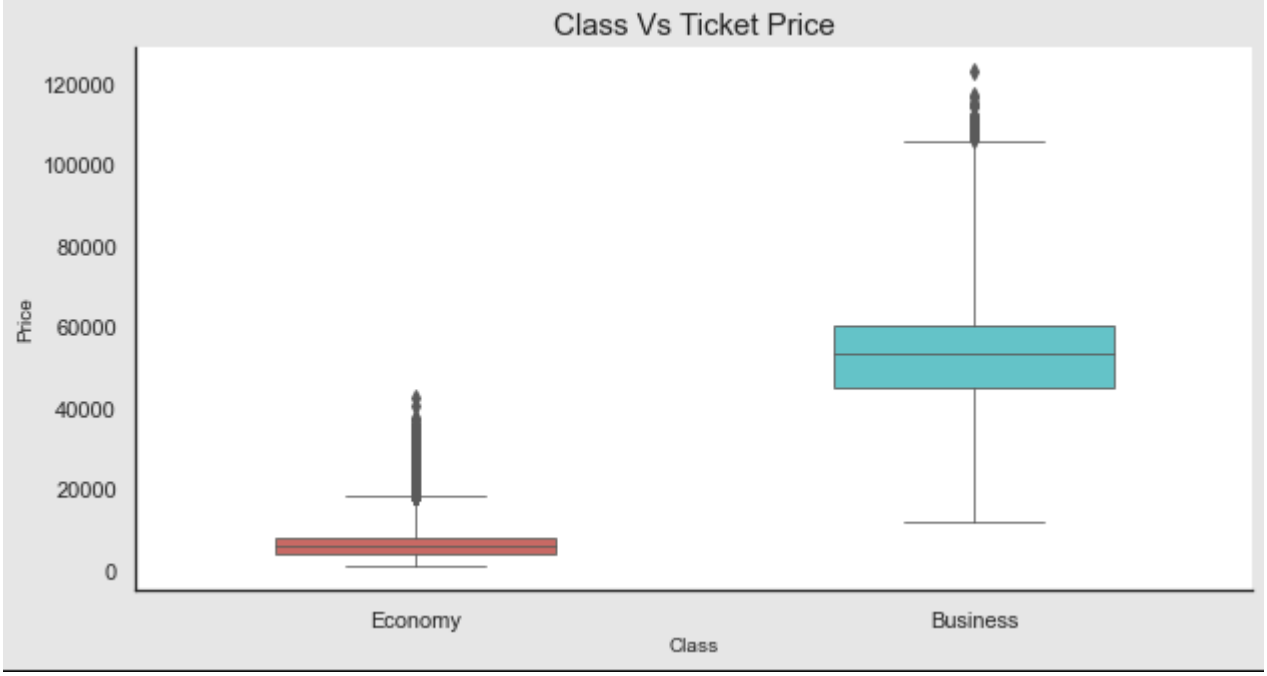


For economy class, we see more trips starting in Delhi and Mumbai compared to other locations. For business class, trips are less frequent and scattered across a wider range of destinations.

Ticket prices do not depend on the departure and destination points. However, observing the graph, we see a few outliers. Nevertheless, we can still see a general trend in ticket prices when considering the departure location.

3.6 Compare the prices of Business Class and Economy Class tickets.

In [17]:
```
# Chốt
# draw the boxplot to compare between the different class with ticket price
plt.figure(figsize=(10,5))
sns.boxplot(x='class',y='price',data=df,palette='hls', width = 0.5, whis = 3, linewidth=0.8)
plt.title('Class Vs Ticket Price',fontsize=15)
plt.xlabel('Class',fontsize=10)
plt.ylabel('Price',fontsize=10)
plt.yticks()
plt.show()
```



On average, economy class tickets are still significantly cheaper than business class tickets. However, it is still possible to find economy class tickets that are as expensive as or more expensive than business class tickets.

