Flight Price Prediction

The objective of the project is to analyze the flight booking dataset obtained from "Ease My Trip" website and predict the flight price. 'Easemytrip' is an internet platform for booking flight tickets, and hence a platform that potential passengers use to buy tickets.

Octoparse scraping tool was used to extract data from the website. Data was collected in two parts: one for economy class tickets and another for business class tickets. A total of 300261 distinct flight booking options was extracted from the site. Data was collected for 50 days, from February 11th to March 31st, 2022.

Dataset contains information about flight booking options from the website Easemytrip for flight travel between India's top 6 metro cities. There are 300261 datapoints and 11 features in the cleaned dataset.

In [1]:
```
# Libraries data handling
import numpy as np
import pandas as pd

# Librarires for visualization
import matplotlib.pyplot as plt
import seaborn as sns

sns.set(font = 'Serif', style = 'white', rc = {'axes.facecolor':'#f1f1f1', 'figure.f
```

In [2]:
```
# Reading the data
df = pd.read_csv('data.csv')

# Droping the first column, which is index
df.drop(columns='Unnamed: 0', inplace=True)

# Displaying the data
df.head()
```

Out[2]:

	airline	flight	source_city	departure_time	stops	arrival_time	destination_city	class	dura
0	SpiceJet	SG-8709	Delhi	Evening	zero	Night	Mumbai	Economy	
1	SpiceJet	SG-8157	Delhi	Early_Morning	zero	Morning	Mumbai	Economy	
2	AirAsia	I5-764	Delhi	Early_Morning	zero	Early_Morning	Mumbai	Economy	
3	Vistara	UK-995	Delhi	Morning	zero	Afternoon	Mumbai	Economy	
4	Vistara	UK-963	Delhi	Morning	zero	Morning	Mumbai	Economy	

Dataset Description

The various features of the cleaned dataset are explained below: 1) Airline: The name of the airline company is stored in the airline column. It is a categorical feature having 6 different airlines. 2) Flight: Flight stores information regarding the plane's flight code. It is a categorical feature. 3) Source City: City from which the flight takes off. It is a categorical feature having 6 unique cities. 4) Departure Time: This is a derived categorical feature obtained created by grouping time periods into bins. It stores information about the departure time and have 6 unique time labels. 5) Stops: A categorical feature with 3 distinct values that stores the numberof stops between the source and destination cities. 6) Arrival Time: This is a derived categorical feature created by grouping time intervals into bins. It has six distinct time labels and keeps information about the arrival time. 7) Destination City: City where the flight will land. It is a categorical feature having 6 unique cities. 8) Class: A categorical feature that contains information on seat class; it has two distinct values: Business and Economy. 9) Duration: A continuous feature that displays the overall amount of time it takes to travel between cities in hours. 10)Days Left: This is a derived characteristic that is calculated by subtracting the trip dateby the booking date. 11) Price: Target variable stores information of the ticket price.

In [3]:

```
# Shape of the data
df.shape
```

Out[3]:

```
(300153, 11)
```

There are 10 features and one target variable and 300K rows

In [4]:

```
# Dataset info
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
```

The data type is correct for all the variables.

In [5]:

```
# Checking for missing values
df.isnull().sum()
```

Out[5]:

```
airline             0
flight              0
source_city         0
departure_time      0
stops               0
arrival_time        0
```

```
destination_city    0
class               0
duration            0
days_left           0
price               0
dtype: int64
```

There are no missing values.

In [6]:

```python
# Checking the mean, median, max
df.describe().T
```

Out[6]:

	count	mean	std	min	25%	50%	75%	max
duration	300153.0	12.221021	7.191997	0.83	6.83	11.25	16.17	49.83
days_left	300153.0	26.004751	13.561004	1.00	15.00	26.00	38.00	49.00
price	300153.0	20889.660523	22697.767366	1105.00	4783.00	7425.00	42521.00	123071.00

