FLIP ROBO

fight Price Prediction Project

Submitted by:

Vikash Kumar Singh

ACKNOWLEDGMENT

In this project, I took references from our DataTrained's video of Shankargouda Tegginmani sir and some websites which are- https://www.youtube.com https://www.kaggle.com , https://www.github.com , https://stackoverflow.com and https://scikitlearn.org in completion of this project. The data source is provided by Flip Robo company.

INTRODUCTION

- **Problem statement**

  Anyone who has booked a flight 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 – So, you have to work on a project where you collect data of flight fares with other features and work to make a model to predict fares of flights.

- **Conceptual Background of the Domain Problem**

  1. In preparation of any Machine Learning Model, we have to have conceptual knowledge of cleansing, scaling, outlier and many more method to perform the EDA analysis
  2. The knowledge of Data Visualization is also required to observe the skewness, outlier, correlation and multicollinearity of the given dataset and you are also required to have knowledge to reduce the following observation.
  3. Use standard techniques (PCA, Scaling, encoding, cross validation, grid search, ensemble techniques) wherever applicable.

- **Review of Literature**

  The literature study gives an overview of this project and the EDA methods, the feature engineering methods that have been used in this study. As well as about evaluation metrics to measure the performance of the algorithms.

- **Motivation for the Problem Undertaken**

  Presently, I am doing an internship with FlipRobo company and they have given us this project to build a Machine Learning Model to predict price of flight tickets.

  This project gives me an opportunity to recall the teaching of Datatrained's mentor and to build multiple classification model with better accuracy. This will help me in upgrading my skills and knowledge.

Analytical Problem Framing

- **Mathematical/ Analytical Modeling of the Problem**

  In the given dataset, target is in continuous form so, the model will be regression model (supervised learning) in which, we will design a predictive model to analyses the relation between features and target variables (ticket price of flights).

Data Sources and their formats

We have scraped the data from the website: yatra.com with different location in India and saved the required data as CSV file.

Link: - https://www.yatra.com/

- **Data Pre-processing Done**

  In pre-processing, we have done the following steps:

  1.) Imported some important libraries in our jupyter notebook.

  2.) Imported the given dataset

  3.) Check the details of each feature from the given dataset.

4.) Cleanse the dataset i.e., check for null values, duplicate value, shape of dataset etc.
5.) Drop the unnecessary features.
6.) Checked the skewness
7.) Transform the dataset with power transformation
8.) Outlier detection
9.) Checked for correlation
10.) Visualization
11.) Separation of features and label
12.) Scaling of the data
13.) Train test split.

- **Data Inputs- Logic- Output Relationships**

	Airline_name	From	To	Arival_time	Stops	Price	Depart hour	Depart min	Duration_hours	Duration_mins
Airline_name	1.000000	0.000048	0.014409	0.109776	-0.002100	0.259712	0.124714	0.188606	0.014563	-0.010949
From	0.000048	1.000000	-0.615891	-0.003407	-0.003096	-0.120409	0.084763	-0.071409	-0.050059	0.023887
To	0.014409	-0.615891	1.000000	0.010467	0.005996	0.080950	-0.067583	0.032619	-0.018129	-0.058812
Arival_time	0.109776	-0.003407	0.010467	1.000000	0.013580	0.150159	-0.067196	0.003942	0.044804	0.013302
Stops	-0.002100	-0.003096	0.005996	0.013580	1.000000	-0.403853	0.011171	-0.030322	-0.657030	-0.049168
Price	0.259712	-0.120409	0.080950	0.150159	-0.403853	1.000000	-0.066381	0.050927	0.425468	0.022272
Depart hour	0.124714	0.084763	-0.067583	-0.067196	0.011171	-0.066381	1.000000	0.010927	0.068553	-0.015768
Depart min	0.188606	-0.071409	0.032619	0.003942	-0.030322	0.050927	0.010927	1.000000	0.074375	-0.028818
Duration_hours	0.014563	-0.050059	-0.018129	0.044804	-0.657030	0.425468	0.068553	0.074375	1.000000	-0.035028
Duration_mins	-0.010949	0.023887	-0.058812	0.013302	-0.049168	0.022272	-0.015768	-0.028818	-0.035028	1.000000

- **State the set of assumptions (if any) related to the problem under consideration**

My only presumption was that the flight price will increase with instant purchasing.