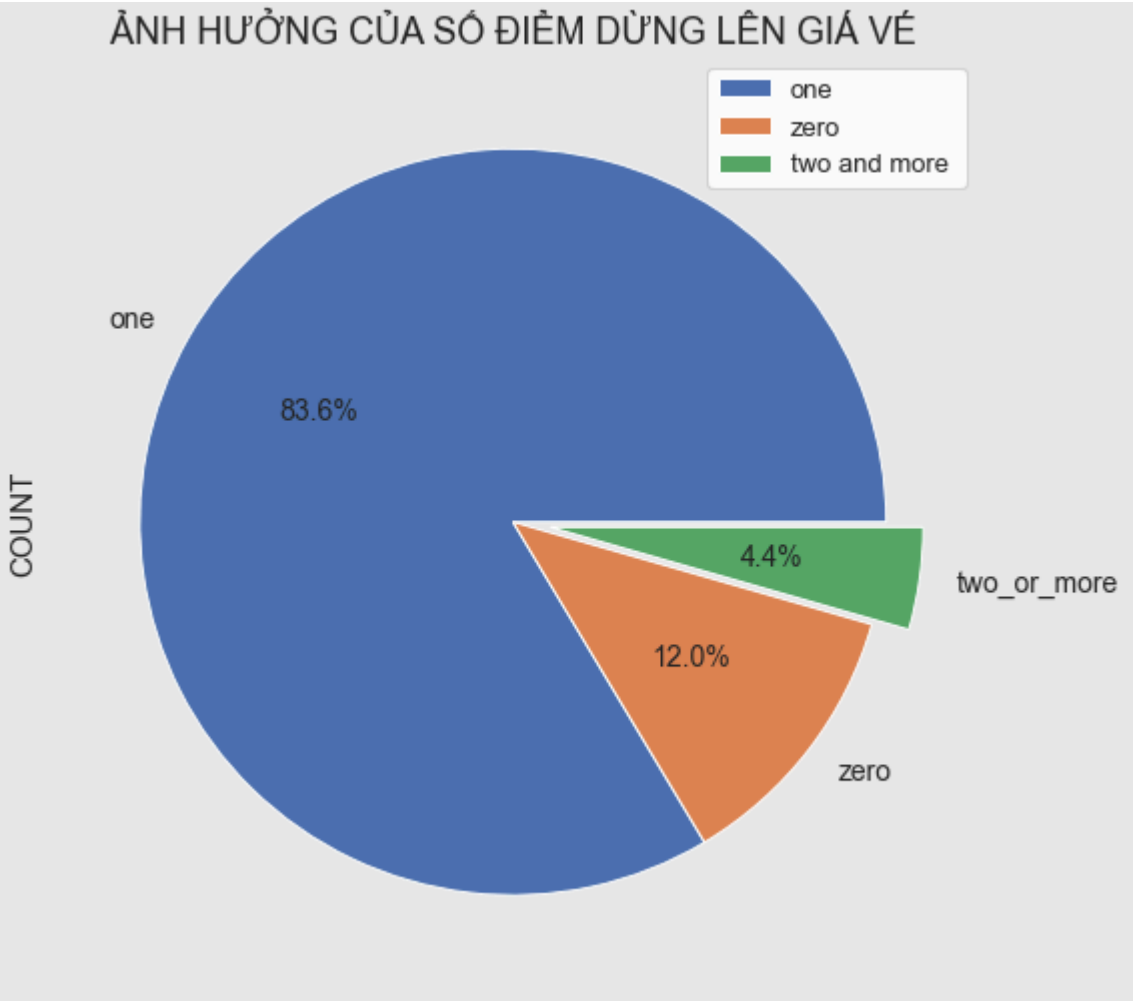
Similarly, in business class, there are tickets that are the same price or cheaper than economy class tickets.

3.7 The number of stops affects ticket prices.

In [18]:
```
# check the values of "stops"
df["stops"].unique()
```

Out[18]:
```
array(['zero', 'one', 'two_or_more'], dtype=object)
```

In [19]:
```
fig = plt.figure(figsize  = (8,10))
explode = (0,0,0.1)
df.stops.value_counts().plot(kind='pie',autopct="%1.1f%%",explode = explode, fontsize = 14)
plt.ylabel('COUNT', fontsize = 15)
plt.title("ẢNH HƯỞNG CỦA SỐ ĐIỂM DỪNG LÊN GIÁ VÉ", fontsize = 18)
plt.legend(['one','zero','two and more '], loc = 'upper right', fontsize = 13)
plt.tight_layout()
```



From this chart, it can be seen that flights with a stopover are generally the most expensive among all flight types. However, flights with no stopovers are the cheapest.