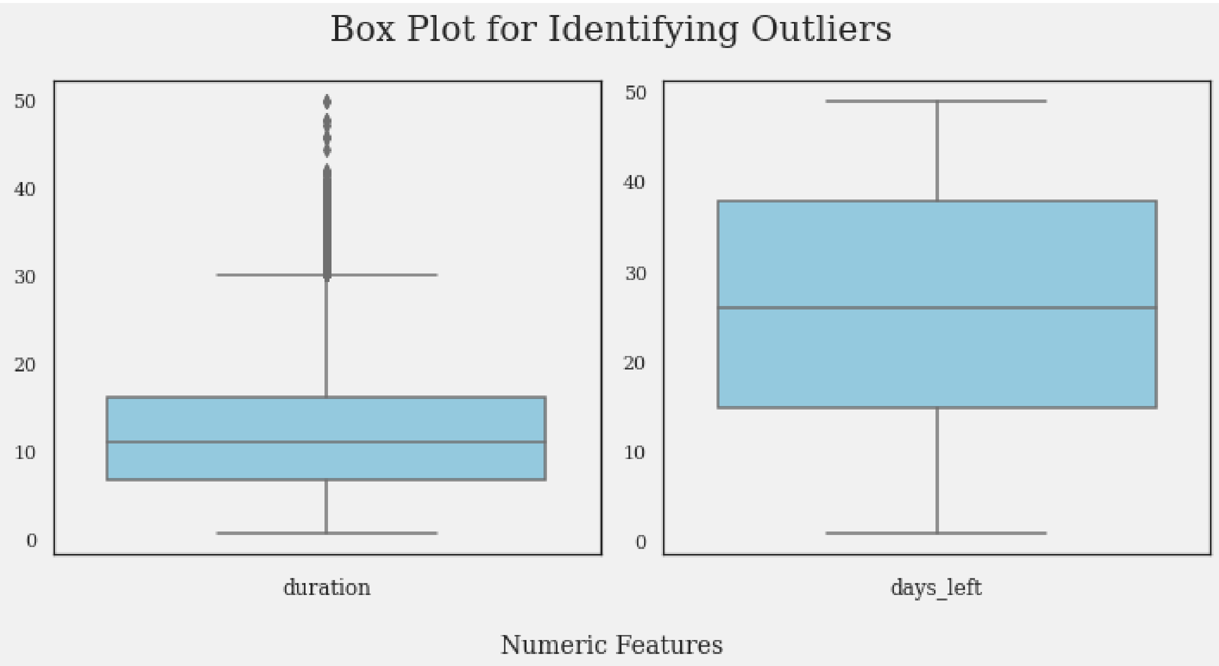
Exploratory Data Analysis

In [7]:

```python
numeric_var = ['duration', 'days_left']

fig, ax = plt.subplots(1,2, figsize = (10,5))
for axis, num_var in zip(ax, numeric_var):
    sns.boxplot(y = num_var,data = df, ax = axis, color = 'skyblue')
    axis.set_xlabel(f"{num_var}", fontsize = 12)
    axis.set_ylabel(None)

fig.suptitle('Box Plot for Identifying Outliers', fontsize = 20)
fig.text(0.5, -0.05, 'Numeric Features', ha = 'center', fontsize = 14)
plt.tight_layout()
```



