FLIP ROBO

FLIGHT PRICE PREDICTION PROJECT

SUBMITTED BY:
ATISH KALANGUTKAR

ACKNOWLEDGMENT

I would like to express my deepest gratitude to my SME (Subject Matter Expert) Shwetank Mishra as well as Flip Robo Technologies who gave me the opportunity to do this project on Housing Price Prediction, which also helped me in doing lots of research wherein I came to know about so many new things.

References:

I have also used few external resources that helped me to complete this project successfully. I ensured that I learn from the samples and modify things according to my project requirement. All the external resources that were used in creating this project are listed below:

1. https://www.google.com/
2. https://scikit-learn.org/stable/index.html
3. https://github.com/
4. https://www.analyticsvidhya.com/
5. www.researchgate.net/

INTRODUCTION

Business Problem Framing:

The tourism industry is changing fast and this is attracting a lot more travelers each year. Airline industry is one of the most sophisticated industry in using complex pricing strategies to maximize revenue, based on proprietary algorithms and hidden variables. That is why the airline companies use complex algorithms to calculate the flight ticket prices. There are several different factors on which the price of the flight ticket depends. The seller has information about all the factors, but buyers are able to access limited information only, which is not enough to predict the airfare prices.

Now a days flight prices are quite unpredictable. The ticket prices change frequently. Customers seeking to get the lowest price for their ticket, while airline companies are trying to keep their overall revenue as high as possible. That's why we will try to use machine learning models to solve this problem. This can help airlines by predicting what prices they can maintain and also it will help customers to predict future flight prices and plan their journey accordingly.

To solve this problem, we have scrapped the prices of flight tickets for various airlines between 3rd February to 7th February for Goa to Delhi states, using which we aim to build a model which predicts the prices of the flights using various input features.

Conceptual Background of the Domain Problem:

Flight tickets prices is a thing which is very hard to guess, as today we might see a price of a particular flight and if we again went to check out the flight ticket prices of the same flight tomorrow than it will be different story this is because the airlines uses sophisticated quasi-academics tactics known as ‘revenue management’ or ‘yeild management’. So anyone who has booked ticket knows how unexpectedly the prices vary. The cheapest available ticket on a given flight gets more and less expensive over time. This usually happens as an attempt to maximize revenue based on:

- Time of purchase patterns(making sure last minute purchases are expensive).
- Keeping the flight as full as they want it(raising prices on a flight which is filling up in order to reduce sales and hold inventory for those expensive last minute expensive purchases).

So here here we are trying to help the customers or buyers to understand the price of the flight tickets by deploying machine learning models. This models would help the sellers and buyers to understand the flight ticket prices in market and accordingly they will be able to book their tickets. This would be a practical implementation of a data analysis, statistics and machine learning techniques to solve a daily problem faces by travelers.

Review of literature:

Literature review covers relevant literature with the aim of gaining insight into the factors that are important to predict the flight ticket prices in the market.In this we have discussed about various applications and methods which inspired us to build our supervised Machine Learning techniques to predict the price of flight tickets. We did a background survey regarding the basic ideas of our project and used those ideas for the collection of data information by doing web scaping from www.kayak.co.in website which is a web platform where customers can book their flight tickets.

This project is more about data exploration, feature engineering and pre processing that can be done on this data. As we have scrapped the data that includes flight related features, we can do better data explorations and derive some interesting features using the available columns.

Different techniques line ensemble techniques have been used to make the predictions.

Motivation for the Problem Undertaken:

Travelling by flight or air travel is the fast medium of transport all around the world which can cut hours of travelling. But we know how unexpectedly the prices vary. So, I was interested in flight price prediction to help the customers to find the right fares based on their needs.

ANALYTICAL PROBLEM FRAMING

Mathematical/ Analytical Modeling of the Problem:

We nee d to develop an efficient and effective machine learning model which predicts the price of flight tickets. So 'Price' is our target variable which is continuous in nature. So it is a Regression problem where we need to use regression algorithms to predict the results.This project is done on three phases:

Data Collection Phase: I have done web scraping to collect the data of flights from the website www.kayak.co.in . As per the requirement we need to build the model to predict the prices of flight tickets.

Data Analysis: After cleaning the data I have done some analysis on the data by using different types visualizations.

Model Building Phase: After collecting the data, I built a machine learning model. Before that, have done all data per-processing steps. The complete life cycle of data science that I have used in this project are as follows:

- Data Cleaning
- Exploratory Data Analysis
- Data Pre-Processing
- Model Building
- Model Evaluation
- Selecting the best model

Data Sources and their formats:

We have collected the dataset from the website www.kayak.co.in. The data is scrapped using web scraping technique and the framework used is selenium. We have scraped about 1558 rows and 10 columns including target variable 'Price'. The dataset contains both categorical and numerical data type. The data description is as follows:

Features

Date: The date of the journey

Airline Name: The name of the airline.

Source: The source from which the service begins.

Departure From: The place or airport from where journey begins.

Destination: The place or airport where the journey ends.

Departure Time: The time when the journey starts from the airport.

Arrival Time: Time of arrival at the airport.

Duration: Total duration of the flight.

Total_Stops: Total stops between the departure from and destination.

Target

Price: The price of the ticket.

Data Pre-Processing Done:

Data pre-processing is the process of converting raw data into a well readable format to be used by machine learning model. I have used following pre-processing steps:

- ◆ Importing required libraries and loading the dataset.
- ◆ Checked some statistical information such as shape of the dataset,unique values and number of unique values present, type of data present in the columns,value counts, checked for duplicates etc.
- ◆ Checked for null values but there were no null values present in the dataset.
- ◆ Taking care of timestamp variables by converting data types of Date, Departure Time and Arrival Time from object data type into date time data types.
- ◆ Done feature engineering on target variable that is ‘Price’ as it was having commas in between the numbers and because of that it was showing data type as object. So replaced them by empty spaces and converted the data type to integer.

- The column duration was having values I terms of hours and minutes, so splitted the data into hours and minutes appended the data into new columns departure hours and departure minutes which was added to the dataset. And than ‘h’ and ‘m’ from columns departure hour and departure minute was replaced by empty spaces. After that their data type was converted from object to integer data type.
- Than separated categorical and numeric features according to the data types.
- Performed uni-variate, bi-variate and multi-variate analysis by plotting distribution plot, count plot, pie plot and bar plot.
- Than checked for the most expensive prices data, cheapest prices data and longest duration data.
- Encoded the categorical features by using Label encoded method.
- Checked whether there are any outliers present or not using box plot on numerical data columns and found that duration hours was having outliers so by using z-score method removed outliers.
- Checked for skewness on numerical data columns and found that arrival hour and duration hours column was having skewness, so by using power transform method treated the skewness of that particular columns.
- Used bar plot to check the correlation between the features and the label and after with the help of heatmap checked whether there is multicollinearity present or not among the features. So found that column departure_from and destination are highly multi correlated with each other. So plotting scatter plot to check the trend or relationship between them.

- After plotting scatter plot we saw a positive relationship them, but to cross verify, i had used vif method, in that their values was showing infinity, so dropped both the columns and also month column was showing NaN value, so dropped that column also.
- Separated feature and label data and than feature scaling was done by using power transform method to avoid any kind of data biasness.
- After that feature selection method was used to select the best features for the prediction.

Data Inputs-Logic-Output Relationships:

The dataset consists of label and features. The features are independent and label is dependent.

I have checked the correlation between the label and the features using heatmap and bar plot in which both positive and negative correlation between the label and features were there.

After that different data mining model were designed to predict the prices of the flight tickets. In this 4 regression models were used namely, XGB, Gradient Boosting, Ada Boost and Random forest.

Hardware and Software Requirements and Tools Used:

To build a machine learning projects it is important to have the followif hardware and software requirements tools.

Hardware Required:

- RAM: 8GB
- CPU: AMD RYZEN 5 4600H

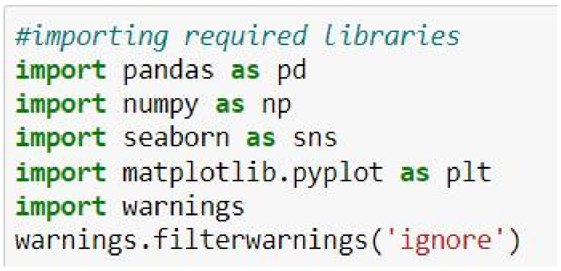
- GPU: AMD Radeon ™ Vega 8 Graphics and NVIDIA GeForce GTX 1650 Ti

Software Required:

- Distribution: Anaconda Navigator
- Programming Language: Python
- Browser based language shell: Jupyter Notebook
- Chrome: To scrap the data

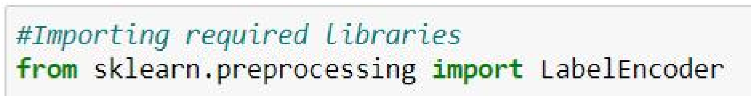
Libraries Required:

- Pre-processing and Visualization



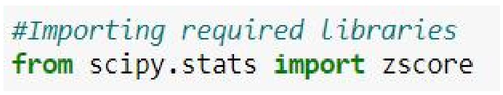
```
#importing required libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

- Encoding



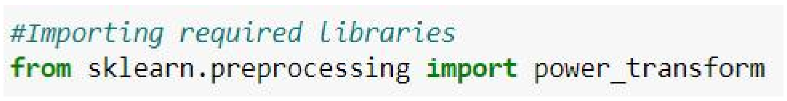
```
#Importing required libraries
from sklearn.preprocessing import LabelEncoder
```

- To remove outliers



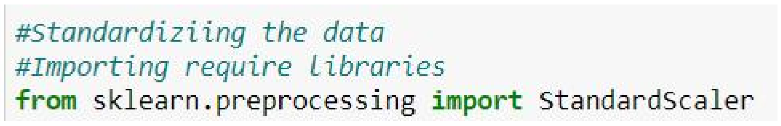
```
#Importing required libraries
from scipy.stats import zscore
```

- To remove skewness



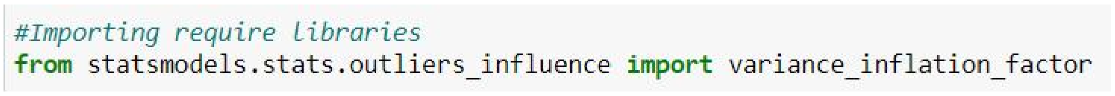
```
#Importing required libraries
from sklearn.preprocessing import power_transform
```

- To standardize the data



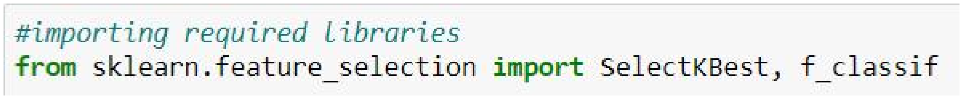
```
#Standardiziing the data
#Importing require libraries
from sklearn.preprocessing import StandardScaler
```

- Vif method



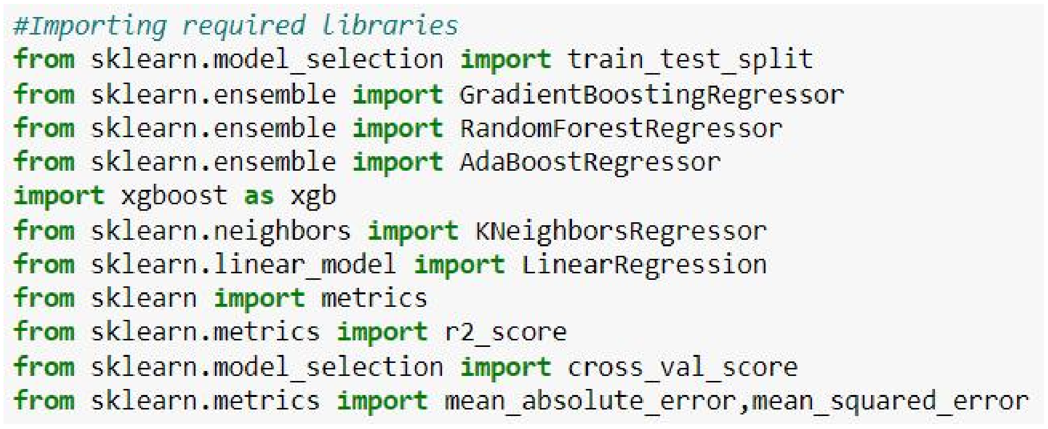
```
#Importing require libraries
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