Furthermore, from this chart, we can see that Vistara has the highest airfare compared to other airlines. In addition, Air India has relatively stable prices compared to the general average.

Most passengers choose airlines like Vistara and Air India over the others, even though the chart shows that Air Asia also has relatively low prices, but it is not as popular.

3.8 How does the price change with variations between different departure and arrival points?

In [20]:
```
# Import Dataset
X = df.loc[:,"duration" ].values
y = df.loc[:,"price"].values
```

In [21]:
```
X = X.reshape(-1,1)
```

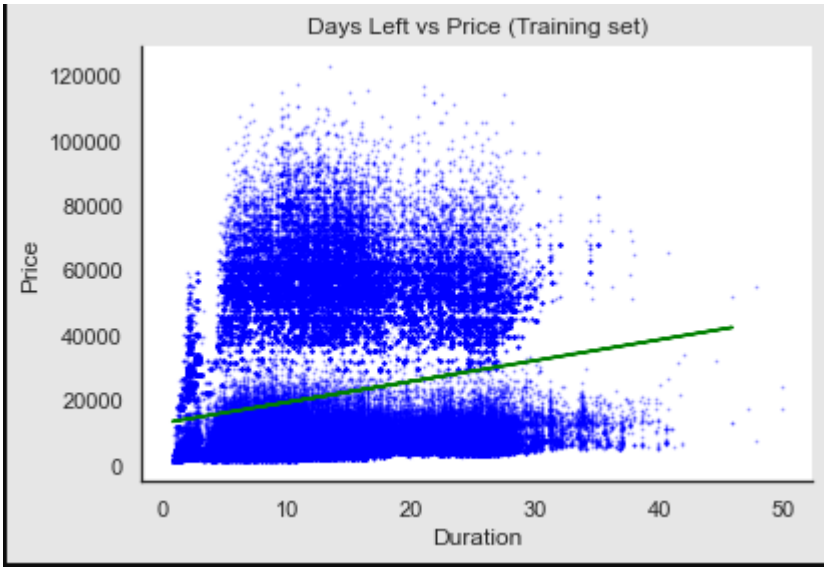
In [22]:
```
# Separating the dataset as Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.8, random_state = 22)
```

In [23]:
```
# Linear Regression


# Training the Simple Linear Regression model on the Training set
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)

# Result prediction on Test set
y_pred = regressor.predict(X_test)
```

In [24]:
```
# Đánh giá mô hình
from sklearn.metrics import r2_score
r2_score(y_test, y_pred)
```

Out[24]:
```
0.041700175722420574
```

In [25]:
```
# Result visualization on Training Set
plt.scatter(X, y, color = 'blue', s = 0.7, alpha= 0.3)
plt.plot(X_train, regressor.predict(X_train), color = 'green')
plt.title('Days Left vs Price (Training set)')
plt.xlabel('Duration')
plt.ylabel('Price')
plt.show()
```



The model shows that ticket prices tend to increase as flight time increases, and the impact of the "flight time" factor is quite clear.

Step 4: Build a model using linear regression, decision tree, and random forest.

4.1 Model Preprocessing

In [26]:
```
# Use the Label Encoder to transform all columns which is object to int
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
for col in df.columns:
    if df[col].dtype=='object':
        df[col]=le.fit_transform(df[col])
```

In [27]:
```
def prepare_X_y(df):
    """
    Feature engineering and create X and y
    :param df: pandas dataframe
    :return: (X, y) output feature matrix (dataframe), target (series)
    """
    # Todo: Split data into X and y (using sklearn train_test_split). Return two dataframes
    feature_names = df.columns.tolist()
    feature_names.remove("price")
    X = df[feature_names].values
    y = df.price.values
    return X, y
```

In [28]:
```
# print the function above
X,y = prepare_X_y(df)
```

In [29]:
```
# spli data
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import MinMaxScaler
RANDOM_STATE = 42
TRAIN_SIZE = 0.7

X_train,X_test ,Y_train, Y_test = train_test_split(X, y, train_size=TRAIN_SIZE, random_state=RANDOM_STATE)
```

