FLIP ROBO

Flight Price Prediction

Submitted By-
Akanksha Amarnani

Acknowledgement

I would like to thank Almighty for giving me the confidence to pursue this project. Further, the concepts from DataTrained Academy guided me to complete the project.
In addition I would like to thank my mentor from Flip Robo Technology, Ms Khushboo Garg for clarifying my doubts and queries.

The references used for the completion of this project are-

- A Framework for Airfare Price Prediction: A Machine Learning Approach, Tiany Wangi *et al,* July 2019, Conference: 2019 IEEE 20th International Conference on Information Reuse and Integration for Data Science (IRI)
- Flight Price Prediction for Users by Machine Learning Techniques, Pavithra Maria , Anitha K, International Advanced Research Journal in Science, Engineering and Technology Vol. 8, Issue 3, March 2021
- Flight Price Prediction: A Case Study, 2022, International Journal for Research in Applied Science & Engineering Technology (IJRASET)

INTRODUCTION

Business Problem Framing-

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 -
1. Time of purchase patterns (making sure last-minute purchases are expensive)
2. 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 back inventory for those expensive last-minute expensive purchases)

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-

The domain related concepts which help us in a better understanding are-

- Web Scraping- Web scraping data of flights from different websites
- Exploratory Data Analysis (EDA)- By conducting explanatory data analysis, we obtain a better understanding of our data. This yields insights that can be helpful later when building a model, as well as insights that are independently interesting.
- Splitting the data- The dataset is split into train and test sample using the train test split
- Modeling- We apply Linear Regression, Random Forest Regressor and Ada Boost Regressor models for prediction of the prices for the house
- Regularization- Models are regularized and the parameters are hypertuned to enhance the efficiency of the models

Review of Literature-

Machine learning is a subfield of Artificial Intelligence (AI) that works with algorithms and technologies to extract useful information from data. Machine learning methods are appropriate in big data since attempting to manually process vast volumes of data would be impossible without the support of machines. Machine learning in computer science attempts to solve problems algorithmically rather than purely mathematically. Therefore, it is based on creating algorithms that permit the machine to learn. However, there are two general groups in machine learning which are supervised and unsupervised. Supervised is where the program gets trained on pre-determined set to be able to predict when a new data is given. Unsupervised is where the program tries to find the relationship and the hidden pattern between the data

The performance of the model build will be measured upon predicting flight ticket prices since the prediction in many regression algorithms relies not only on a specific feature but on an unknown number of attributes that result in the value to be predicted. Determining the listed price of a flight ticket is a challenging task, due to the many factors such as its source, destination, duration, etc.. The focus of this project is developing machine learning models that can accurately predict the price of a flight ticket based on its features, in order to make informed purchases. We implement and evaluate various learning methods on a dataset consisting of the flight ticket prices departing from various cities in India. The data used in the experiment will be handled by using a combination of pre-processing methods to improve the prediction accuracy