- **Hardware and Software Requirements and Tools Used**
  - **Hardware tools:**

    1. Windows laptop
    2. i5 processor
    3. 4GB ram 4. 250 GB SSD card

  - **Software tools:**

    1. windows 10
    2. Anaconda Navigator
    3. Jupyter Notebook
    4. Python

    • **Libraries and packages:**

    1. Pandas
    2. NumPy
    3. SciPy
    4. Seaborn
    5. Mat plot
    6. Sklearn

Model/s Development and Evaluation

- **Identification of possible problem-solving approaches (methods)**

  From the domain knowledge and my own understanding, I have followed these steps:

  1. Importing the given test and train dataset
  2. Checking null values in the dataset
  3. Imputation of given dataset
  4. Transformation of given dataset
  5. Standardization of given dataset
  6. Splitting the train and test dataset

- **Testing of Identified Approaches (Algorithms)**

  1. Linear Regression
  2. Random Forest Regressor
  3. Decision Tree Regressor
  4. Support Vector Regressor
  5. KNeighbors Regressor
  6. XGBRegressor
  7. Lasso
  8. Ridge
  9. AdaBoost Regressor
  10. Gradient Boosting Regressor

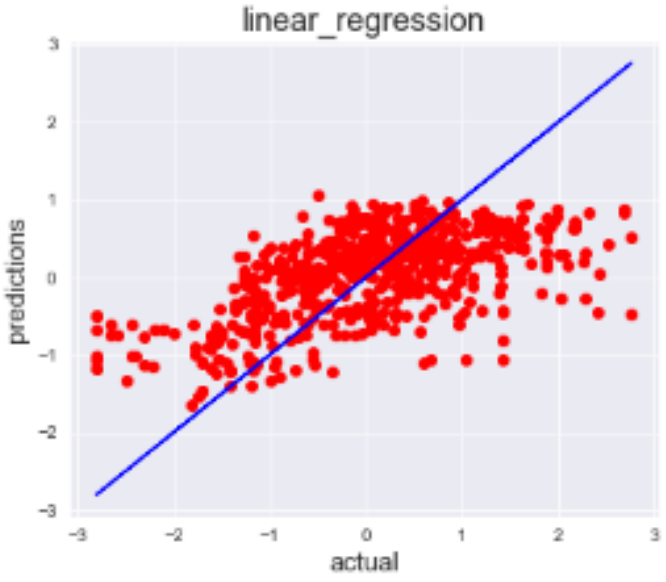
- **Run and evaluate selected models**
    1. **Linear regression**

```
from sklearn.linear_model import LinearRegression
import statsmodels.api as sm

regressor=LinearRegression()
regressor.fit(X_train,y_train)
y_pred = regressor.predict(X_test)
regression_results(y_test,y_pred)
model_accuracy(regressor)

plt.figure(figsize = (6,5))
plt.scatter(x = y_test, y=y_pred, color = 'r')
plt.plot(y_test,y_test, color = 'b')
plt.xlabel('actual', fontsize = 15)
plt.ylabel('predictions', fontsize = 15)
plt.title('linear_regression', fontsize = 18)
plt.show()
```

```
Explained_variance:  0.3417
R2:  0.3388
Adjusted_r2:  0.3291
MAE:  0.6821
MSE:  0.7266
RMSE:  0.8524
Accuracy: 28.56 %
Standard Deviation: 2.21 %
```



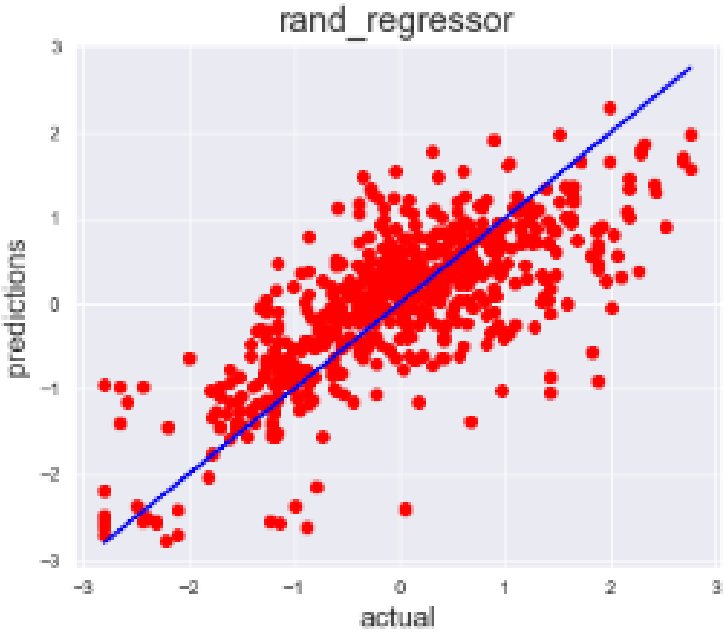
2. **Random forest regressor**

2. RandomForestRegressor

```
rand_regressor = RandomForestRegressor()
rand_regressor.fit(X_train, y_train)
y_pred_rf = rand_regressor.predict(X_test)
regression_results(y_test,y_pred_rf)
model_accuracy(rand_regressor)

plt.figure(figsize = (6,5))
plt.scatter(x = y_test, y=y_pred_rf, color = 'r')
plt.plot(y_test,y_test, color = 'b')
plt.xlabel('actual', fontsize = 15)
plt.ylabel('predictions', fontsize = 15)
plt.title('rand_regressor', fontsize = 18)
plt.show()
```

```
Explained_variance:  0.541
R2:  0.54
Adjusted_r2:  0.5332
MAE:  0.5388
MSE:  0.5055
RMSE:  0.711
Accuracy: 48.93 %
Standard Deviation: 5.37 %
```