In [30]:
```
from sklearn.preprocessing import MinMaxScaler
```

In [31]:
```
# norm data
mmscaler=MinMaxScaler()
X_train=mmscaler.fit_transform(X_train)
X_test=mmscaler.fit_transform(X_test)
```

4.2 Using Decision Tree

In [32]:
```
# apply decision tree
modeldcr = DecisionTreeRegressor()
pipe = Pipeline(steps=[("mmscaler", mmscaler), ("modeldcr", modeldcr)])
modeldcr.fit(X_train,Y_train)
```

Out[32]:
```
DecisionTreeRegressor()
```

In [33]:
```
# calculate y_pred
y_pred = modeldcr.predict(X_test)
```

In [33]:
```
# calculate y_pred
y_pred = modeldcr.predict(X_test)
```

In [34]:
```
from sklearn.metrics import mean_squared_error, r2_score

# Evaluate the mean squared error of the model
mse = mean_squared_error(Y_test, y_pred)
print(f"Mean squared error: {mse}")

# Evaluate the R2 score of the model
r2 = r2_score(Y_test, y_pred)
print(f"R2 score: {r2}")
```

```
Mean squared error: 13794189.16689001
R2 score: 0.9731978103867023
```

In [35]:
```
# checking the difference of ticket price from one source_city to destination_city
df.groupby(['airline','source_city','destination_city'],as_index=False)['price'].mean().head(10)
```

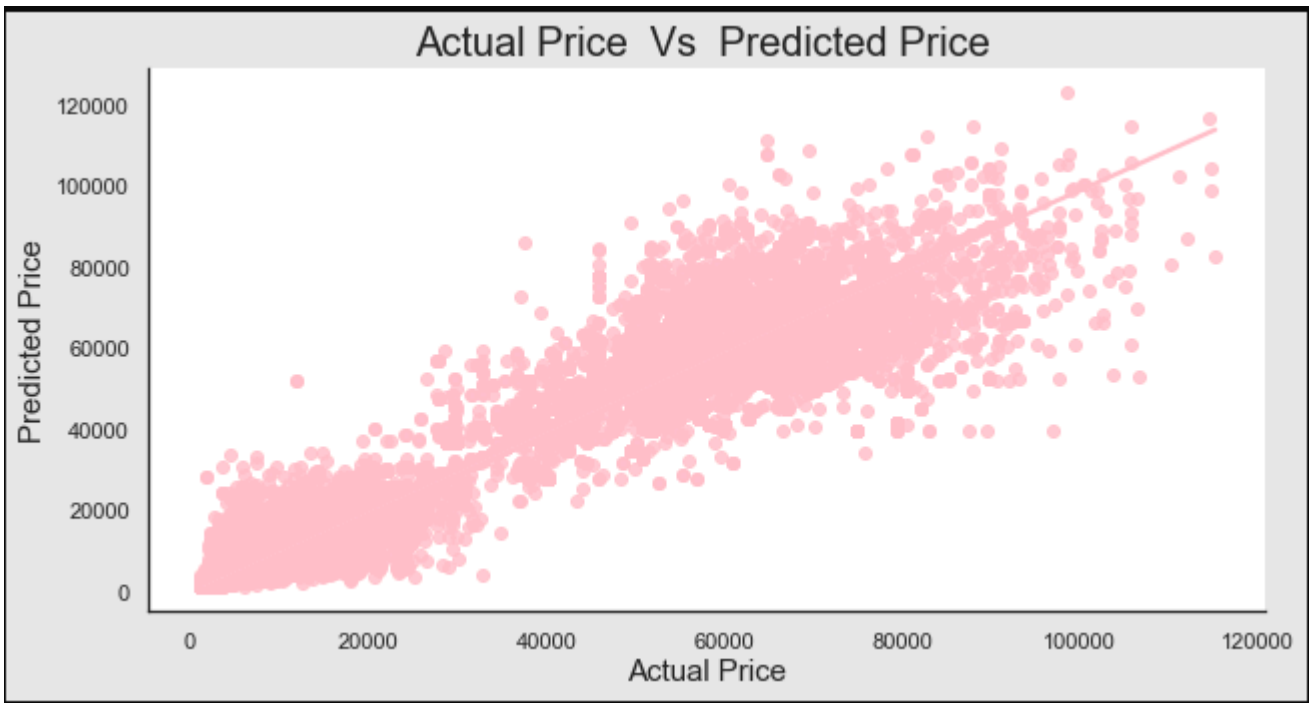
Out[35]:

	airline	source_city	destination_city	price
0	0	0	1	2073.043478
1	0	0	2	4807.092426
2	0	0	3	2931.494792
3	0	0	4	4443.468160
4	0	0	5	3342.385350
5	0	1	0	1914.760870
6	0	1	2	3697.314003
7	0	1	3	2053.182540
8	0	1	4	3682.338762
9	0	1	5	2691.100000

In [36]:
```
# copy new data into "df_bk"
df_bk=df.copy()
```

In [37]:
```
# calculate the output
out=pd.DataFrame({'Price_actual':Y_test,'Price_pred':y_pred})
result=df_bk.merge(out,left_index=True,right_index=True)
```

In [38]:
```
# plot the regplot
plt.figure(figsize=(10,5))
sns.regplot(x='Price_actual',y='Price_pred',data=result,color='pink')
plt.title('Actual Price  Vs  Predicted Price ',fontsize=20)
plt.xlabel('Actual Price',fontsize=15)
plt.ylabel('Predicted Price',fontsize=15)
plt.show()
```



Overall, this model still has quite a few outliers on the regplot, so the decision tree model is not yet an accurate model for predicting this pattern.

4.3 Using Random - Forest

In [39]:
```
# apply random forest
from sklearn.ensemble import RandomForestRegressor
modelrfr = RandomForestRegressor()
pipe1 = Pipeline(steps=[("mmscaler", mmscaler), ("modelrfr", modelrfr)]) #Build a pipeline with a scaler and a model
modelrfr.fit(X_train,Y_train)
```

Out[39]:
```
RandomForestRegressor()
```

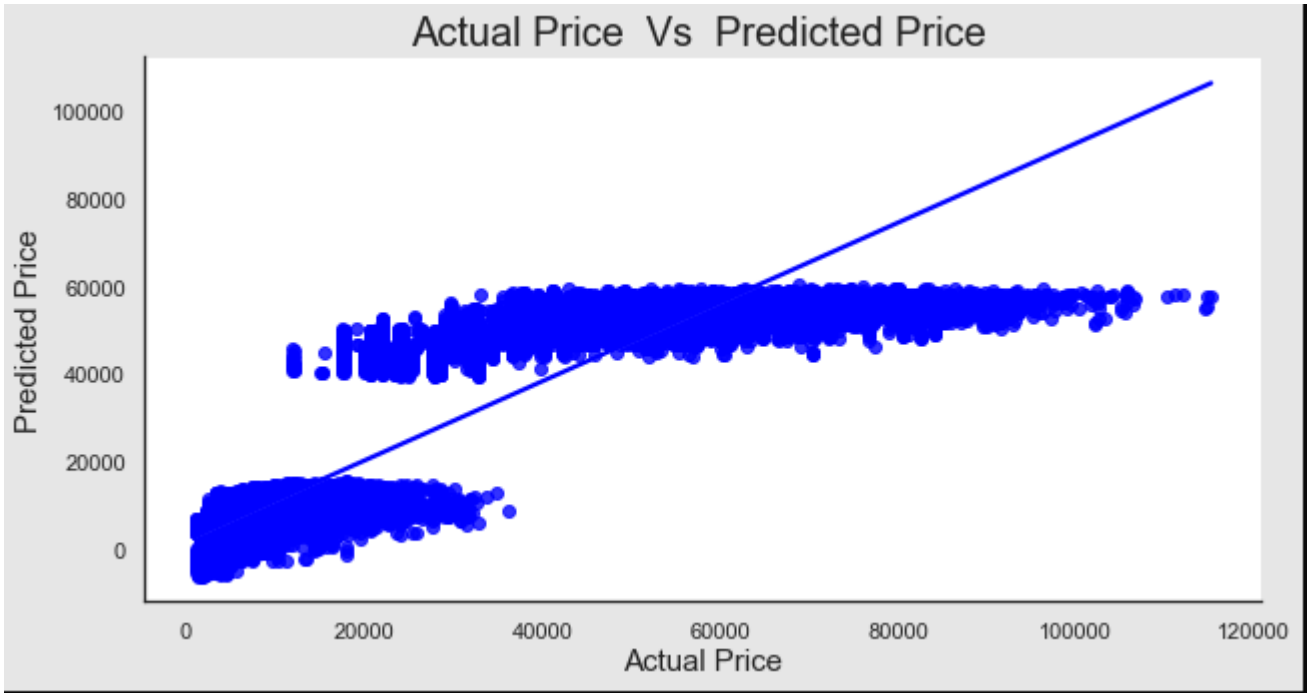
In [40]:
```
#calculate y_pred
y_pred1 = modelrfr.predict(X_test)
```