Motivation for the Problem Undertaken-

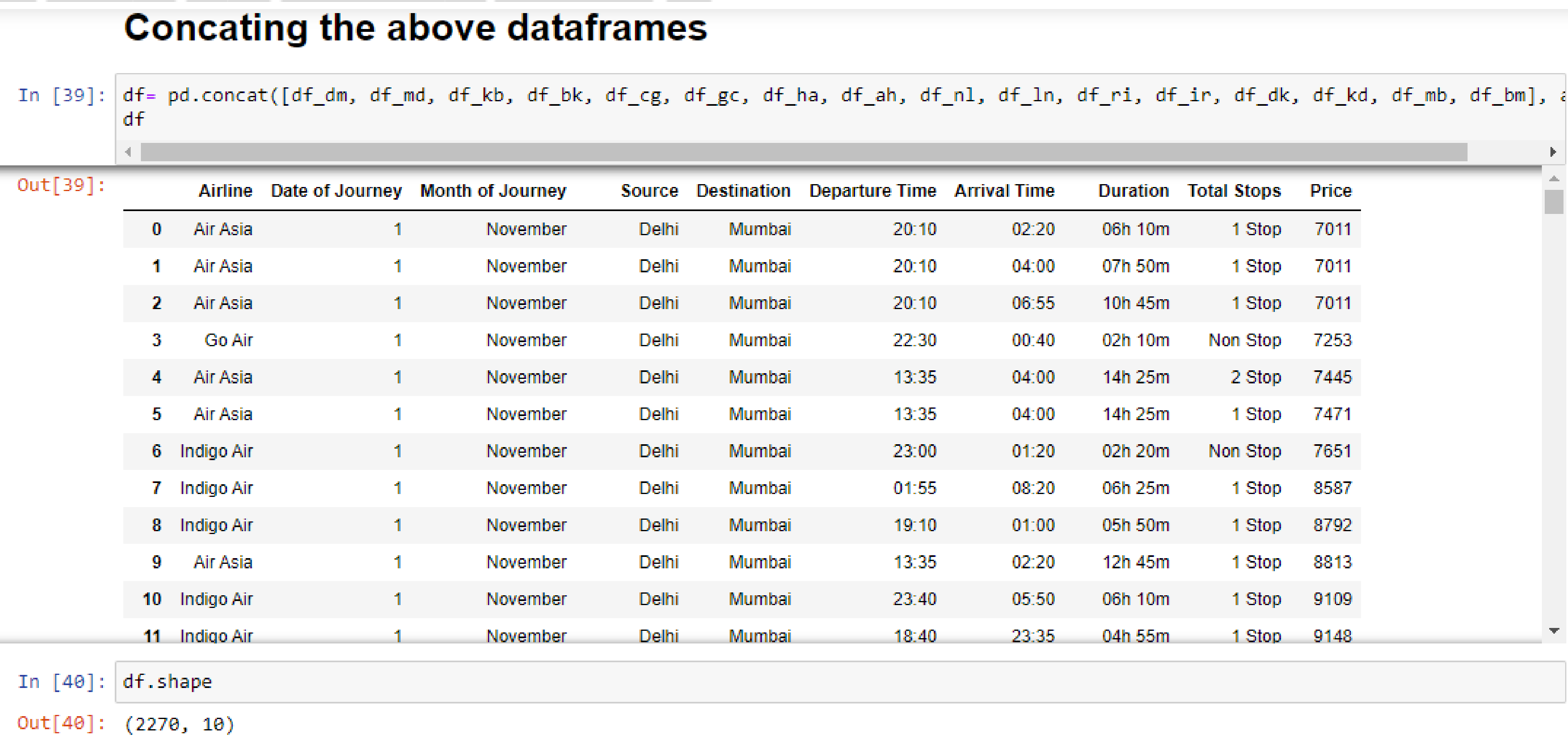
The prediction of flight ticket price is considered an unpredictable, cumbersome process. Being a data analyst, it seems to be my responsibility to add the element of prediction in every unpredictable scenario, to solve the cumbersome unsure process into a more reliable, dependent matter. Therefore, the project motivated me to go further and predict the unpredictable. Further, every project has a lot to offer as well. The project and its attributes imparted a lot of knowledge about the aviation market, its dependable and the various criteria which varies the prices of flight tickets

Data Collection

The data is scraped from 3 websites-

1. vimaansafar.com
2. yatra.com
3. makemytrip.com

2270 flight data is scraped from all the three websites.



Concating the above dataframes

```
In [39]: df= pd.concat([df_dm, df_md, df_kb, df_bk, df_cg, df_gc, df_ha, df_ah, df_nl, df_ln, df_ri, df_ir, df_dk, df_kd, df_mb, df_bm], a
         df
```

Out[39]:

	Airline	Date of Journey	Month of Journey	Source	Destination	Departure Time	Arrival Time	Duration	Total Stops	Price
0	Air Asia	1	November	Delhi	Mumbai	20:10	02:20	06h 10m	1 Stop	7011
1	Air Asia	1	November	Delhi	Mumbai	20:10	04:00	07h 50m	1 Stop	7011
2	Air Asia	1	November	Delhi	Mumbai	20:10	06:55	10h 45m	1 Stop	7011
3	Go Air	1	November	Delhi	Mumbai	22:30	00:40	02h 10m	Non Stop	7253
4	Air Asia	1	November	Delhi	Mumbai	13:35	04:00	14h 25m	2 Stop	7445
5	Air Asia	1	November	Delhi	Mumbai	13:35	04:00	14h 25m	1 Stop	7471
6	Indigo Air	1	November	Delhi	Mumbai	23:00	01:20	02h 20m	Non Stop	7651
7	Indigo Air	1	November	Delhi	Mumbai	01:55	08:20	06h 25m	1 Stop	8587
8	Indigo Air	1	November	Delhi	Mumbai	19:10	01:00	05h 50m	1 Stop	8792
9	Air Asia	1	November	Delhi	Mumbai	13:35	02:20	12h 45m	1 Stop	8813
10	Indigo Air	1	November	Delhi	Mumbai	23:40	05:50	06h 10m	1 Stop	9109
11	Indigo Air	1	November	Delhi	Mumbai	18:40	23:35	04h 55m	1 Stop	9148

```
In [40]: df.shape
Out[40]: (2270, 10)
```

EDA Steps and Visualization

- The web scraped data is saved onto an excel sheet
- The excel datasheet is extracted and saved in a dataframe
- The shape of the dataframe is checked-

  There are 2270 rows and 10 columns
- **The columns are as follows-**
  - Airline
  - Date of Journey
  - Month of Journey
  - Source
  - Destination
  - Departure Time
  - Arrival Time
  - Duration
  - Total Stops
  - Price

The data type of each column is-

- Airline - object
- Date of Journey - int64
- Month of Journey - object
- Source - object
- Destination - object
- Departure Time - object
- Arrival Time - object
- Duration - object
- Total Stops - object
- Price - object

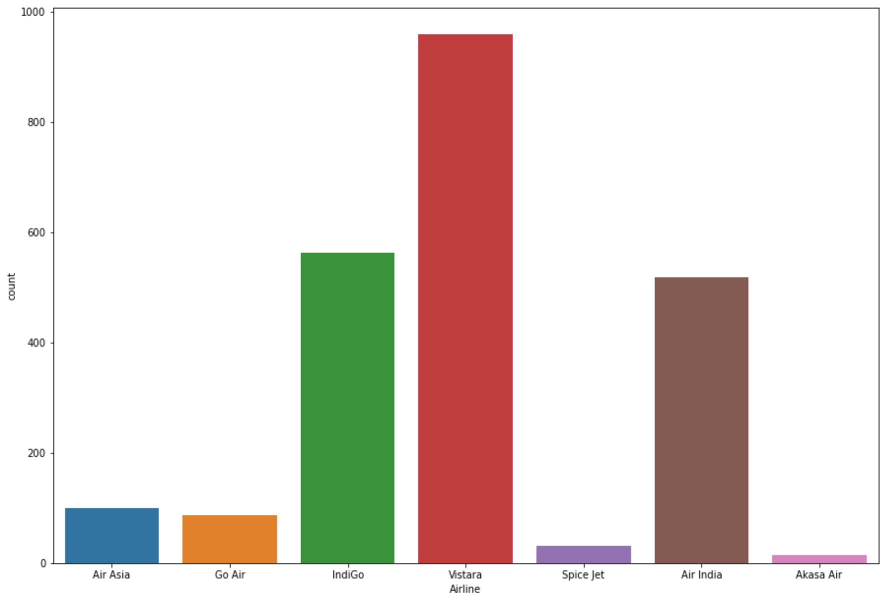
The null values are checked. The whitespaces, and dashes ('—') are replaced by null values-

- There are no null values

<u>The data visualization, value counts and encoding of object data type into numeric form for each column</u>