3. Decision tree regressor

```
from sklearn.tree import DecisionTreeRegressor

decision_tree=DecisionTreeRegressor(criterion='mse',splitter='random',random_state=10)
decision_tree.fit(X_train, y_train)
y_pred_dt = decision_tree.predict(X_test)
regression_results(y_test,y_pred_dt)
model_accuracy(decision_tree)

plt.figure(figsize = (6,5))
plt.scatter(x = y_test, y=y_pred_dt, color = 'r')
plt.plot(y_test,y_test, color = 'b')
plt.xlabel('actual', fontsize = 15)
plt.ylabel('predictions', fontsize = 15)
plt.title('decision_tree', fontsize = 18)
plt.show()
```

```
Explained_variance:  0.3132
R2:  0.3118
Adjusted_r2:  0.3017
MAE:  0.6223
MSE:  0.7562
RMSE:  0.8696
Accuracy: 22.91 %
Standard Deviation: 7.41 %
```

4. SVR

```
from sklearn.svm import SVR

svr=SVR()
svr.fit(X_train, y_train)
y_pred_svr = svr.predict(X_test)

regression_results(y_test,y_pred_svr)
model_accuracy(svr)

plt.figure(figsize = (6,5))
plt.scatter(x = y_test, y=y_pred_svr, color = 'r')
plt.plot(y_test,y_test, color = 'b')
plt.xlabel('actual', fontsize = 15)
plt.ylabel('predictions', fontsize = 15)
plt.title('svr', fontsize = 18)
plt.show()
```

```
Explained_variance:  0.4422
R2:  0.438
Adjusted_r2:  0.4298
MAE:  0.6064
MSE:  0.6176
RMSE:  0.7858
Accuracy: 39.31 %
Standard Deviation: 3.86 %
```

5. KNeighbors regressor

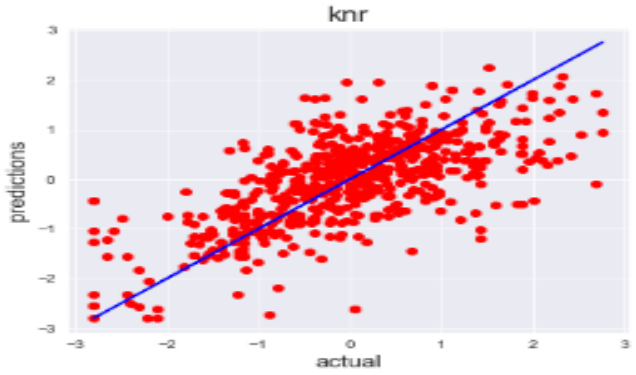
```
from sklearn.neighbors import KNeighborsRegressor

knr=KNeighborsRegressor(n_neighbors=2)
knr.fit(X_train, y_train)
y_pred_knr = knr.predict(X_test)

regression_results(y_test,y_pred_knr)
model_accuracy(knr)

plt.figure(figsize = (6,5))
plt.scatter(x = y_test, y=y_pred_knr, color = 'r')
plt.plot(y_test,y_test, color = 'b')
plt.xlabel('actual', fontsize = 15)
plt.ylabel('predictions', fontsize = 15)
plt.title('knr', fontsize = 18)
plt.show()
```

```
Explained_variance:  0.4168
R2:  0.4167
Adjusted_r2:  0.4082
MAE:  0.6043
MSE:  0.6409
RMSE:  0.8006
Accuracy: 26.51 %
Standard Deviation: 4.04 %
```