In [41]:
```
from sklearn.metrics import mean_squared_error, r2_score

# Evaluate the mean squared error of the model
mse = mean_squared_error(Y_test, y_pred1)
print(f"Mean squared error: {mse}")

# Evaluate the R2 score of the model
r2 = r2_score(Y_test, y_pred1)
print(f"R2 score: {r2}")
```

```
Mean squared error: 8219187.650908797
R2 score: 0.9840300706897874
```

In [42]:
```
# calculate "out"
out=pd.DataFrame({'Price_actual':Y_test,'Price_pred':y_pred1})
result=df_bk.merge(out,left_index=True,right_index=True)
```

In [48]:
```
# plot the regplot
plt.figure(figsize=(10,5))
sns.regplot(x='Price_actual',y='Price_pred',data=result,color='blue')
plt.title('Actual Price  Vs  Predicted Price ',fontsize=20)
plt.xlabel('Actual Price',fontsize=15)
plt.ylabel('Predicted Price',fontsize=15)
plt.show()
```



This model has fewer outliers, indicating that the random forest model is a good model for making predictions for this problem.

In addition, from this chart, we can see a positive linear relationship between the two variables, meaning that the price increases linearly when one variable changes, and the other variable changes.

4.4 Using Linear Regression

In [43]:
```
# apply the linear regression
from sklearn.linear_model import LinearRegression
modelmlg = LinearRegression()
pipe1 = Pipeline(steps=[("mmscaler", mmscaler), ("modelmlg", modelmlg)]) #Build a pipeline with a scaler and a model
modelmlg.fit(X_train,Y_train)
```