Airline

Replacing redundant airline names

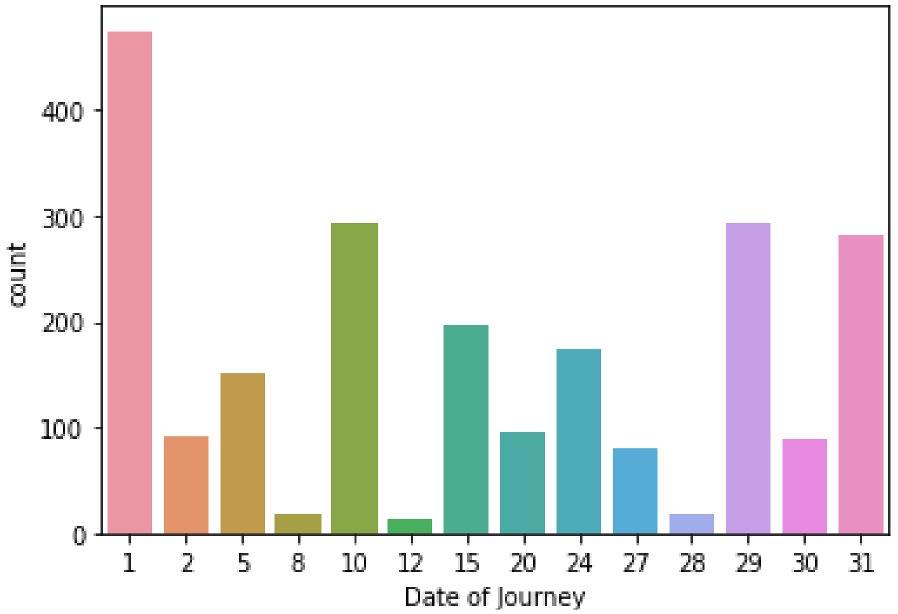


- 959 flights are Vistara flights
- 562 flights are Indigo flights
- 518 flights are Air India flights
- 100 flights are Air Asia flights
- 86 flights are Go Air flights
- 31 flights are Spice Jet flights
- Air 14 flights are Akasa Air flights

↓

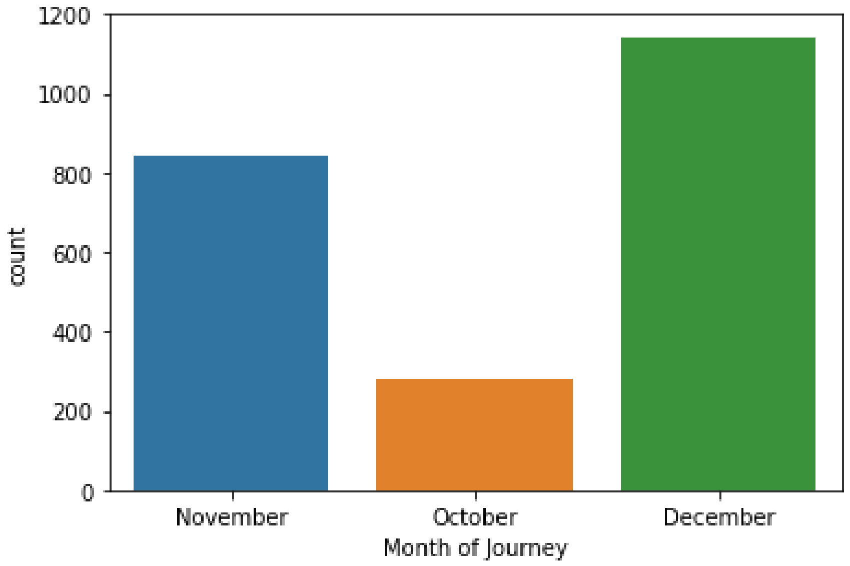
Encoding object data in numeric using Label Encoder

Date of Journey



- 475 flights depart on the 1st date of the month
- 294 flights depart on the 10th date of the month
- 293 flights depart on the 29th date of the month
- 282 flights depart on the 31st date of the month
- 197 flights depart on the 15th date of the month
- 173 flights depart on the 24th date of the month
- 151 flights depart on the 5th date of the month
- 96 flights depart on the 20th date of the month
- 2 91 flights depart on the 2nd date of the month
- 89 flights depart on the 30th date of the month
- 79 flights depart on the 27th date of the month
- 19 flights depart on the 28th date of the month
- 18 flights depart on the 8th date of the month
- 13 flights depart on the 12th date of the month