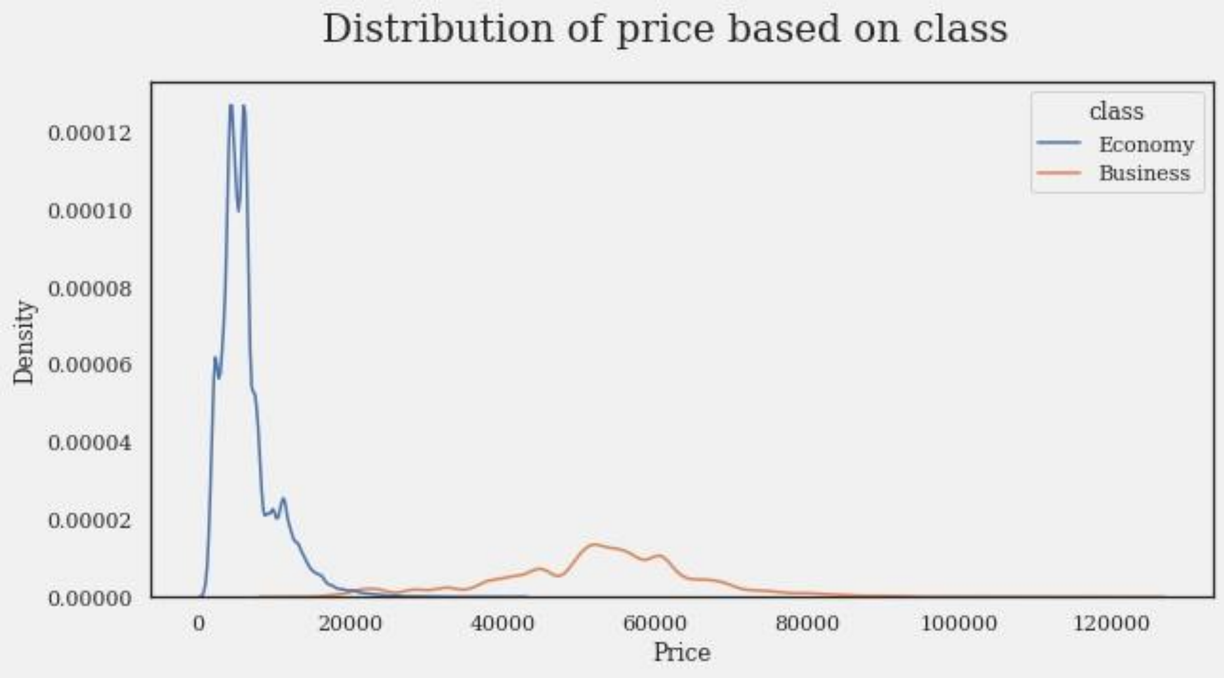
`duration` contains some values which fall beyond the IQR (Inter Quantile Range), but we must decide whether to call them outliers or not. For this case, I am considering 0.05 threshold for the outlier identification. Any data point which lies beyond 95 percentile is an outlier.

There can be instances where the `duration` is very high, but if we consider that, then model may not perform well.

In [8]:
```
# Considering 95% percentile for duration
df = df[df['duration'] <= df['duration'].quantile(0.95)]
```

How does the ticket price vary between Economy and Business class?

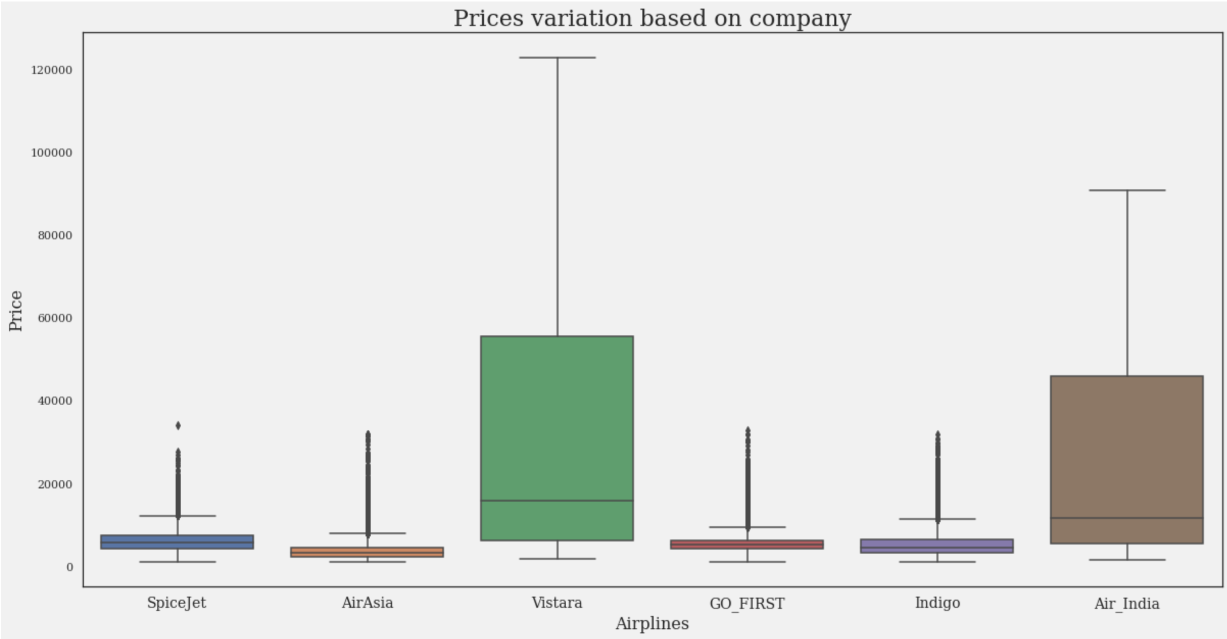
In [9]:
```
fig, ax = plt.subplots(1,1, figsize = (10,5))
sns.kdeplot(x='price', data=df, hue='class')
ax.set_xlabel('Price', fontsize=12)
fig.suptitle('Distribution of price based on class', fontsize = 20);
```