- Feature Selection



```
#importing required libraries
from sklearn.feature_selection import SelectKBest, f_classif
```

- Evaluation metrics and machine learning Algorithms



```
#Importing required libraries
from sklearn.model_selection import train_test_split
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import AdaBoostRegressor
import xgboost as xgb
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import LinearRegression
from sklearn import metrics
from sklearn.metrics import r2_score
from sklearn.model_selection import cross_val_score
from sklearn.metrics import mean_absolute_error,mean_squared_error
```

MODEL/S DEVELOPMENT AND EVALUATION

Identification of Possible Problem-solving Approaches (Methods):

I have used both statistical and analytical approaches to solve the problem which mainly includes the pre-processing of the data also used EDA techniques and heat map to check the correlation of features and label.

Encoded data using label encoder, removed outliers and and skewness using z-score method and power transform method. Scaled the data using standard scaler and used feature selection method so that best features should be used for the prediction. Also, before building the model, I made sure that input data is cleaned and scaled.

Than checked for the best random state to be used on our regression machine learning model pertaining to the feature importance details.And than created multiple regression models along with evaluation metrics. In this data 'Price' is our target variable which is continuous in nature so this is a regression problem. So I have used 4 regression algorithms and predicted the flight ticket prices.

By going through each and very aspect of the machine learning models I have selected Xtreme Gradient Boosting(XGB) as best suitable algorithm to create our final model as it is giving highest testing score(R2 SCORE) and less root mean squared error(RMSE) among all the algorithms used. Performed hyper parameter tuning on best model and at last saved the final model.

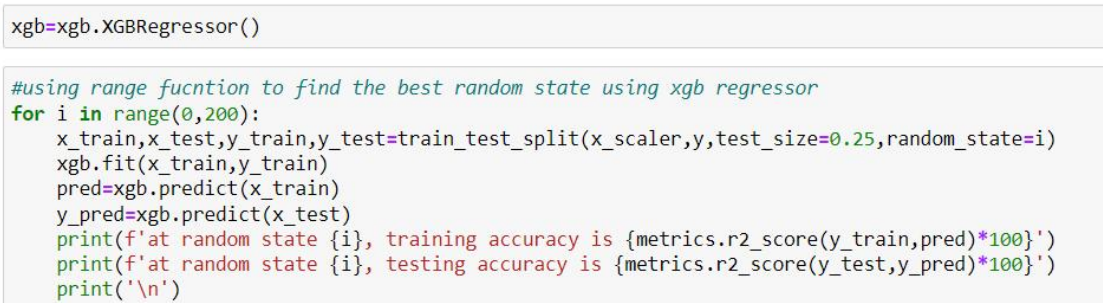
Testing of Identified Approaches(Algorithms):

Since 'Price' is my target variable which is continuous in nature, so that we came to know that it is regression type problem and I have used following regression models, they are:

1. Xtreme Gradient Boosting Regressor (XGB)
2. Gradient Boosting Regressor
3. Ada Boost Regressor
4. Random Forest Regressor

Run and Evaluate Selected Models:

I have used 4 regression algorithms after choosing random state amongst 0-200 number. I have used XGB Regressor to find the best random forest state and code is as below:



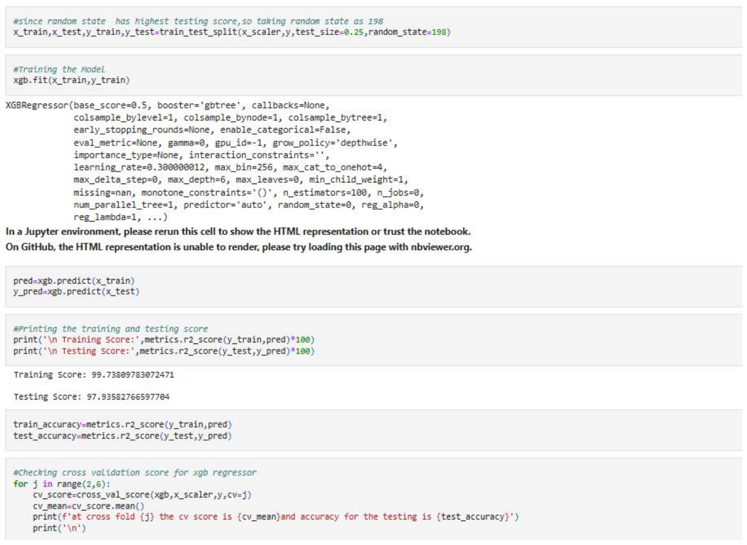
```
xgb=xgb.XGBRegressor()
```

```
#using range fucntion to find the best random state using xgb regressor
for i in range(0,200):
    x_train,x_test,y_train,y_test=train_test_split(x_scaler,y,test_size=0.25,random_state=i)
    xgb.fit(x_train,y_train)
    pred=xgb.predict(x_train)
    y_pred=xgb.predict(x_test)
    print(f'at random state {i}, training accuracy is {metrics.r2_score(y_train,pred)*100}')
    print(f'at random state {i}, testing accuracy is {metrics.r2_score(y_test,y_pred)*100}')
    print('\n')
```

Maximum testing score (R2 score) is 97.93 on random state 198.

Model Building:

Xtreme Gradient Boosting



```
#since random state  has highest testing score,so taking random state as 198
x_train,x_test,y_train,y_test=train_test_split(x_scaler,y,test_size=0.25,random_state=198)
```

```
#Training the Model
xgb.fit(x_train,y_train)
```

```
XGBRegressor(base_score=0.5, booster='gbtree', callbacks=None,
             colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
             early_stopping_rounds=None, enable_categorical=False,
             eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthwise',
             importance_type=None, interaction_constraints='',
             learning_rate=0.300000012, max_bin=256, max_cat_to_onehot=4,
             max_delta_step=0, max_depth=6, max_leaves=0, min_child_weight=1,
             missing=nan, monotone_constraints='()', n_estimators=100, n_jobs=0,
             num_parallel_tree=1, predictor='auto', random_state=0, reg_alpha=0,
             reg_lambda=1, ...)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
pred=xgb.predict(x_train)
y_pred=xgb.predict(x_test)
```

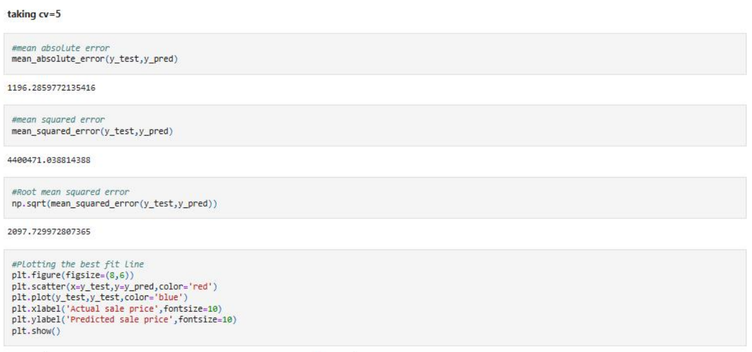
```
#Printing the training and testing score
print('\n Training Score:',metrics.r2_score(y_train,pred)*100)
print('\n Testing Score:',metrics.r2_score(y_test,y_pred)*100)
```

```
 Training Score: 99.73809783072471

 Testing Score: 97.93582766597704
```

```
train_accuracy=metrics.r2_score(y_train,pred)
test_accuracy=metrics.r2_score(y_test,y_pred)
```

```
#Checking cross validation score for xgb regressor
for j in range(2,6):
    cv_score=cross_val_score(xgb,x_scaler,y,cv=j)
    cv_mean=cv_score.mean()
    print(f'at cross fold {j} the cv score is {cv_mean}and accuracy for the testing is {test_accuracy}')
    print('\n')
```



taking cv=5

```
#mean absolute error
mean_absolute_error(y_test,y_pred)
```

```
1196.2859772135416
```

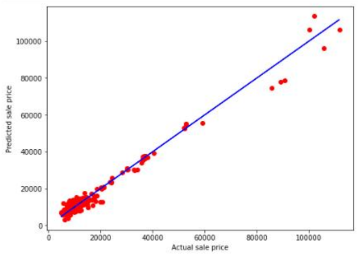
```
#mean squared error
mean_squared_error(y_test,y_pred)
```

```
4400471.038814388
```

```
#Root mean squared error
np.sqrt(mean_squared_error(y_test,y_pred))
```

```
2097.729972807365
```

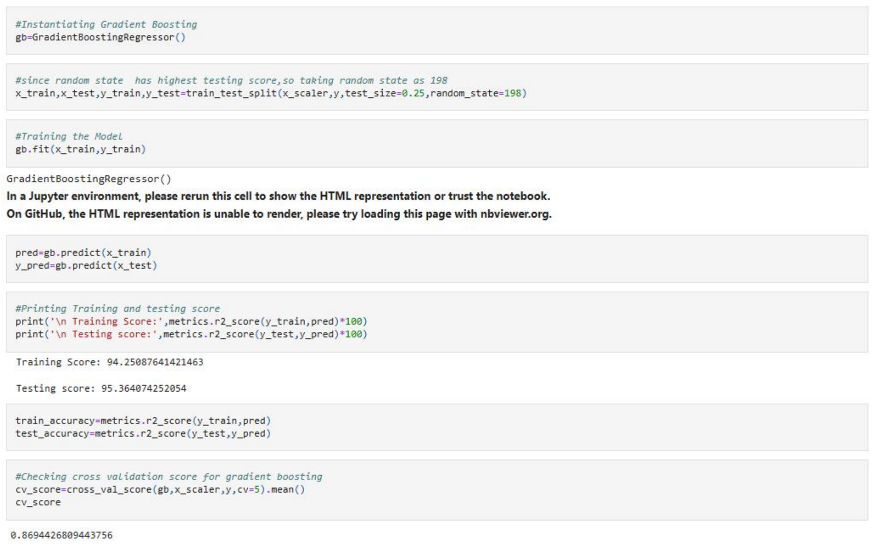
```
#Plotting the best fit line
plt.figure(figsize=(8,6))
plt.scatter(x=y_test,y=y_pred,color='red')
plt.plot(y_test,y_test,color='blue')
plt.xlabel('Actual sale price',fontsize=10)
plt.ylabel('Predicted sale price',fontsize=10)
plt.show()
```



Created XGB Regressor model and checked for its evaluation metrics. The model is giving testing score as 97.93%.