Month of Journey

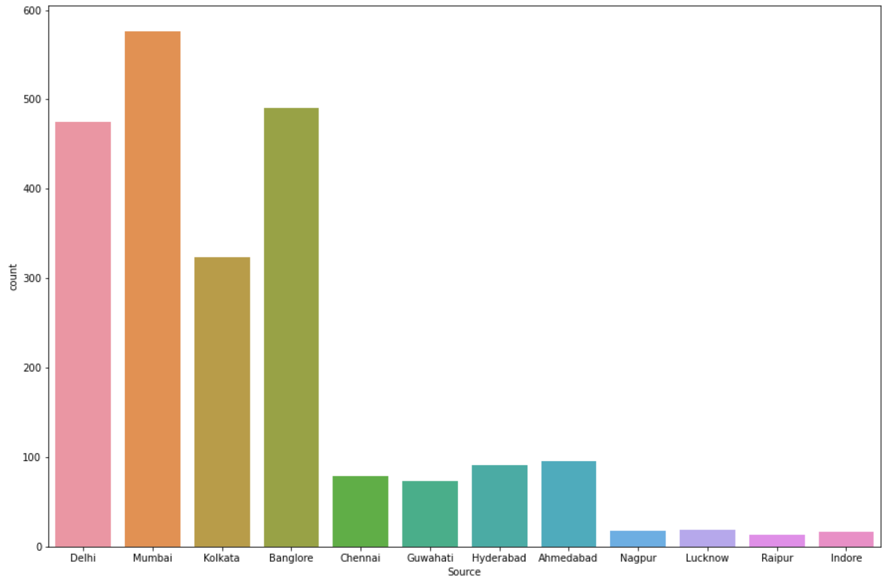


- 1143 flights depart in December
- 845 flights depart in November
- 282 flights depart in October

↓

Encoding object data in numeric using Label Encoder

Source

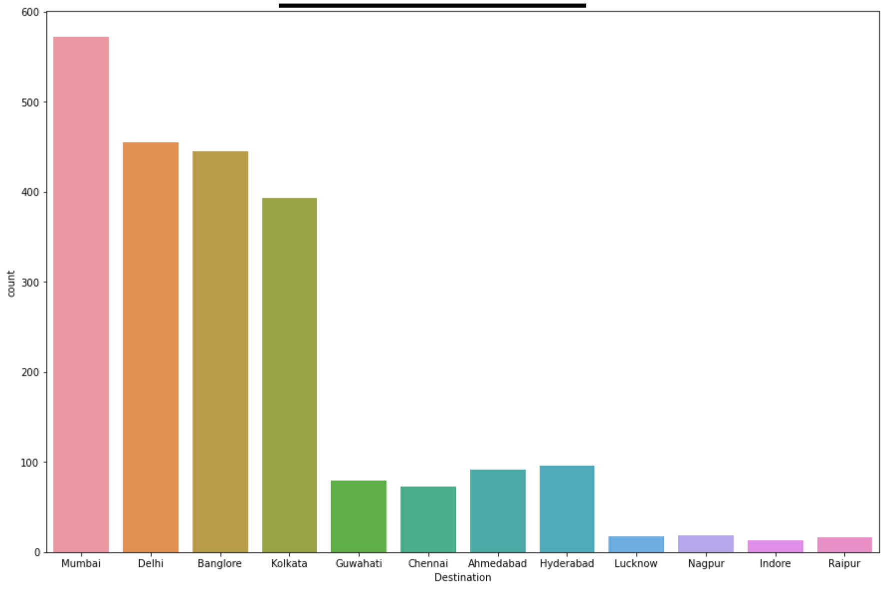


- 576 flights depart from Mumbai
- 490 flights depart from Banglore
- 475 flights depart from Delhi
- 324 flights depart from Kolkata
- 96 flights depart from Ahmedabad
- 91 flights depart from Hyderabad
- 79 flights depart from Chennai
- 73 flights depart from Guwahati
- 19 flights depart from Lucknow
- 18 flights depart from Nagpur
- 16 flights depart from Indore
- 13 flights depart from Raipur