As obvious, the price of business class is much more than economy class. The distribution of price for business class is more spread than for economy class.

Does price vary with Airlines?

In [10]:
```
plt.figure(figsize=(20, 10))
ax = sns.boxplot(x='airline', y='price', data=df)
ax.set_ylabel('Price', fontsize=16)
ax.set_xlabel('Airplines', fontsize=16)
ax.set_xticklabels(df['airline'].unique(), fontsize=14)
ax.set_title('Prices variation based on company', fontsize=22);
```

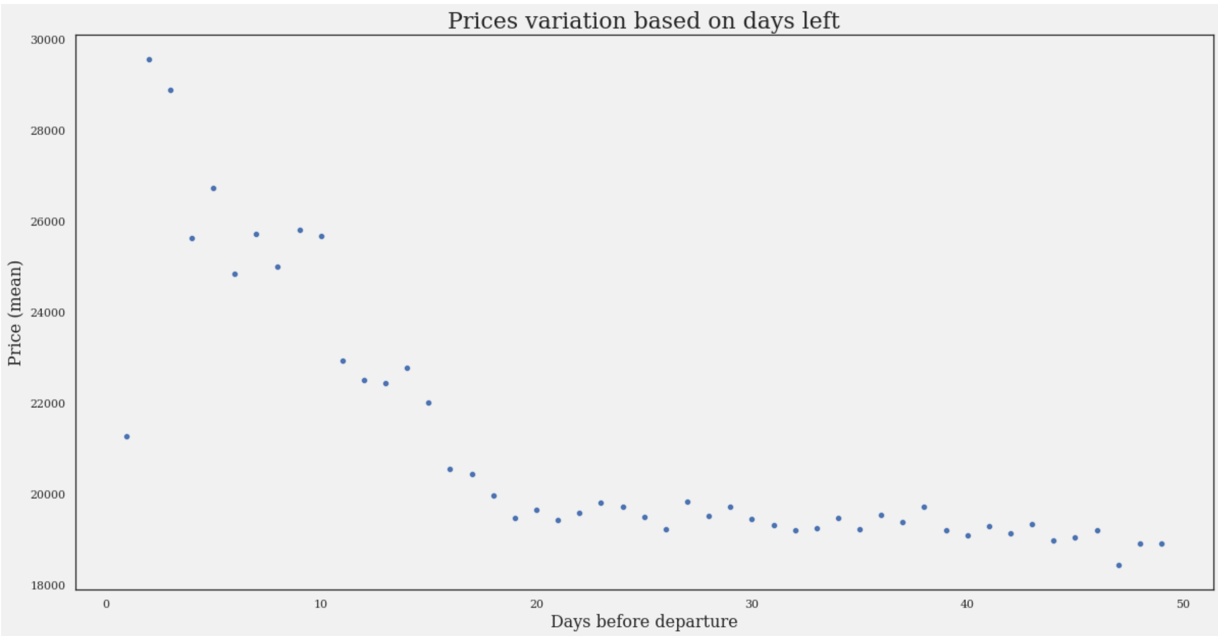


The median flight price for Indigo, GO First, Air Asia, and Spice Jet is almost same. For Vistara and Air India, it is more. Also, the flight price variation for these two airplanes is much more than others. It can be said that, in this dataset, Vistara is having the costliest flight price.