From the graph we can observe how our model is mapping and also we can observe the straight line which is our actual dataset and dots are the predictions that the model has given.

Gradient Boosting



```
#Instantiating Gradient Boosting
gb=GradientBoostingRegressor()
```

```
#since random state  has highest testing score,so taking random state as 198
x_train,x_test,y_train,y_test=train_test_split(x_scaler,y,test_size=0.25,random_state=198)
```

```
#Training the Model
gb.fit(x_train,y_train)
```

```
GradientBoostingRegressor()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
pred=gb.predict(x_train)
y_pred=gb.predict(x_test)
```

```
#Printing Training and testing score
print('\n Training Score:',metrics.r2_score(y_train,pred)*100)
print('\n Testing score:',metrics.r2_score(y_test,y_pred)*100)
```

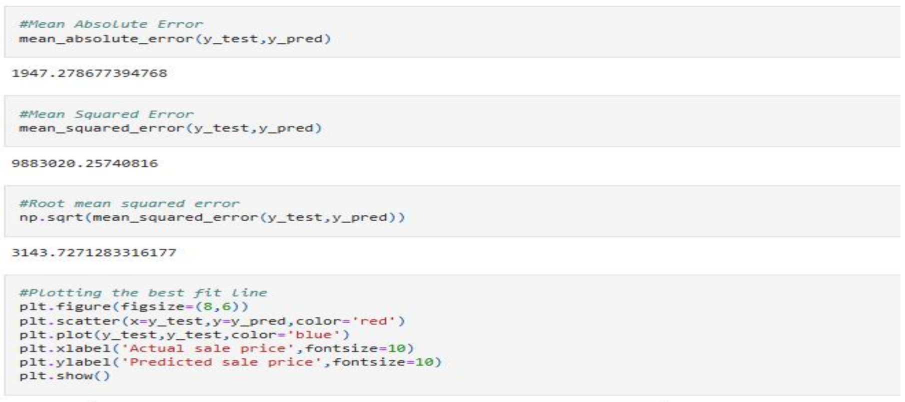
```
 Training Score: 94.25087641421463

 Testing score: 95.364074252054
```

```
train_accuracy=metrics.r2_score(y_train,pred)
test_accuracy=metrics.r2_score(y_test,y_pred)
```

```
#Checking cross validation score for gradient boosting
cv_score=cross_val_score(gb,x_scaler,y,cv=5).mean()
cv_score
```

```
0.8694426809443756
```



```
#Mean Absolute Error
mean_absolute_error(y_test,y_pred)
```

```
1947.278677394768
```

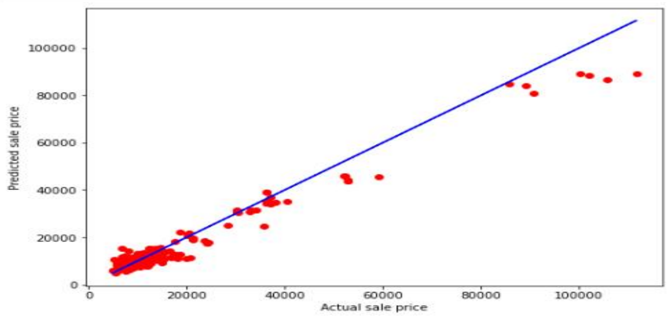
```
#Mean Squared Error
mean_squared_error(y_test,y_pred)
```

```
9883020.25740816
```

```
#Root mean squared error
np.sqrt(mean_squared_error(y_test,y_pred))
```

```
3143.7271283316177
```

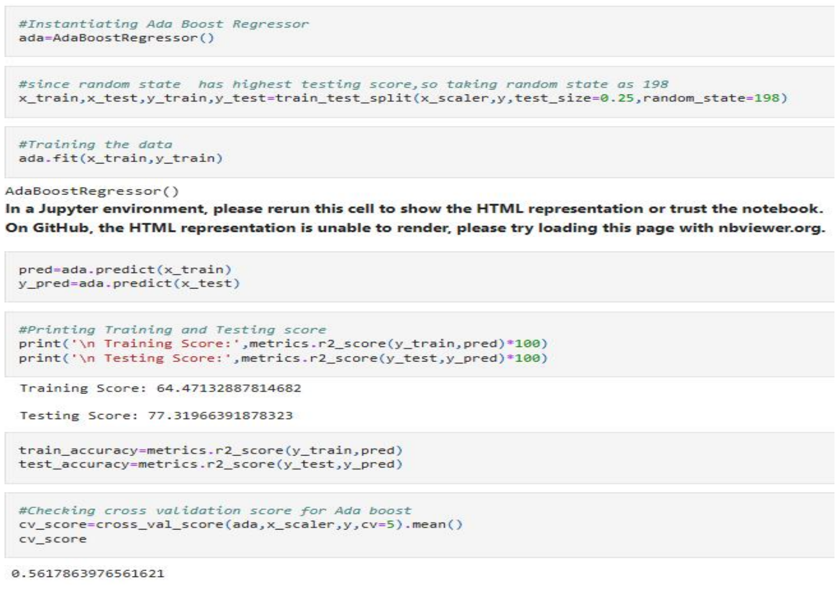
```
#Plotting the best fit line
plt.figure(figsize=(8,6))
plt.scatter(x=y_test,y=y_pred,color='red')
plt.plot(y_test,y_test,color='blue')
plt.xlabel('Actual sale price',fontsize=10)
plt.ylabel('Predicted sale price',fontsize=10)
plt.show()
```



Created Gradient Boosting Regressor model and checked for its evaluation metrics. The model is giving testing score as 95.36%.

From the graph we can observe how our model is mapping and also we can observe the straight line which is our actual dataset and dots are the predictions that the model has given.

Ada Boost



```
#Instantiating Ada Boost Regressor
ada=AdaBoostRegressor()
```

```
#since random state  has highest testing score,so taking random state as 198
x_train,x_test,y_train,y_test=train_test_split(x_scaler,y,test_size=0.25,random_state=198)
```

```
#Training the data
ada.fit(x_train,y_train)
```

```
AdaBoostRegressor()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
pred=ada.predict(x_train)
y_pred=ada.predict(x_test)
```

```
#Printing Training and Testing score
print('\n Training Score:',metrics.r2_score(y_train,pred)*100)
print('\n Testing Score:',metrics.r2_score(y_test,y_pred)*100)
```

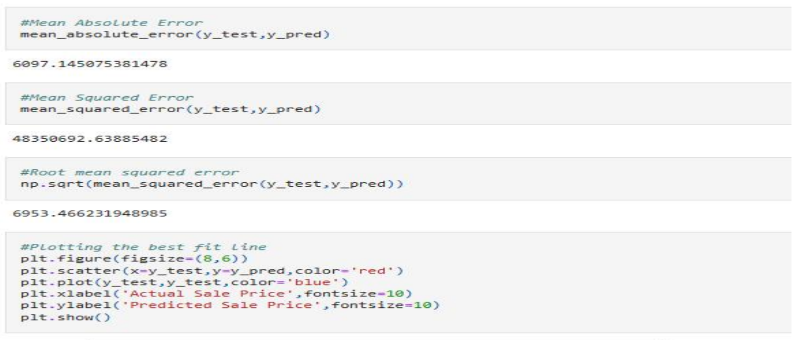
```
 Training Score: 64.47132887814682

 Testing Score: 77.31966391878323
```

```
train_accuracy=metrics.r2_score(y_train,pred)
test_accuracy=metrics.r2_score(y_test,y_pred)
```

```
#Checking cross validation score for Ada boost
cv_score=cross_val_score(ada,x_scaler,y,cv=5).mean()
cv_score
```

```
0.5617863976561621
```



```
#Mean Absolute Error
mean_absolute_error(y_test,y_pred)
```

```
6097.145075381478
```

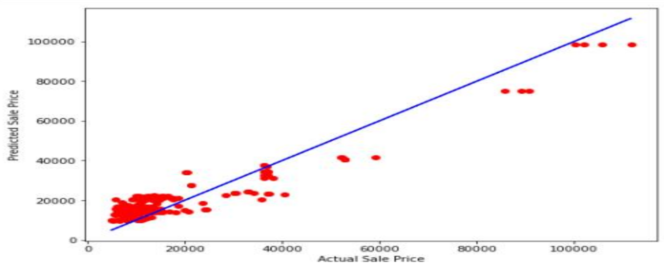
```
#Mean Squared Error
mean_squared_error(y_test,y_pred)
```

```
48350692.63885482
```

```
#Root mean squared error
np.sqrt(mean_squared_error(y_test,y_pred))
```

```
6953.466231948985
```

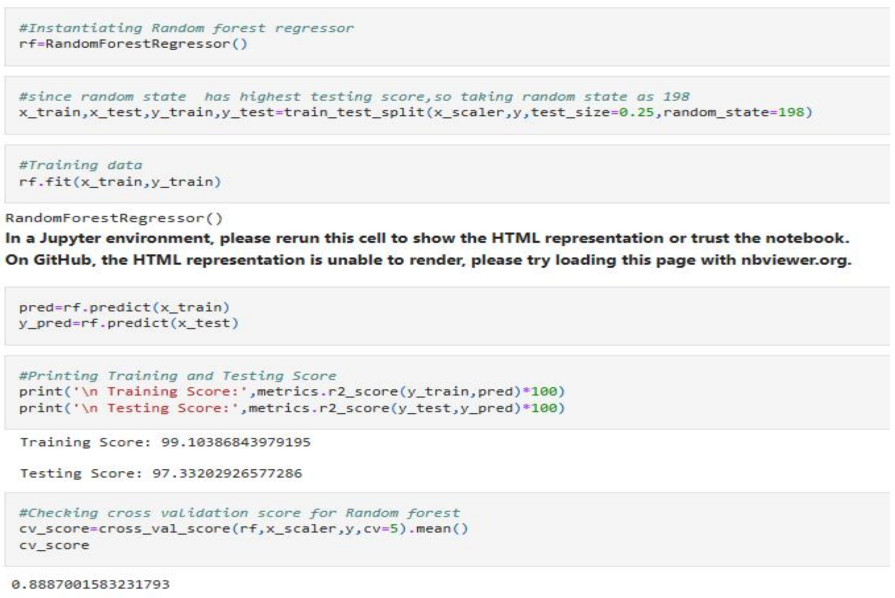
```
#Plotting the best fit line
plt.figure(figsize=(8,6))
plt.scatter(x=y_test,y=y_pred,color='red')
plt.plot(y_test,y_test,color='blue')
plt.xlabel('Actual Sale Price',fontsize=10)
plt.ylabel('Predicted Sale Price',fontsize=10)
plt.show()
```



Created Ada Boost Regressor model and checked for its evaluation metrics. The model is giving testing score as 77.31%.