↓

Encoding object data in numeric using Label Encoder

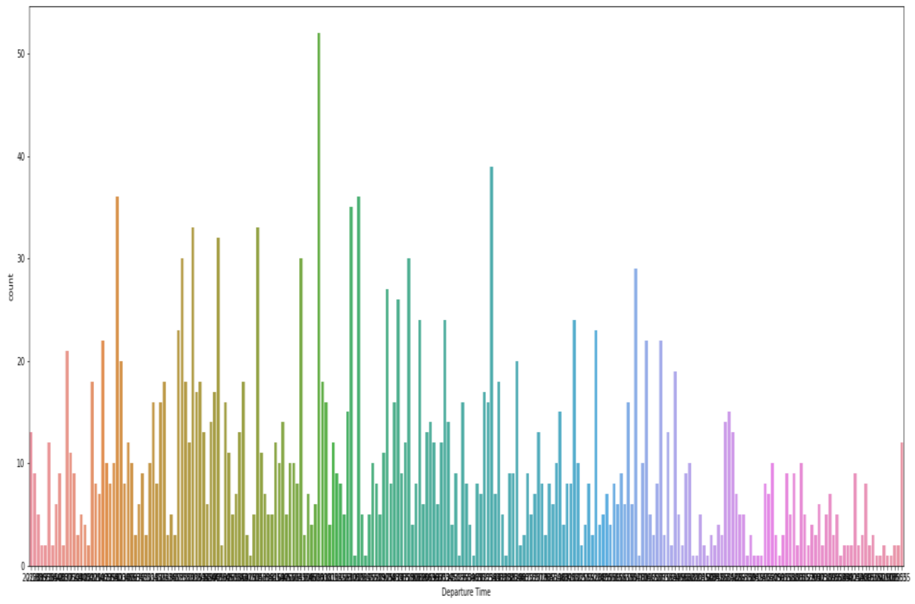
Destination



- 572 flights arrive at Mumbai
- 455 flights arrive at Delhi
- 445 flights arrive at Bangalore
- 393 flights arrive at Kolkata
- 96 flights arrive at Hyderabad
- 91 flights arrive at Ahmedabad
- 79 flights arrive at Guwahati
- 73 flights arrive at Chennai
- 19 flights arrive at Nagpur
- 18 flights arrive at Lucknow
- 16 flights arrive at Raipur
- 13 flights arrive at Indore