How is the price affected when tickets are bought in just 1 or 2 days before departure?

In [11]:

```
plt.figure(figsize=(20, 10))
ax = sns.scatterplot(x='days_left', y='price', data=df.groupby(['days_left'])['price
ax.set_ylabel('Price (mean)', fontsize=16)
ax.set_xlabel('Days before departure', fontsize=16)
ax.set_title('Prices variation based on days left', fontsize=22);
```

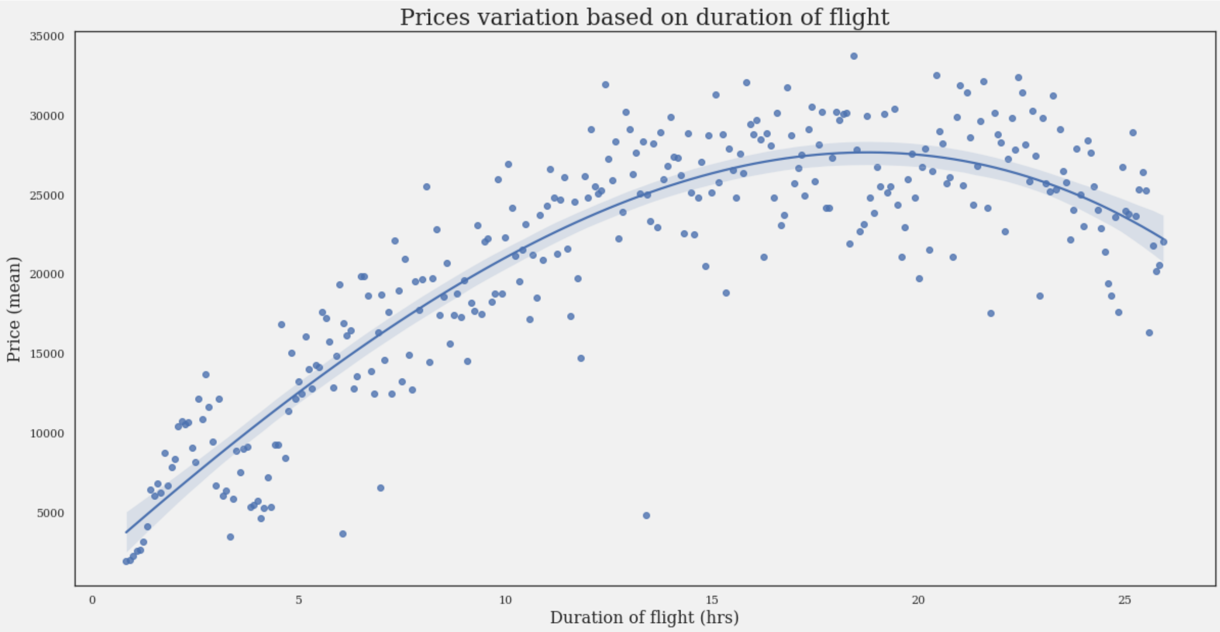


The price variation is not much if the flight is booked 20-50 days before, but as the difference between booking date and departure decreases below 20 days, the price starts to rise exponentially.

Does the price change with the duration of the flight?

In [12]:

```
plt.figure(figsize=(20, 10))
ax = sns.regplot(x='duration', y='price',
                 data=df.groupby(['duration'])['price'].mean().reset_index(),
                 order=3)
ax.set_ylabel('Price (mean)', fontsize=16)
ax.set_xlabel('Duration of flight (hrs)', fontsize=16)
ax.set_title('Prices variation based on duration of flight', fontsize=22);
```