From the graph we can observe how our model is mapping and also we can observe the straight line which is our actual dataset and dots are the predictions that the model has given.

Random Forest



```
#Instantiating Random forest regressor
rf=RandomForestRegressor()
```

```
#since random state  has highest testing score,so taking random state as 198
x_train,x_test,y_train,y_test=train_test_split(x_scaler,y,test_size=0.25,random_state=198)
```

```
#Training data
rf.fit(x_train,y_train)
```

```
RandomForestRegressor()
```

**In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.**

```
pred=rf.predict(x_train)
y_pred=rf.predict(x_test)
```

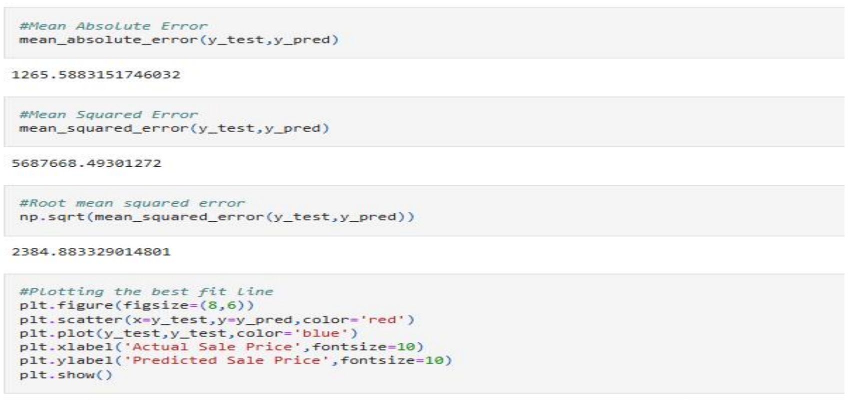
```
#Printing Training and Testing Score
print('\n Training Score:',metrics.r2_score(y_train,pred)*100)
print('\n Testing Score:',metrics.r2_score(y_test,y_pred)*100)
```

```
 Training Score: 99.10386843979195

 Testing Score: 97.33202926577286
```

```
#Checking cross validation score for Random forest
cv_score=cross_val_score(rf,x_scaler,y,cv=5).mean()
cv_score
```

```
0.8887001583231793
```



```
#Mean Absolute Error
mean_absolute_error(y_test,y_pred)
```

```
1265.5883151746032
```

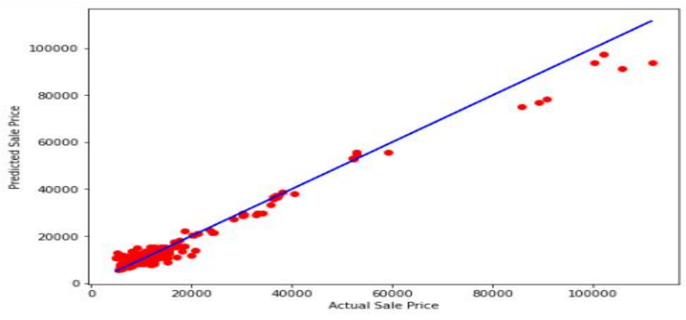
```
#Mean Squared Error
mean_squared_error(y_test,y_pred)
```

```
5687668.49301272
```

```
#Root mean squared error
np.sqrt(mean_squared_error(y_test,y_pred))
```

```
2384.883329014801
```

```
#Plotting the best fit line
plt.figure(figsize=(8,6))
plt.scatter(x=y_test,y=y_pred,color='red')
plt.plot(y_test,y_test,color='blue')
plt.xlabel('Actual Sale Price',fontsize=10)
plt.ylabel('Predicted Sale Price',fontsize=10)
plt.show()
```



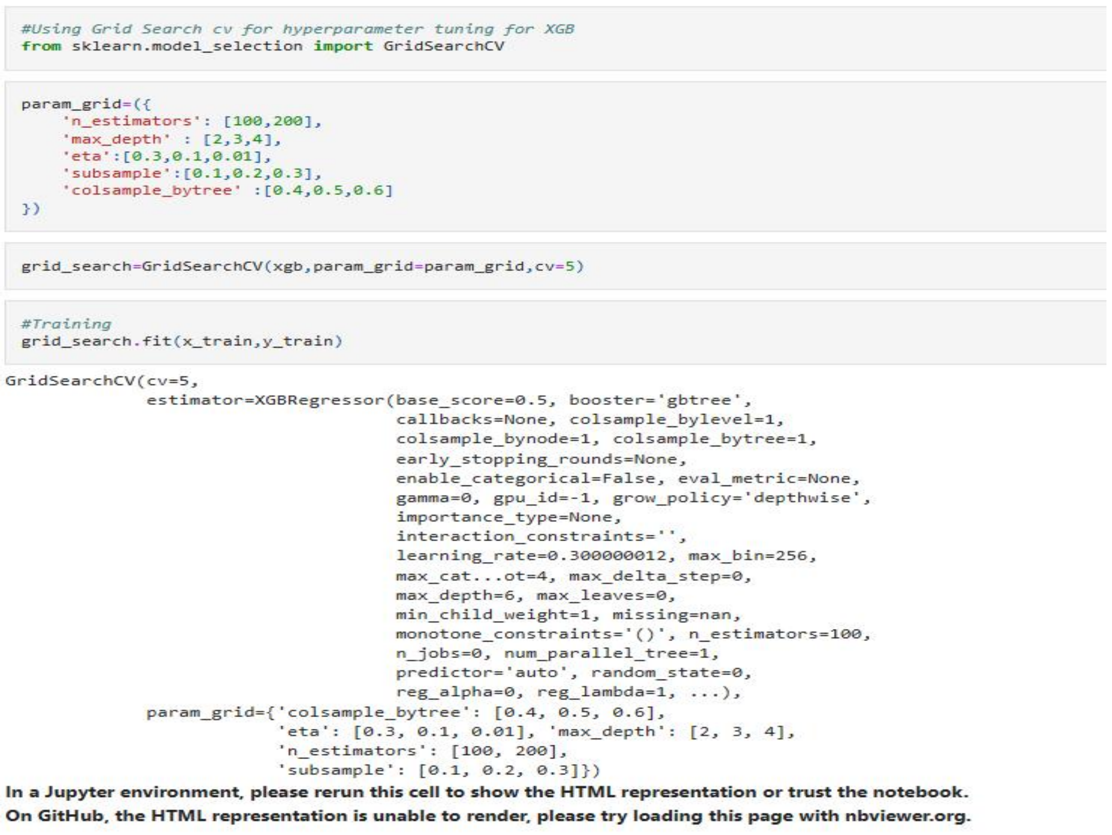
Created Ada Boost Regressor model and checked for its evaluation metrics. The model is giving testing score as 97.33%.

From the graph we can observe how our model is mapping and also we can observe the straight line which is our actual dataset and dots are the predictions that the model has given.

Model Selection:

From the above created models, we can conclude that ‘Xtreme Gradient Boosting Regressor’ as the best fitting model as it is giving high testing score(R2 SCORE) and low MAE, MSE and RMSE values. Then we used hyper parameter tuning to tuned the model and increase our model score.

Hyper Parameter Tuning:



```
#Using Grid Search cv for hyperparameter tuning for XGB
from sklearn.model_selection import GridSearchCV
```

```
param_grid=({
    'n_estimators': [100,200],
    'max_depth' : [2,3,4],
    'eta':[0.3,0.1,0.01],
    'subsample':[0.1,0.2,0.3],
    'colsample_bytree' :[0.4,0.5,0.6]
})
```

```
grid_search=GridSearchCV(xgb,param_grid=param_grid,cv=5)
```

```
#Training
grid_search.fit(x_train,y_train)
```

```
GridSearchCV(cv=5,
             estimator=XGBRegressor(base_score=0.5, booster='gbtree',
                                    callbacks=None, colsample_bylevel=1,
                                    colsample_bynode=1, colsample_bytree=1,
                                    early_stopping_rounds=None,
                                    enable_categorical=False, eval_metric=None,
                                    gamma=0, gpu_id=-1, grow_policy='depthwise',
                                    importance_type=None,
                                    interaction_constraints='',
                                    learning_rate=0.300000012, max_bin=256,
                                    max_cat...ot=4, max_delta_step=0,
                                    max_depth=6, max_leaves=0,
                                    min_child_weight=1, missing=nan,
                                    monotone_constraints='()', n_estimators=100,
                                    n_jobs=0, num_parallel_tree=1,
                                    predictor='auto', random_state=0,
                                    reg_alpha=0, reg_lambda=1, ...),
             param_grid={'colsample_bytree': [0.4, 0.5, 0.6],
                         'eta': [0.3, 0.1, 0.01], 'max_depth': [2, 3, 4],
                         'n_estimators': [100, 200],
                         'subsample': [0.1, 0.2, 0.3]})
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
#best Parameters
grid_search.best_params_
```

```
{'colsample_bytree': 0.6,
 'eta': 0.3,
 'max_depth': 4,
 'n_estimators': 200,
 'subsample': 0.3}
```

```
#Training with the best parameters
xgb1=xgb.XGBRegressor(colsample_bytree=0.6,eta=0.3,max_depth=4,n_estimators=200,subsample=0.3)
xgb1.fit(x_train,y_train)
```

```
XGBRegressor(base_score=0.5, booster='gbtree', callbacks=None,
             colsample_bylevel=1, colsample_bynode=1, colsample_bytree=0.6,
             early_stopping_rounds=None, enable_categorical=False, eta=0.3,
             eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthwise',
             importance_type=None, interaction_constraints='',
             learning_rate=0.300000012, max_bin=256, max_cat_to_onehot=4,
             max_delta_step=0, max_depth=4, max_leaves=0, min_child_weight=1,
             missing=nan, monotone_constraints='()', n_estimators=200, n_jobs=0,
             num_parallel_tree=1, predictor='auto', random_state=0, reg_alpha=0, ...)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
#Printing the training and testing score
print('\n Training Score:',metrics.r2_score(y_train,pred)*100)
print('\n Testing Score:',metrics.r2_score(y_test,y_pred)*100)
```

```
 Training Score: 99.73809783072471

 Testing Score: 97.93582766597704
```

```
#Checking cross validation score for XGBoost
cv_score=cross_val_score(xgb1,x_scaler,y,cv=5).mean()
cv_score
```

```
0.8870995778788291
```

```
#Mean Absolute Error
mean_absolute_error(y_test,y_pred)
```

```
1196.2859772135416
```