Encoding object data in numeric using Label Encoder

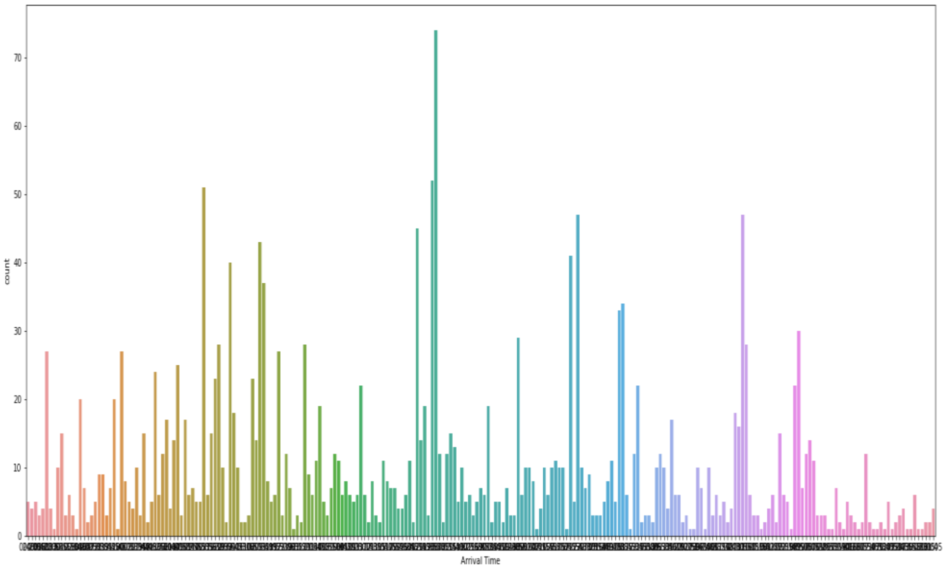
Departure Time



- Majority of the flights depart in the morning between 6.00 to 10.00

Converting time to continous number

Arrival Time

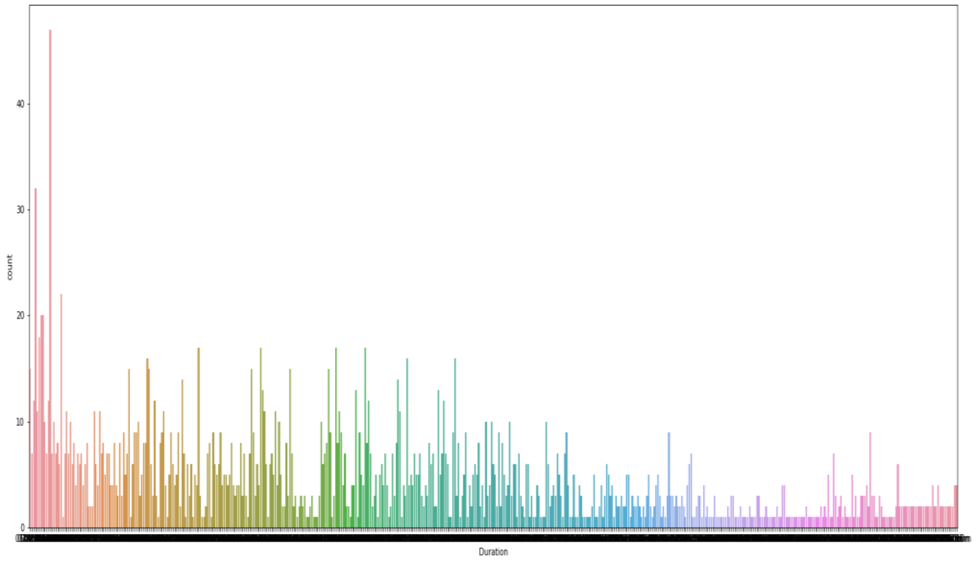


- Majority of the flights arrive in the evening between 7.00 to 12.00

↓

Converting time to continous number

Duration



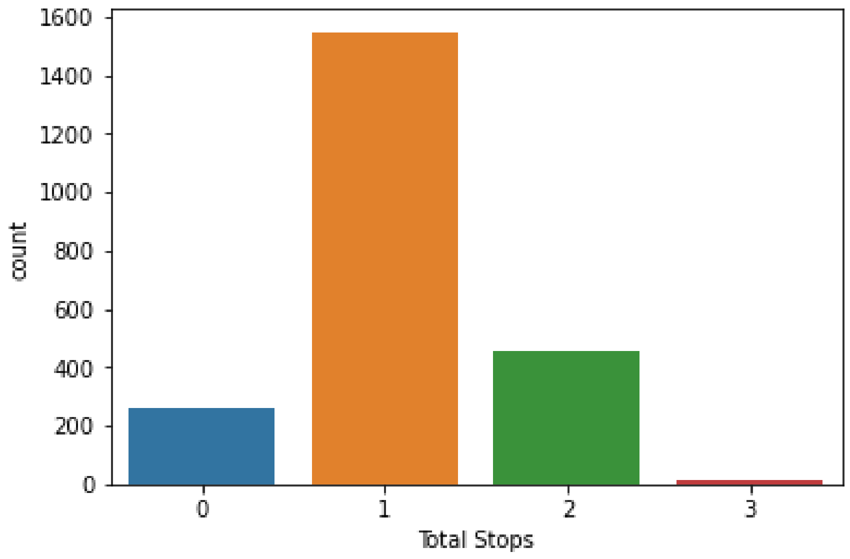
- Majority of the flights have a duration of 2-6 hours

↓

Converting time to continous number

Total Stops

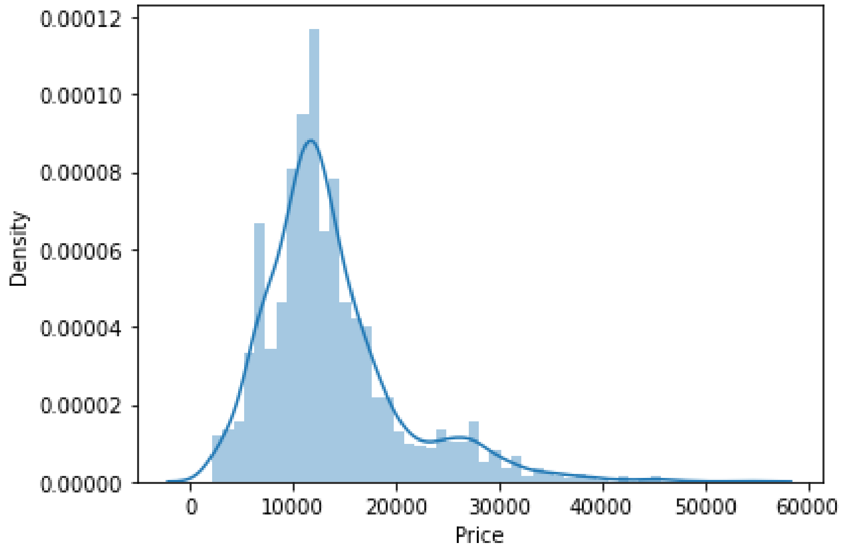
Replacing redundant stop names and converting into integer type