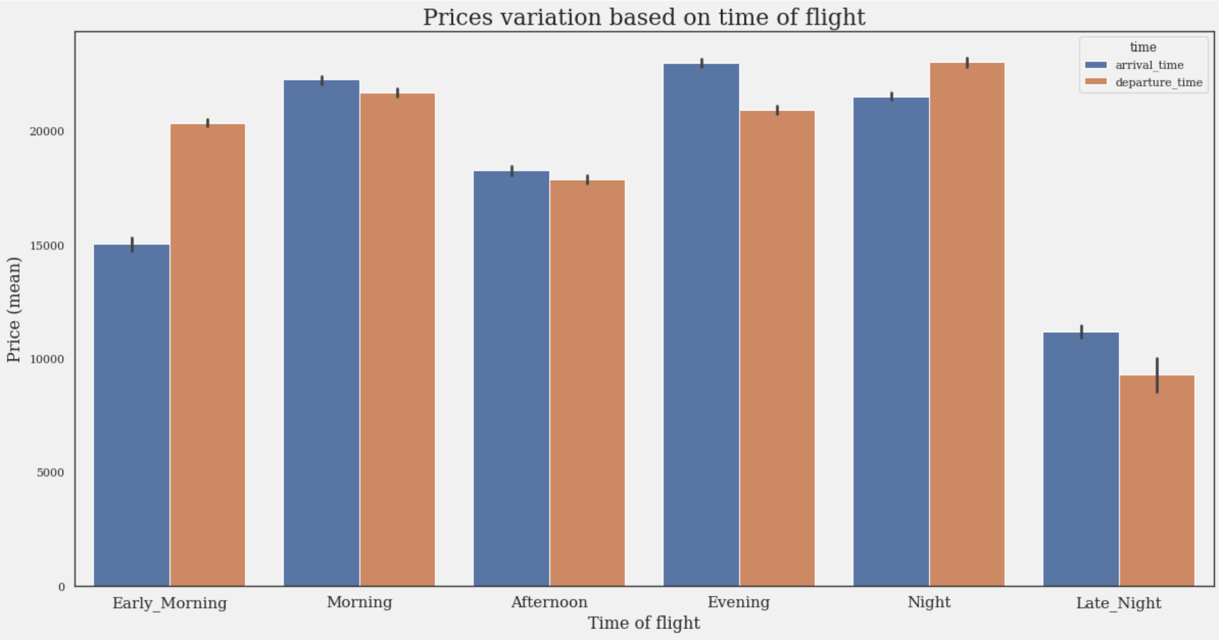
The price of the flight increases as the duration increase upto 20 hrs and after that it follows a downward trend. I think as the duration of flight is more, less people are interested in buying it,so the price drops.

Does ticket price change based on the departure time and arrival time?

In [13]:

```
df_time = df.melt(id_vars=['price'], value_vars=['arrival_time', 'departure_time'])
df_time.columns = ['price', 'time', 'time_1']

order = ['Early_Morning', 'Morning', 'Afternoon', 'Evening', 'Night', 'Late_Night']
plt.figure(figsize=(20, 10))
ax = sns.barplot(x='time_1', y='price', estimator=np.mean, data=df_time, hue='time',
ax.set_ylabel('Price (mean)', fontsize=16)
ax.set_xlabel('Time of flight', fontsize=16)
ax.set_xticklabels(order, fontsize=15)
ax.set_title('Prices variation based on time of flight', fontsize=22);
```



The early morning and late night flights are relatively cheaper. So, leaving late night and arriving early morning is the cheapest way to book a flight.

How the price changes with change in Source and Destination?

In [14]:
```
plt.figure(figsize=(20, 10))
ax = sns.relplot(x='destination_city', y="price", data=df, col="source_city", col_w
ax.fig.subplots_adjust(top=0.9)
ax.fig.suptitle('Airline prices based on the source and destination cities', fontsiz
```

```
<Figure size 1440x720 with 0 Axes>
```

The flight from Chennai to Bangalore are costlier than others, while boarding from Delhi is mostly cheaper. Also, boarding from Bangalore is costlier than others.