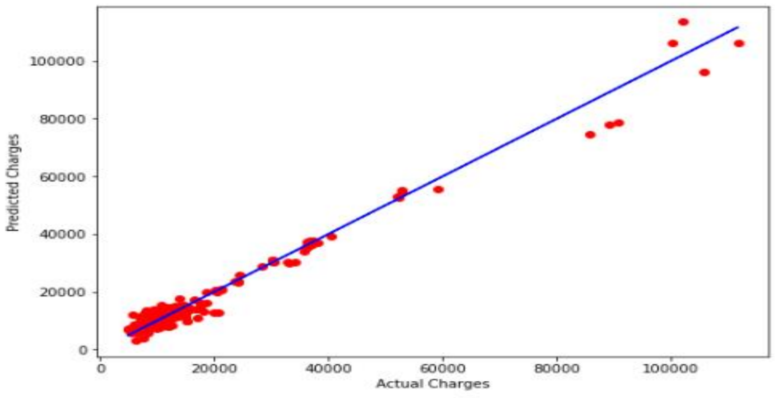
```
#Mean Squared Error
mean_squared_error(y_test,y_pred)
```

```
4400471.038814388
```

```
#Root mean squared error
np.sqrt(mean_squared_error(y_test,y_pred))
```

```
2097.729972807365
```

```
#Plotting the best fit line
plt.figure(figsize=(8,6))
plt.scatter(x=y_test,y=y_pred,color='red')
plt.plot(y_test,y_test,color='blue')
plt.xlabel('Actual Charges',fontsize=10)
plt.ylabel('Predicted Charges',fontsize=10)
plt.show()
```



After performing hyper parameter tuning using best parameters of XGB, the testing scores remained same but the cross validation score increased from 85% to 88% .

From the graph we can observe how our model is mapping and also we can observe the straight line which is our actual dataset and dots are the predictions that the model has given.

Saving the model:

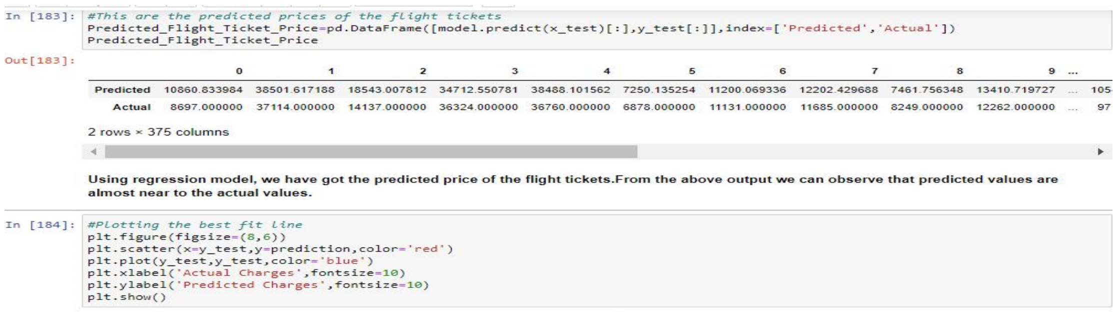
```
#saving the model
#Importing required libraries
import pickle
pickle.dump(xgb1,open('Flight Price Prediction Project','wb'))
```

Loading the saved model and predicting the flight ticket price:

```
#Loading the saved model
model=pickle.load(open('Flight Price Prediction Project','rb'))
```

```
#Prediction
prediction=model.predict(x_test)
prediction
```

```
array([ 10860.834 ,  38501.617 ,  18543.008 ,  34712.55  ,  38488.1   ,
         7250.1353,  11200.069 ,  12202.43  ,   7461.7563,  13410.72  ,
```



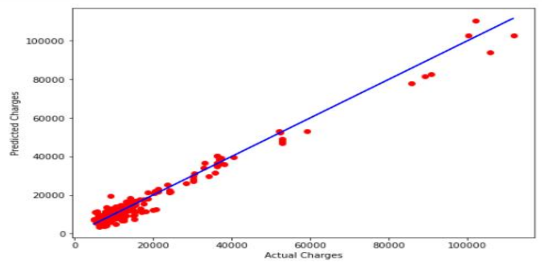
```
In [183]: #This are the predicted prices of the flight tickets
          Predicted_Flight_Ticket_Price=pd.DataFrame([model.predict(x_test)[:],y_test[:]],index=['Predicted','Actual'])
          Predicted_Flight_Ticket_Price
Out[183]:
```

	0	1	2	3	4	5	6	7	8	9	...	
Predicted	10860.833984	38501.617188	18543.007812	34712.550781	38488.101562	7250.135254	11200.069336	12202.429688	7461.756348	13410.719727	...	105
Actual	8697.000000	37114.000000	14137.000000	36324.000000	36760.000000	6878.000000	11131.000000	11685.000000	8249.000000	12262.000000	...	97

2 rows × 375 columns

Using regression model, we have got the predicted price of the flight tickets.From the above output we can observe that predicted values are almost near to the actual values.

```
In [184]: #Plotting the best fit line
          plt.figure(figsize=(8,6))
          plt.scatter(x=y_test,y=prediction,color='red')
          plt.plot(y_test,y_test,color='blue')
          plt.xlabel('Actual Charges',fontsize=10)
          plt.ylabel('Predicted Charges',fontsize=10)
          plt.show()
```



The graph shows how our model is mapping and the plot gives the linear relation between predicted and actual price of the flight tickets. The blue line is the best fitting line which gives us the actual values/data and red dots gives us the predicted values/data.

Key Metrics for success in solving problem under Consideration:

The key metrics used here were r2_score, cross_val_score, MAE, MSE and RMSE. We tried to find out the best parameters and also to increase our scores by using Hyper parameter Tuning and we will be using the GridSearchCV method.

1. Cross Validation: Cross-validation helps to find out the over fitting and under fitting of the model. In the cross validation the model is made to run on different subsets of the dataset which will get multiple measures of the model. If we take 5 folds, the data will be divided into 5 pieces where each part being 20% of the full dataset. While running the Cross-validation the 1st part (20%) of the 5 parts will be kept out as a holdout set for validation and everything else is used for training data. This way we will get the first estimate of the model quality of the dataset.
In the similar way further iterations are made for the second 20% of the dataset is held as a holdout set and remaining 4 parts are used for training data during the process. This way we will get the second estimate of the model quality of the dataset. These steps are

repeated during the cross-validation process to get the remaining estimate of the model quality.

2. **R2 Score:** It is a statistical measure that represents the goodness of fit of a regression model. The ideal value for r-square is 1. The closer the value of r-square to 1, the better is the model fitted.

3. Mean Squared Error (MSE): MSE of an estimator (of a procedure for estimating an unobserved quantity) measures the average of the squares of the errors — that is, the average squared difference between the estimated values and what is estimated. MSE is a risk function, corresponding to the expected value of the squared error loss. RMSE is the Root Mean Squared Error.

4. Mean Absolute Error (MAE): MAE measures the average magnitude of the errors in a set of predictions, without considering their direction. It's the average over the test sample of the absolute differences between prediction and actual observation where all individual differences have equal weight.

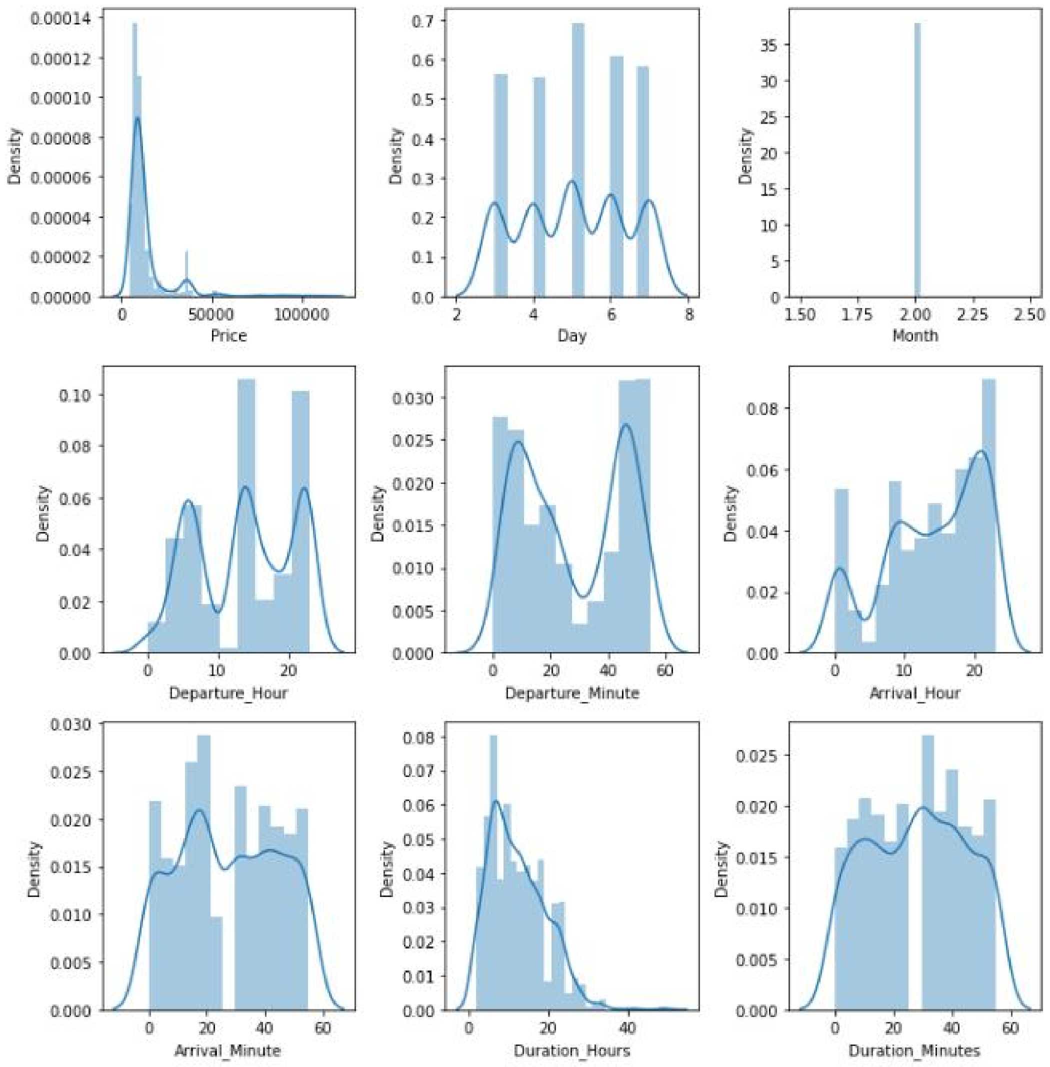
5. Hyper parameter Tuning: There is a list of different machine learning models. They all are different in some way or the other, but what makes them different is nothing but input parameters for the model. These input parameters are named as Hyper parameters. These hyper parameters will define the architecture of the model, and the best part about these is that you get a choice to select

these for your model. You must select from a specific list of hyper parameters for a given model as it varies from model to model.We are not aware of optimal values for hyper parameters which would generate the best model output. So, what we tell the model
is to explore and select the optimal model architecture automatically. This selection procedure for hyper parameters is known as Hyper parameter Tuning. We can do tuning by using GridSearchCV.
GridSearchCV is a function that comes in the Sci kit-learn (or SK-learn) model selection package. An important point here to note is that we need to have the Sci kit-learn library installed on the computer. This function helps to loop through predefined
hyper parameters and fit your estimator (model) on your training set. So, in the end, we can select the best parameters from the listed hyper parameters.