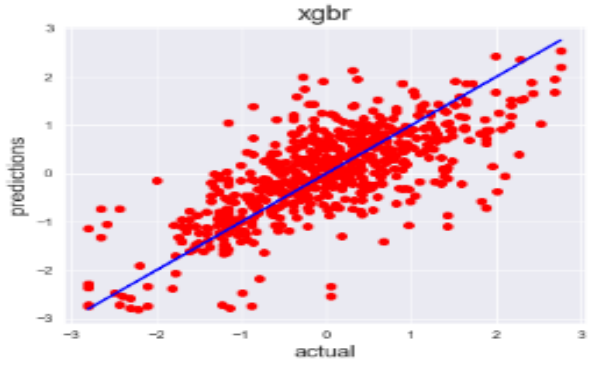
6. XGBRegresso

```
from xgboost import XGBRegressor
xgbr=XGBRegressor(random_state=10)
xgbr.fit(X_train, y_train)
y_pred_xgbr = xgbr.predict(X_test)

regression_results(y_test,y_pred_xgbr)
model_accuracy(xgbr)

plt.figure(figsize = (6,5))
plt.scatter(x = y_test, y=y_pred_xgbr, color = 'r')
plt.plot(y_test,y_test, color = 'b')
plt.xlabel('actual', fontsize = 15)
plt.ylabel('predictions', fontsize = 15)
plt.title('xgbr', fontsize = 18)
plt.show()
```

```
Explained_variance:  0.4867
R2:  0.4856
Adjusted_r2:  0.4781
MAE:  0.5607
MSE:  0.5652
RMSE:  0.7518
Accuracy: 44.14 %
Standard Deviation: 5.83 %
```



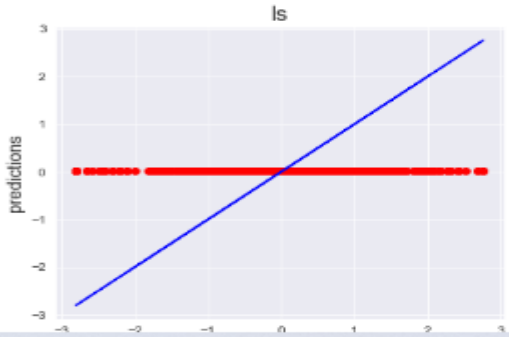
7. Lasso

```
ls = Lasso()
ls=Lasso(random_state=10)
ls.fit(X_train, y_train)
y_pred_ls = ls.predict(X_test)

regression_results(y_test,y_pred_ls)
model_accuracy(ls)

plt.figure(figsize = (6,5))
plt.scatter(x = y_test, y=y_pred_ls, color = 'r')
plt.plot(y_test,y_test, color = 'b')
plt.xlabel('actual', fontsize = 15)
plt.ylabel('predictions', fontsize = 15)
plt.title('ls', fontsize = 18)
plt.show()
```

```
Explained_variance:  0.0
R2:  -0.0032
Adjusted_r2:  -0.0179
MAE:  0.8253
MSE:  1.1024
RMSE:  1.05
Accuracy: -0.31 %
Standard Deviation: 0.30 %
```



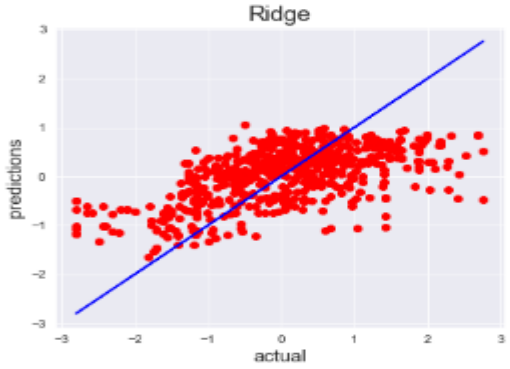
8. Ridge

```
rd = Ridge()
rd=Ridge(random_state=10)
rd.fit(X_train, y_train)
y_pred_rd = rd.predict(X_test)

regression_results(y_test,y_pred_rd)
model_accuracy(rd)

plt.figure(figsize = (6,5))
plt.scatter(x = y_test, y=y_pred_rd, color = 'r')
plt.plot(y_test,y_test, color = 'b')
plt.xlabel('actual', fontsize = 15)
plt.ylabel('predictions', fontsize = 15)
plt.title('Ridge', fontsize = 18)
plt.show()
```

```
Explained_variance:  0.3417
R2:  0.3388
Adjusted_r2:  0.329
MAE:  0.6821
MSE:  0.7266
RMSE:  0.8524
Accuracy: 28.56 %
Standard Deviation: 2.21 %
```



9. **AdaBoosting regressor**

```
abr = AdaBoostRegressor()
abr=AdaBoostRegressor(random_state=10)
abr.fit(X_train, y_train)
y_pred_abr = abr.predict(X_test)

regression_results(y_test,y_pred_abr)
model_accuracy(abr)

plt.figure(figsize = (6,5))
plt.scatter(x = y_test, y=y_pred_abr, color = 'r')
plt.plot(y_test,y_test, color = 'b')
plt.xlabel('actual', fontsize = 15)
plt.ylabel('predictions', fontsize = 15)
plt.title('AdaBoost Regressor', fontsize = 18)
plt.show()
```

```
Explained_variance:  0.4898
R2:  0.4565
Adjusted_r2:  0.4485
MAE:  0.6465
MSE:  0.5972
RMSE:  0.7728
Accuracy: 37.63 %
Standard Deviation: 2.91 %
```