Machine Learning

In [15]:
```
# Data prepration for ML
from sklearn.model_selection import train_test_split

# Train and test
# As there are many unique values of the flight, I am removing it
X  = df.drop(columns=['price', 'flight'])
y = df['price']

# one hot encoding
X_encoded  = pd.get_dummies(X, drop_first=True)

X_train, X_test, y_train, y_test = train_test_split(X_encoded, y, train_size=0.8, ra
```

In [16]:
```
# Scaling the data
from sklearn.preprocessing import StandardScaler

scalar = StandardScaler()
X_train_scaled  =  pd.DataFrame(scalar.fit_transform(X_train),  columns=X_train.columns
X_test_scaled = pd.DataFrame(scalar.transform(X_test), columns=X_train.columns)
```

In [17]:
```
# Shape of the training data
X_train.shape
```

Out[17]:
```
(228576, 30)
```

Linear Regression

In [18]:
```
from sklearn.linear_model import LinearRegression

lr = LinearRegression()
```

In [19]:
```
def model_evaluation(model, name=''):
    model.fit(X_train_scaled,  y_train)
    # Performance Evaluation
    print(f'The r2 score for {name} model on train set is : ', model.score(X_train_s
    print(f'The r2 score for {name} model on test set is : ', model.score(X_test_sca

    # Coefficients
    df_lr = pd.DataFrame()
    df_lr['features']  =  X_train_scaled.columns
    df_lr['coef'] = model.coef_

    # Selecting top 5 features
    print('\n Top 5 imp features : \n')
    print(df_lr.reindex(df_lr['coef'].abs().sort_values(ascending=False).index)[:5])
```

In [20]:
```
# Plain Linear Regression Model
model_evaluation(lr,  'Linear  Regression')
```

```
The r2 score for Linear Regression model on train set is :  0.9112600781960993
The r2 score for Linear Regression model on test set is :  0.9109574246452464

 Top 5 imp features :

          features          coef
```

```
29    class_Economy -20860.914829
18        stops_zero  -2483.705554
6   airline_Vistara   1991.513369
1          days_left  -1755.447654
4    airline_Indigo    761.698062
```

The r2 score for both the train and test set are quite close, so there is no overfitting. Also, the r2 scores are around 91%, which good. This means that 91% variance in the flight price can be attributed to these features.

The economy class, affect the price of the flight most, which is obvious if the class is business,then price is high else low. So, the coefficient of the `class_Economy` is negative.

In [21]:

```python
# L1 regularisation
from sklearn import linear_model
lasso = linear_model.Lasso()
model_evaluation(lasso, 'L1 Regularisation')
```

```
The r2 score for L1 Regularisation model on train set is :  0.9112599093087402
The r2 score for L1 Regularisation model on test set is :  0.9109576842610083

 Top 5 imp features :

           features          coef
29    class_Economy -20859.690519
18        stops_zero  -2482.541544
6   airline_Vistara   1982.806300
1          days_left  -1754.548742
4    airline_Indigo    753.367476
```

Very similar results as simple linear regression model

In [22]:

```python
# L2 regularisation
ridge = linear_model.Ridge()
model_evaluation(ridge, 'L2 Regularisation')
```

```
The r2 score for L2 Regularisation model on train set is :  0.9112600781748711
The r2 score for L2 Regularisation model on test set is :  0.9109574187366954

 Top 5 imp features :

           features          coef
29    class_Economy -20860.802298
18        stops_zero  -2483.688686
6   airline_Vistara   1991.502329
1          days_left  -1755.440903
4    airline_Indigo    761.648690
```

The values of the coefficients of L2 regularization and the performance are also very similar to the simple linear regression

In [23]:

```python
# Elastic Net
elastic_net = linear_model.ElasticNet()
model_evaluation(elastic_net, 'Elastic Net')
```

```
The r2 score for Elastic Net model on train set is :  0.815589793236619
The r2 score for Elastic Net model on test set is :  0.8150818416431319

 Top 5 imp features :

           features          coef
```

```
29    class_Economy -13189.294471
6   airline_Vistara   2275.336262
18       stops_zero  -1584.694982
1         days_left  -1211.572029
4    airline_Indigo   -972.773809
```

The performance of the Elastic Net model is lower than the others also in the coefficients, `airline_Vistara` is more imp than `stops_zero`, which opposite for other models.