Visualization:

I used pandas profiling to get the over viewed visualization on the pre processed data. Pandas is an open source python module with which we can do an exploratory data analysis to get detailed description of the features and it helps in visualizing and understanding the distribution of each variable.
I have analyzed the data using distribution plot, pie plot,count plot, bar plot and box plots to get the relation between the features and label.

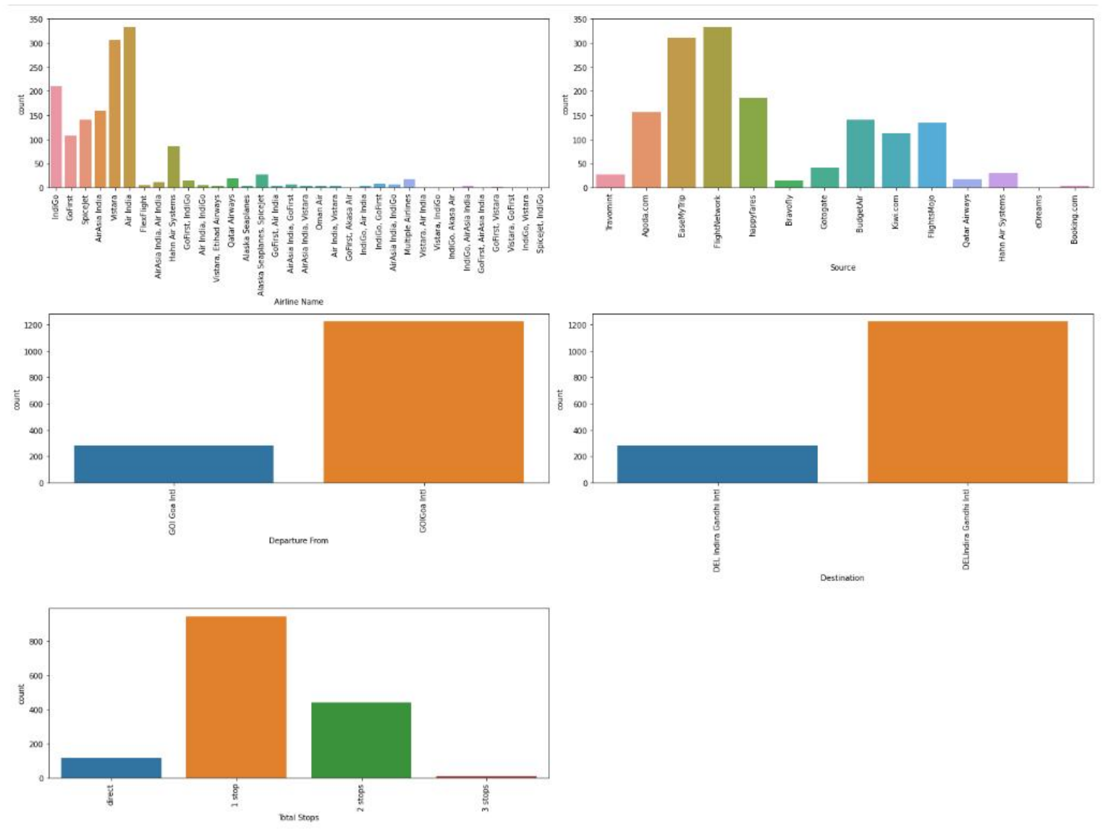
Distribution Plots:



In above plots we can see that:

- In column arrival hour and duration skewness is present and also data is not distributed normally as we don't see a proper bell shaped curve.
- So to cross verify this we will check the skewness and if the values is more than +5 or -5 than by using power transform method we will treat the skewness.

Count Plots:

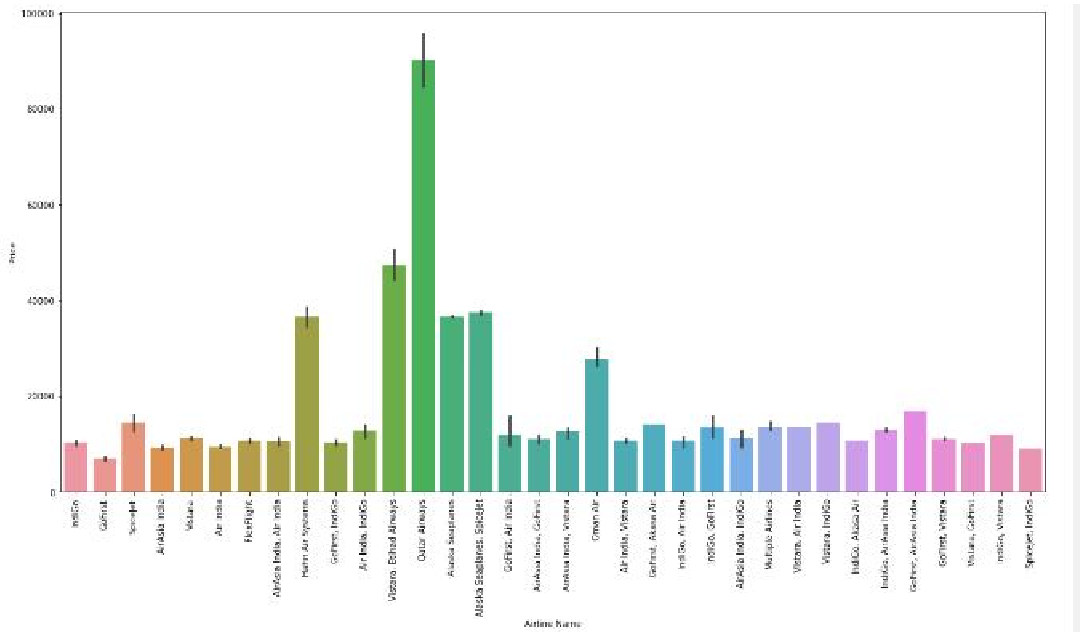


In above plots we can see that:

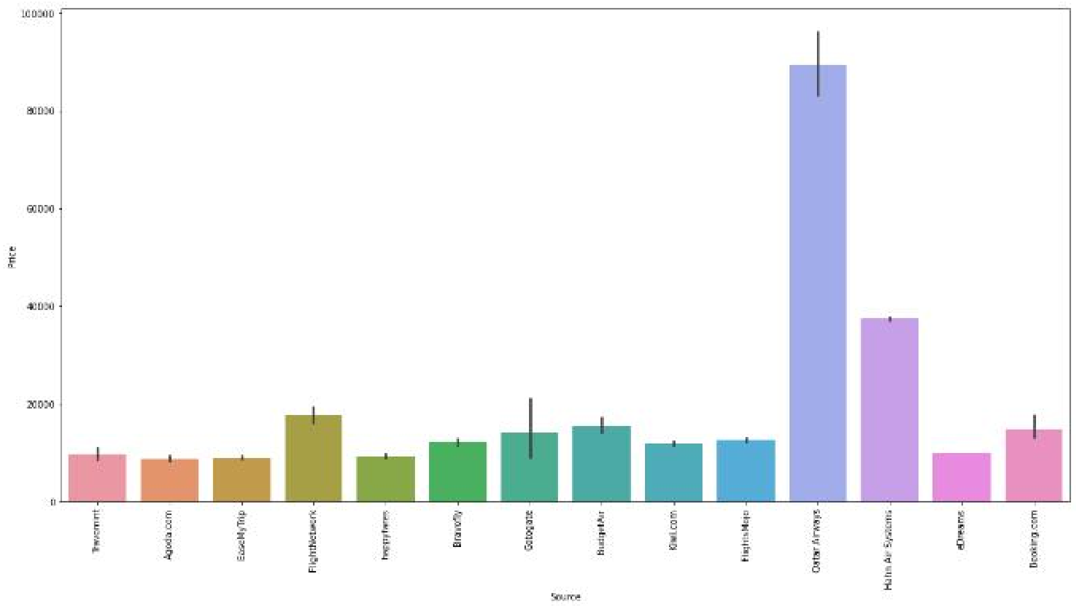
- The most of the flights which were travelling from Goa to Delhi from 3rd Feb to 7th Feb will be Air Asia amongst all the other airline companies.
- Most of the prices which we have got from the source/site are from Flight Network.
- All the flights departure is from Goa International Airport and arrival will be on Delhi Indira Gandhi Airport.
- The data which we got from the date 3rd Feb to 7th Feb, in this most of the flights will be taking 1stop while travelling from Goa to Delhi.