Out[43]:
```
LinearRegression()
```

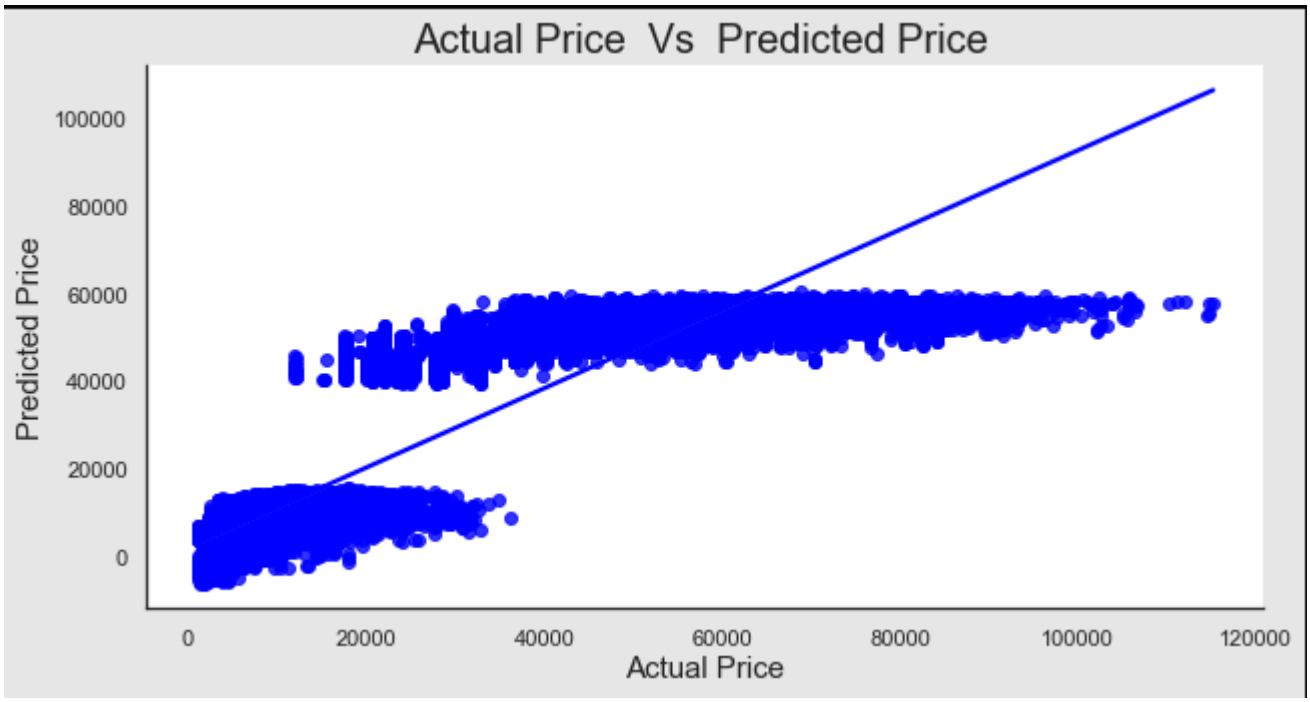
In [44]:
```
# calculate y_pred2
y_pred2 = modelmlg.predict(X_test)
```

In [45]:
```
from sklearn.metrics import mean_squared_error, r2_score

# Evaluate the mean squared error of the model
mse = mean_squared_error(Y_test, y_pred2)
print(f"Mean squared error: {mse}")

# Evaluate the R2 score of the model
r2 = r2_score(Y_test, y_pred2)
print(f"R2 score: {r2}")
```

```
Mean squared error: 49070241.26461302
R2 score: 0.9046562364171967
```

In [46]:
```
# calculate "output" and display it by dataframe
out=pd.DataFrame({'Price_actual':Y_test,'Price_pred':y_pred2})
result=df_bk.merge(out,left_index=True,right_index=True)
```

In [47]:
```
# plot regplot
plt.figure(figsize=(10,5))
sns.regplot(x='Price_actual',y='Price_pred',data=result,color='blue')
plt.title('Actual Price  Vs  Predicted Price ',fontsize=20)
plt.xlabel('Actual Price',fontsize=15)
plt.ylabel('Predicted Price',fontsize=15)
plt.show()
```



From the chart, we can see that using a linear regression model is unsuitable for this type of problem because there are many outliers that deviate from the regression line, thus leading to inaccurate price prediction results.

CONCLUSIONS

If you're looking to save money on airfare, there are several factors to consider before buying. One of the most important is the timing of your flights. It's worth noting that the time of day and day of the week can significantly impact ticket prices. Generally, flights departing and arriving at night are usually cheaper than those departing and arriving during peak hours.

For flights with a 2-3 hour journey time with no layovers, late-night departures and arrivals are often a more viable option for finding cheap tickets. These flights typically depart after 9 or 10 PM and arrive at their destination in the early morning. While not the most convenient in terms of schedule, they can be the best way to save money on airfare.

Another factor to consider when booking your flight is the number of layovers. In some cases, flights with more layovers may be cheaper than flights with fewer layovers, depending on the route and airline. You should do some research to find out which airlines offer the best deals on your chosen route and how many layovers they typically have.

When it comes to finding the best deals on airline tickets, it's important to be flexible and broaden your observation of different options. By considering factors such as time, number of layovers, and airline options, you can increase your chances of finding a great deal on your next flight.

REFERENCES:

https://www.kaggle.com/datasets/shubhambathwal/flight-price-prediction?select=business.csv&fbclid=IwAR334B0t9upMMAYsekqKpE8zKqdI9OvLoBA6GH0aI_bUO-0NFwDU1hiAf4A