10. **GradientBoosting classifier**

```
gbr = GradientBoostingRegressor()
gbr=GradientBoostingRegressor(random_state=10)
gbr.fit(X_train, y_train)
y_pred_gbr = gbr.predict(X_test)

regression_results(y_test,y_pred_gbr)
model_accuracy(gbr)

plt.figure(figsize = (6,5))
plt.scatter(x = y_test, y=y_pred_gbr, color = 'r')
plt.plot(y_test,y_test, color = 'b')
plt.xlabel('actual', fontsize = 15)
plt.ylabel('predictions', fontsize = 15)
plt.title('GradientBoosting Regressor', fontsize = 18)
plt.show()
```

```
Explained_variance:  0.5799
R2:  0.5793
Adjusted_r2:  0.5732
MAE:  0.5379
MSE:  0.4623
RMSE:  0.6799
Accuracy: 52.63 %
Standard Deviation: 3.00 %
```

We have built different models of regression, in which we are evaluating them with cross validation score, Explained variance,r2 score ,adjusted r2 score, MSE, RMSE .Among all, Random forest regressor model has highest accuracy.

For further improvement in the accuracy, we have done hyperparameter tuning for this this model and it improved the accuracy.

After hyperparameter tuning.

```
param = {
              'min_samples_split':[2,5,10],
              'min_samples_leaf':[1,3,4],
              'max_depth':[5,10,15],
              'learning_rate': [0.1,0.2,0.3],
               'n_estimators':[150,200,250],
              }
```

```
grd = GridSearchCV(gbr,param_grid = param)
grd.fit(X_train,y_train)

print('Best Params ',grd.best_params_)
```

```
Best Params  {'learning_rate': 0.1, 'max_depth': 5, 'min_samples_leaf': 4, 'min_samples_split': 10, 'n_estimators': 150}
```

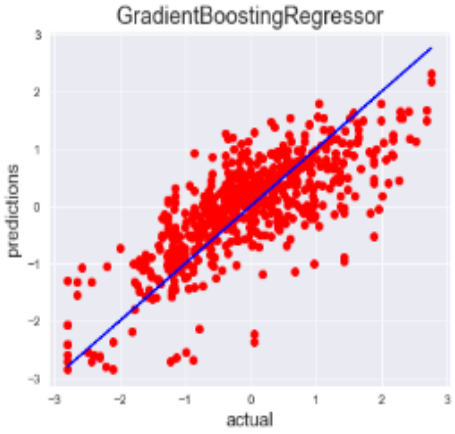
After tuning its accuracy has increased by very little margin.

```
gbr= GradientBoostingRegressor(learning_rate=0.1, n_estimators=150, min_samples_split=10, min_samples_leaf=4, max_depth=5)
gbr.fit(X_train, y_train)
y_pred =gbr.predict(X_test)

regression_results(y_test,y_pred)
model_accuracy(gbr)

plt.figure(figsize = (6,5))
plt.scatter(x = y_test, y=y_pred, color = 'r')
plt.plot(y_test,y_test, color = 'b')
plt.xlabel('actual', fontsize = 15)
plt.ylabel('predictions', fontsize = 15)
plt.title('GradientBoostingRegressor', fontsize = 18)
plt.show()
```

```
Explained_variance:  0.5728
R2:  0.5724
Adjusted_r2:  0.5661
MAE:  0.5256
MSE:  0.4699
RMSE:  0.6855
Accuracy: 52.03 %
Standard Deviation: 4.22 %
```



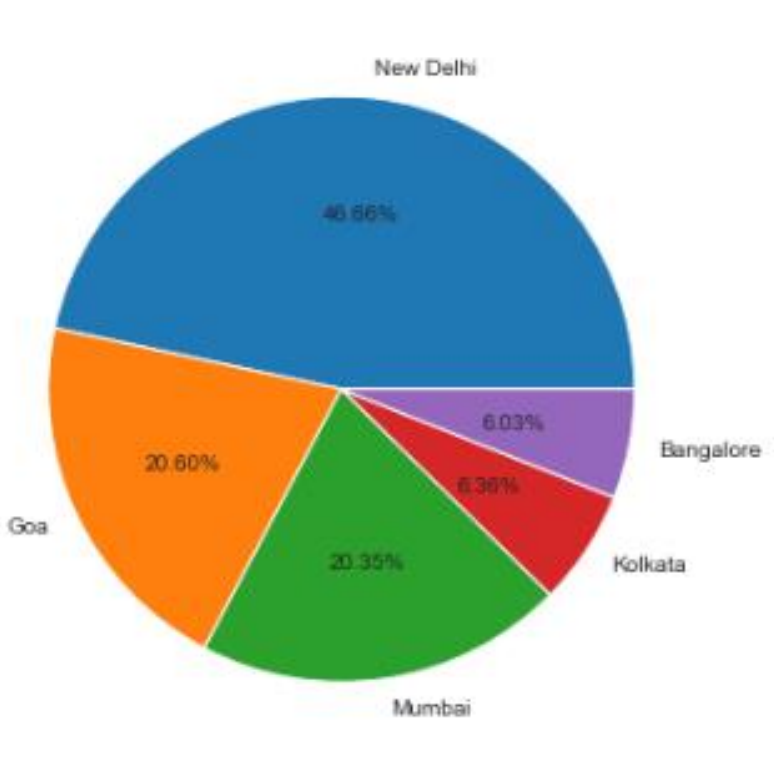

```
loaded_model=pickle.load(open('Flight_Price_Prediction_Project','rb'))
result=loaded_model.score(X_test,y_test)
print(result*100)
```

```
57.239916271683676
```