Bar Plots:

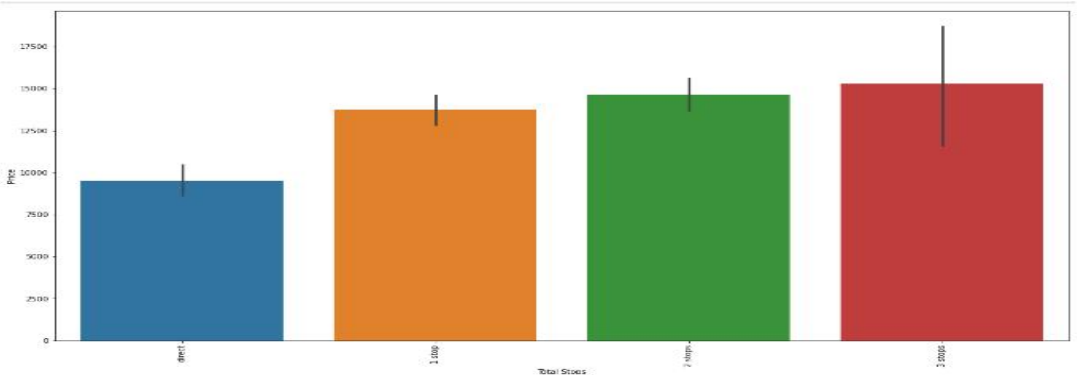
Airline Name vs Price



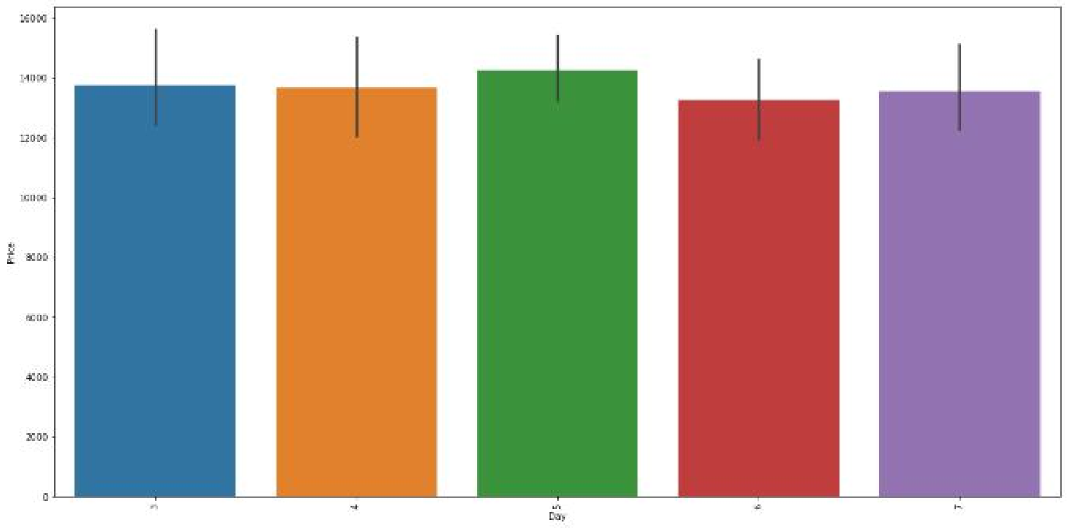
Source vs Price



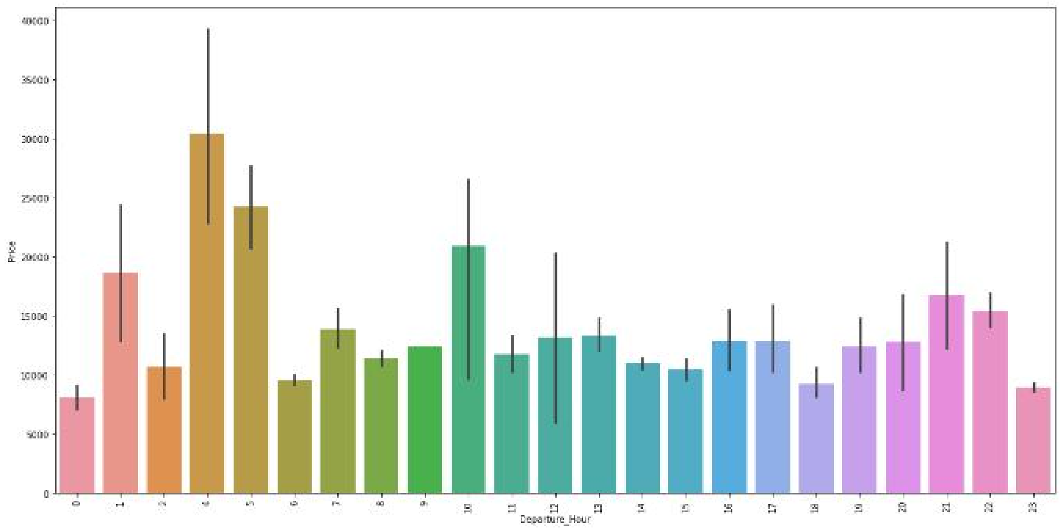
Total stops vs Price



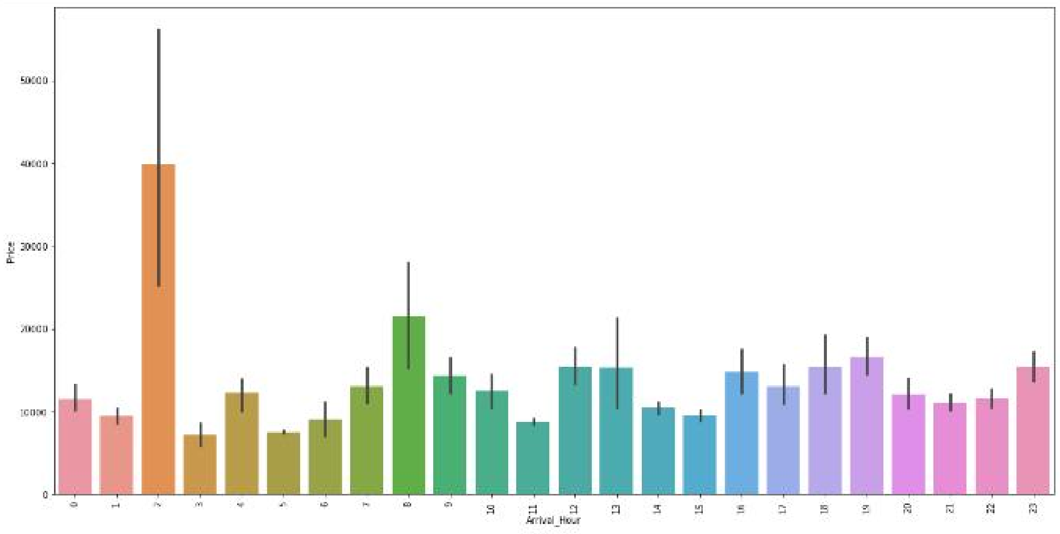
Day vs Price



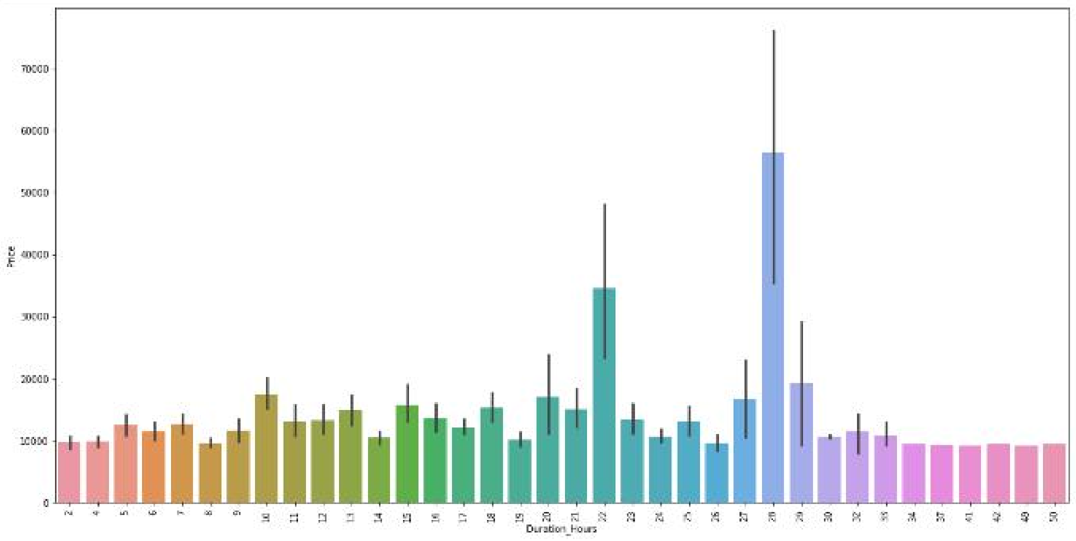
Departure Hour vs Price



Arrival vs Price



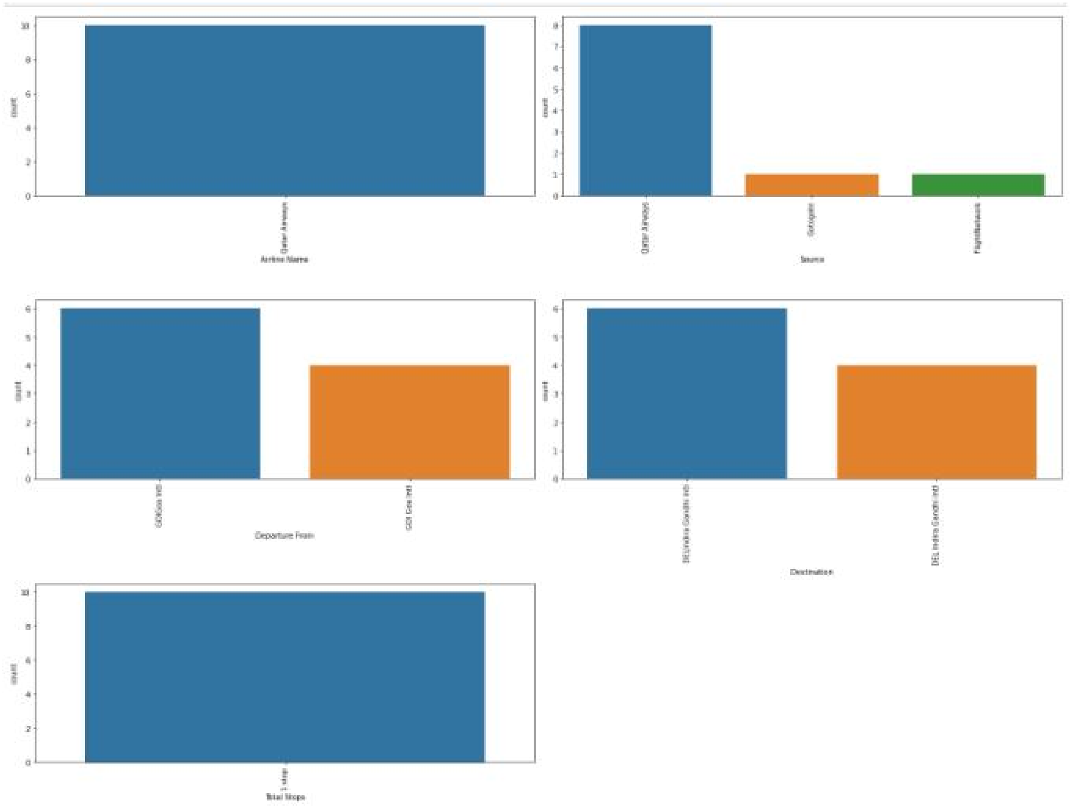
Duration Hour vs Price



Observations:

- In plot Airline Name vs Price we can see that Qatar airways have the highest prices for flight ticket which is more than 80000.
- In plot Source vs Price we can see that source Qatar airways showing highest prices for flight ticket.
- In plot Total Stops vs Price we can see that whenever there is a 3 stop in between while travelling from Goa to Delhi prices are higher as compared to others.
- In plot Day vs Price we can see that on 5th Feb prices of flight tickets are higher than the other dates may be because of Sunday.
- In plot Departure Hour vs Price we can see that early morning ticket prices are higher that is at 4am ticket prices are higher as compared to the other timings.
- In plot Arrival Hour vs Price we can see that whenever the arrival hour is at 2am the ticket prices are higher as compared to others.
- In plot Duration Hour vs Price we can see that the duration hours which is 28 hours have the highest ticket price.