Support Vector Regression

In [24]:

```
from sklearn.svm import LinearSVR
```

In [25]:

```
# L1 Regularisation
svr_l1 = LinearSVR(fit_intercept='epsilon_insensitive')
model_evaluation(svr_l1, 'Support  Vector  L1  Regression')
```

```
The r2 score for Support Vector L1 Regression model on train set is :  0.89932102512
5888
The r2 score for Support Vector L1 Regression model on test set is :  0.898920144690
6759

 Top 5 imp features : 

             features          coef
29      class_Economy -20222.410958
6     airline_Vistara   1719.068602
1           days_left  -1486.109106
2   airline_Air_India    786.327928
18         stops_zero   -767.111310
```

The SVR L1 Regularisation is showing small drop in the performance compared to Linear Regression. The top coefficient remains the same, but there are some changes for the remaining positions.

In [26]:

```
# L2 Regularisation
svr_l2  =  LinearSVR(fit_intercept='squared_epsilon_insensitive')
model_evaluation(svr_l2, 'Support Vector L2 Regression')
```

```
The r2 score for Support Vector L2 Regression model on train set is :  0.89927631870
03587
The r2 score for Support Vector L2 Regression model on test set is :  0.898872659985
9933

 Top 5 imp features : 

             features          coef
29      class_Economy -20217.602012
6     airline_Vistara   1717.833836
1           days_left  -1487.000938
2   airline_Air_India    786.211092
18         stops_zero   -762.857706
```

The SVR L2 Regularisation is giving same performance that of L1 regularisation. Also, the top 5imp features of these two are also same, though there is very small change in the coefficient values.

Random Forest Regression

In [27]:

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import GridSearchCV
```

In [28]:

```
forest = RandomForestRegressor(n_estimators=200, oob_score=True)
parameters = {'max_depth': [3, 5, 7, 9],
              'min_samples_leaf' : [100, 500, 1000, 2000],
              'min_samples_split' : [100, 500, 1000, 2000]}

clf = GridSearchCV(forest, param_grid=parameters, cv=5)
clf.fit(X_train_scaled, y_train)
```

Out [28]:

```
▾ RandomForestRegressor
RandomForestRegressor(n_estimators=200, oob_score=True)
```

In [29]:

```
print(f'The r2 score for Random Forest model on train set is : ', clf.score(X_train_
print(f'The r2 score for Random Forest model on test set is : ', clf.score(X_test_sc
```

```
The r2 score for Random Forest model on train set is :  0.9974844675546009
The r2 score for Random Forest model on test set is :  0.985262807682884
```

The performance of Random Forest model is better than all previous models. The r2 score is 99% which is very good.

In [30]:

```
# Coefficients
df_forest = pd.DataFrame()
df_forest['features'] = X_train_scaled.columns
df_forest['coef'] = clf.feature_importances_

# Selecting top 5 features
print('\n Top 5 imp features : \n')
print(df_forest.reindex(df_forest['coef'].abs().sort_values(ascending=False).index)[
```

```
 Top 5 imp features : 

             features      coef
29      class_Economy  0.879713
0            duration  0.059725
1           days_left  0.017995
2   airline_Air_India  0.005525
6     airline_Vistara  0.004512
```

The top imp feature is same in all the models. The top 5 imp features in Random Forest are like previous models, with change of their importance.

Individual Contributions

Since we were only 2 people in the group we managed to decide and take necessary actions on deciding and completing the assignments. There were many changes from the initial phase to the end of the assignment. However, we strongly believe that both of us contributed and learned little new things equally. Both took help from multiple of you tube videos on completing the assignment. Also, I have given a brief on the topics we covered.

Antony Vishal Rajendra Prasad
- Decided the topic Flight Price Prediction
- EDA
- Processed the outliners with Box plot
- Linear Regression
- SVM
- RF Regression
- Gave idea to insert image of code from the python Application for better step by step understanding.
- Report Formatting

Madan Kumar GovindaRaj
- Helped in selecting and defining dataset.
- EDA
- Helped with Identifying the irrelevant data
- ML pre-processing
- Linear Regression
- SVM
- RF Regression
- Report