- **Key Metrics for success in solving problem under consideration**
  a. Cross validation score
  b. Explained variance
  b. R2 score
  c. Adjusted r2 score
  d. MSE
  e. RMSE score
  f. Accuracy

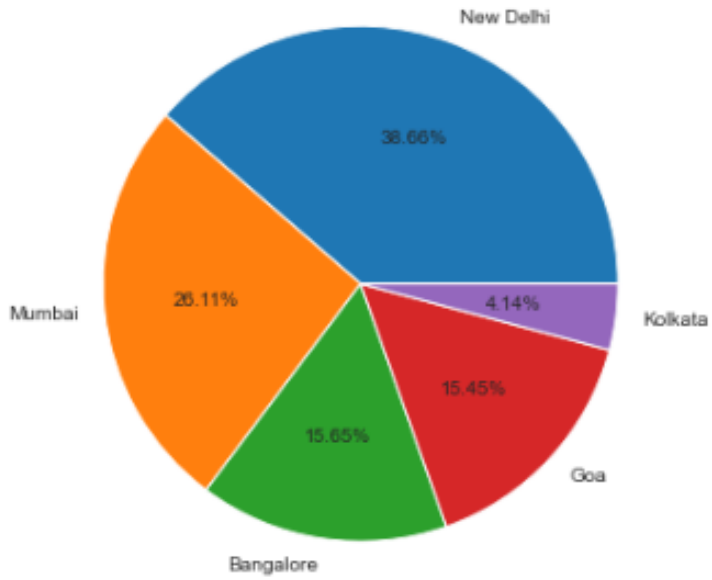
- **Visualizations**

  1. Pie plot of source location.



In this plot, we can clearly see that the
Delhi has maximum flights to different places followed by Goa and Mumbai in our dataset.

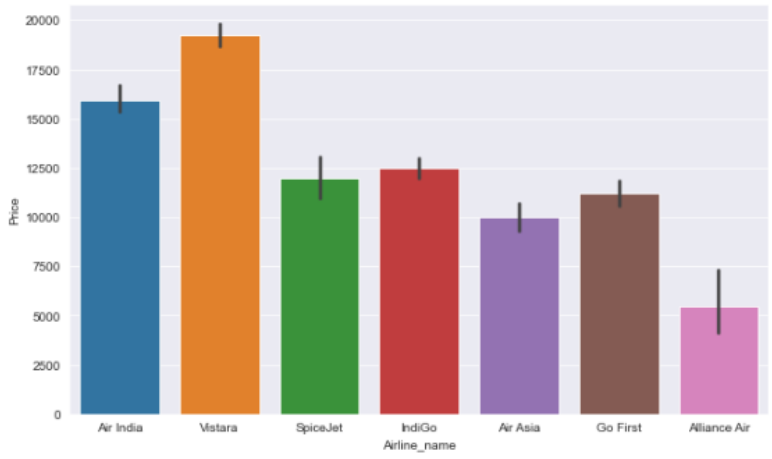
2. Pie plot of destination location



In this plot, we can clearly see that the
Delhi has maximum flights to different places followed by Mumbai and Goa in our dataset.

3. Bar plot of price of different airlines

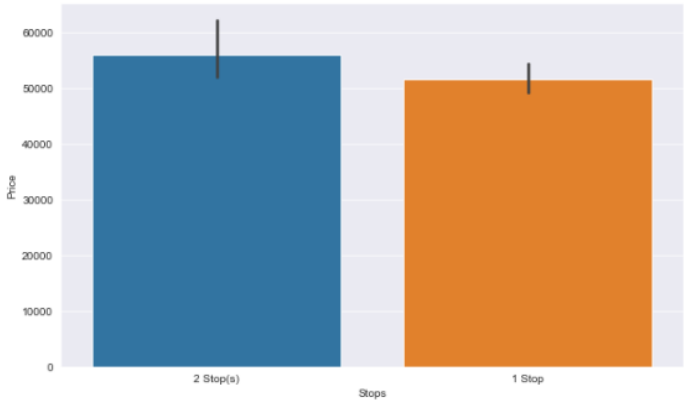
```
plt.figure(figsize=(10,6))
df_good = df1.sort_values(by="Price",ascending=False)
sns.barplot(x="Airline_name",y="Price",data=df_good)
plt.show()
```



As per this dataset, Vistara airline's has costliest fare followed by Air India and other different airlines.