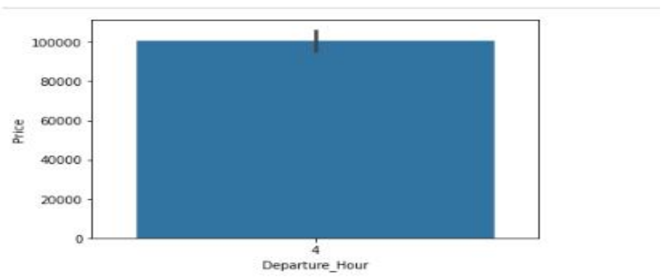
Highest Ticket Prices Count Plot(Highest 10)



Observation:

- In the above plots we can see the plots plotted from the data which have the highest 10 prices from date 3rd Feb to 7th Feb.
- So in above plots we can see that Qatar airways have the most expensive flight tickets.
- Again Qatar airways source have the highest of the flight tickets.
- And while travelling all the Qatar airways took only 1 stop in between.

Bar Plot:

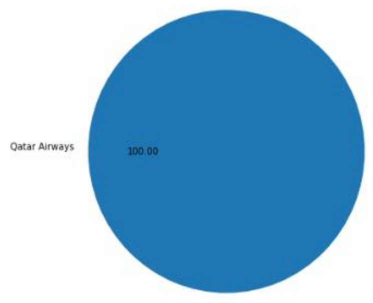


Observation:

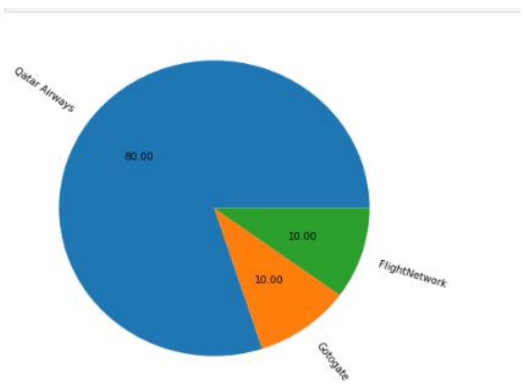
- So in this plot we can see that whenever Qatar airways departure time was 4am that is in the early morning, the prices of the flight tickets was always expensive.

Pie Plots:

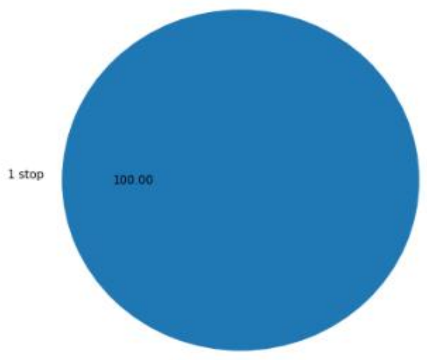
Airline Name



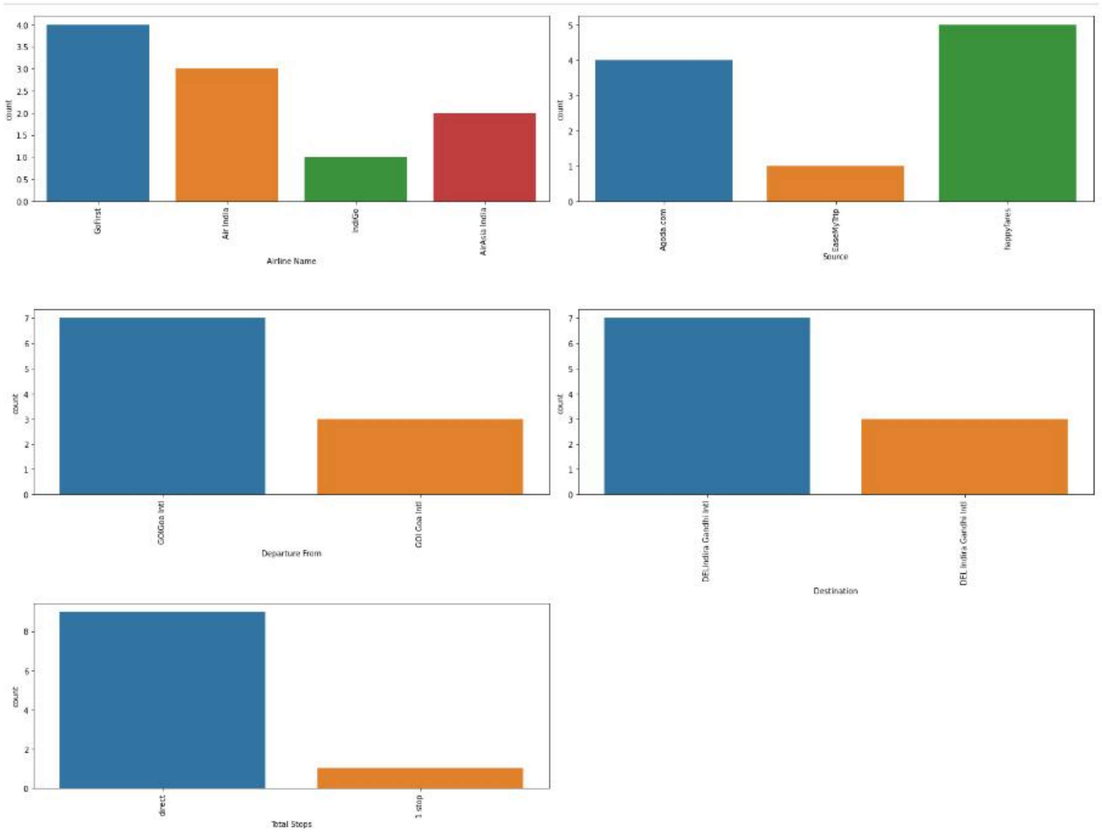
Source



Total Stops



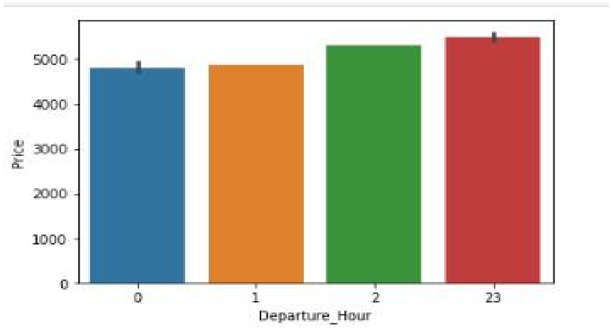
Lowest Ticket Prices Count Plot(Lowest 10)



Observation:

- In the above plots we can see the plots plotted from the data which have the lowest 10 prices from date 3rd Feb to 7th Feb.
- So in above plots we can see that Go First, Air India, Indigo and Air Asia India have the cheapest flight tickets.
- Agoda.com, Easemytrip and Happyfares source have the cheapest of the flight tickets.
- And while travelling from Goa to Delhi Most of the flights did’nt took any stops in between.

Bar Plot:

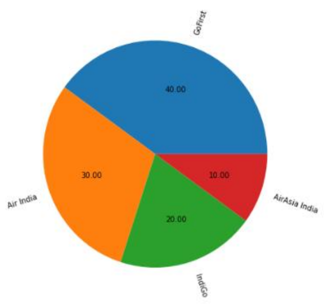


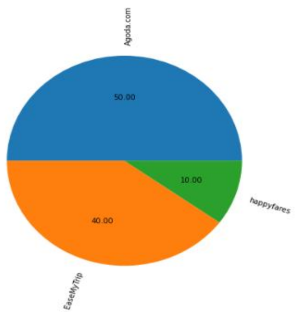
Observation:

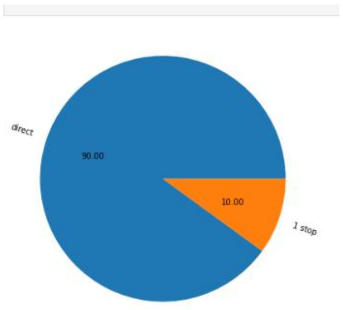
- So in this plot we can see that whenever the flight departure timing was in the midnight or close to midnight that is at 11pm or 12 am or 1am or 2am that time prices of the flight tickets were cheapest.

Pie Plots:

Airline Name **Source** **Total Stops**







INTERPRETATION OF THE RESULTS

Results:

- Null values were not present.
- Duplicates were present in the dataset.
- Dataset was not normally distributed.

- Most of the features does not have outliers except one column.
- Only two columns data was found to be skewed.
- Xtreme Gradient Boosting was found to be best suited for the dataset.

CONCLUSION

Key Findings and Conclusions of the Study:

The case study aims to give an idea of applying machine learning algorithms to predict the price of the flight tickets. After the completion of this project, we got an insight of how to collect data, pre-processing the data, analyse the data, cleaning the data and building a model.

In this we have used 4 learning models to predict the price of the flight ticket.

From our detailed analysis we can determine the following:

- Flight ticket prices change during the morning and evening time of the day.
- Some flights more specifically Qatar Airways which are departing in the early morning that is at 4 AM are having most expensive ticket prices.
- Some flights such as Go First, Air Asia, Indigo they have the cheapest ticket price especially in the late night or around mid night that is around 11 PM, 12 AM, 1AM like that.
- From the analysis we came to know that late nights flights are cheaper compared to working hours.
- The last minute flights are always expensive.

- As in our data jet airways is not there, so I will compare between Indigo and Spice jet. So by looking at the plots we can see that Spice jet are slightly expensive than that of Indigo.
- Morning flights as in early morning flights which departs at 4am are most expensive specifically the Qatar Airways.

Learning Outcomes of the Study in respect of Data Science:

While working on this project I have learned many things about the features of the flights and about the flight ticket selling website and came to know how the machine learning models have helped to predict the price of the flight tickets.

I found that the project was quite interesting as I came to know about the trend of the flight tickets, when the prices gonna rise, when the prices will be cheaper, which airline and source are the cheaper ones and at what time prices are expensive and cheapest. So overall the experience was good as I came across a new challenge.

Visualization plays an important role in data analysis as it gives you a perfect idea of what is going through the dataset and also data cleaning was one of the most important and crucial thing about the project.

Finally we achieved what we had visualized by predicting the flight ticket prices and by building the flight price evaluation model that could help the buyers to understand the future flight ticket prices.

Limitations of this work and Scope for Future Work:

Limitations: The main limitation is the low number of records that have been used. Many of the values in some columns were having object data type which I had taken care.

Future Work: The greatest shortcoming of this work is the shortage of data.And I have scrapped the data of 5 days. So anybody wishing to expand upon it should seek alternative sources of historical data manually over period of time. And should scrapped the data of the months of festive seasons. So that we can get a information of that also how the prices changes when there are festive seasons.