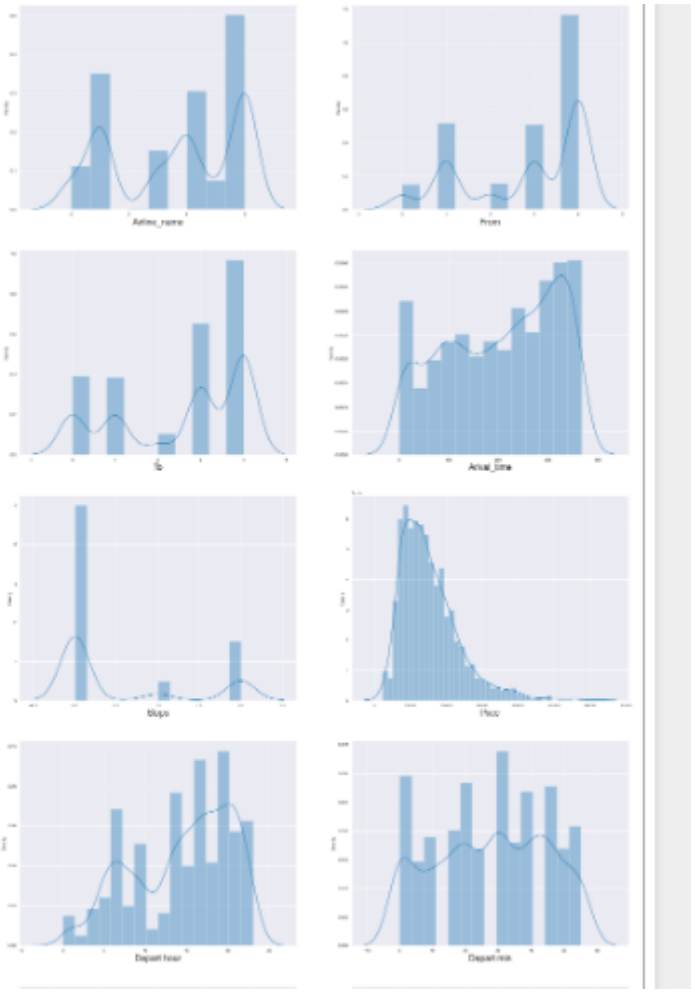
3. Bar plot of number of stoppages during the travel.

```
plt.figure(figsize=(10,6))
df_good = df1.sort_values(by="Price",ascending=False).iloc[0:10]
sns.barplot(x="Stops",y="Price",data=df_good)
plt.show()
```

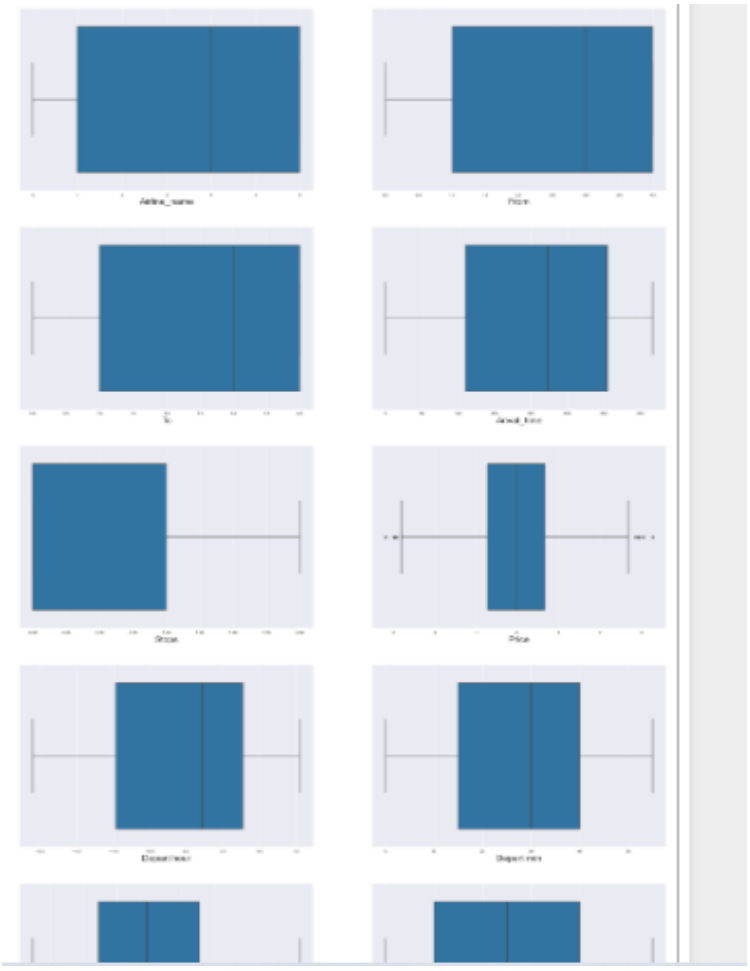


Maximum stoppage are two.

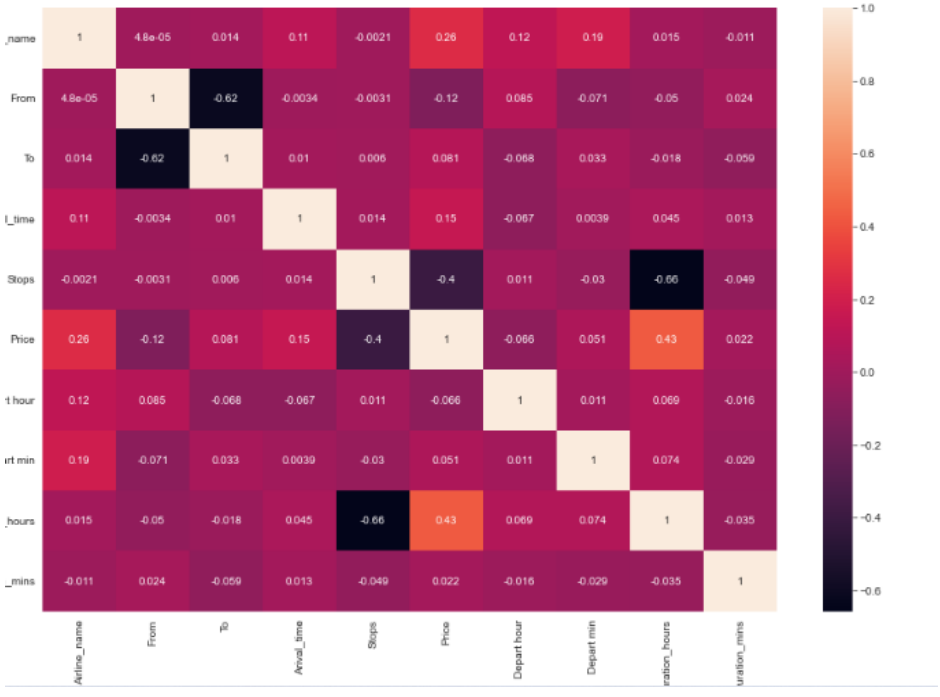
5. distribution of numerical features



7. Box plot to detect outlier



8. Heatmap



- **Interpretation of the Results**

  After Visualization, we have observed the skewness, outlier, correlation and multicollinearity of the data, and have tried to reduce it by power transformation technique, label encoder technique and interquartile range.

  After doing 10 different modelling, we analyzed that the Gradient Boosting Regressor model is best among all.

CONCLUSION

- **Key Findings and Conclusions of the Study**

Post models building and choosing the appropriate model I want ahead and scrape the data and join the dataset. After applying all the data preprocessing steps as the dataset, I was then able to get the predicted price result. Once the dataset with feature columns is predicted label was format, I exported the value in a comma separated value file to be accessed as needed. We have performed Linear regression, Random Forest regressor, Decision Tree regressor, SVR, KNeighbours regressor, Lasso, Ridge, Adaboost regressor, XGB regressor and Gradient boosting regressor and hyperparameter tuning as we understand that parameterization can drive the significant result in the performance.

Conclusion

```
loaded_model=pickle.load(open('Flight_Price_Prediction_Project','rb'))
result=loaded_model.score(X_test,y_test)
print(result*100)
```

```
57.239916271683676
```

- **Learning Outcomes of the Study in respect of Data Science**

i. From different visualization, we have learned about the skewness, outlier and multicollinearity of the dataset and it helped in making better model.

ii. Data cleansing- we learned, how to fill or drop the null values data for the dataset by performing multiple statistical operation.

iii. Machine learning algorithm- we have performed 10 multiple regression model to predict the flight price prediction, out of all Gradient boosting regressor have performed best among the others. We have also the Hyper Para meter tuning to the improve of model accuracy.

- **Limitations of this work and Scope for Future Work**

Limitations of this project were that we haven't cover all regressor algorithms in our Data Science course, instead of it, they have focused on the basic algorithm. In Hyper Para Meter tuning, parameter for different model is very difficult to choose.

"Thank you"