- 1548 flights have 1(one) stop
- 451 flights have 2(two) stops
- 259 flights are non-stop
- 12 flights have 3(three) stops

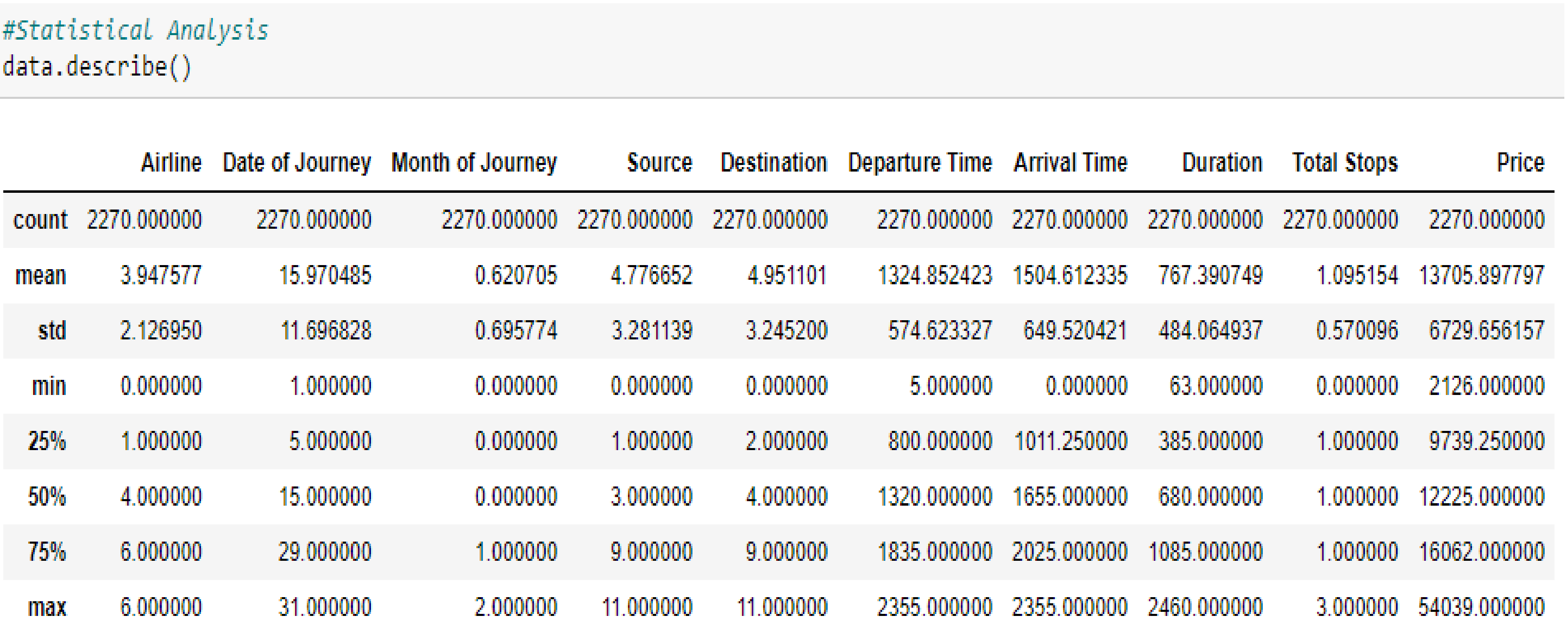
Price

Replacing ',' in price and conerting into integer type



- Majority of the flights have a price of 7000 - 14000

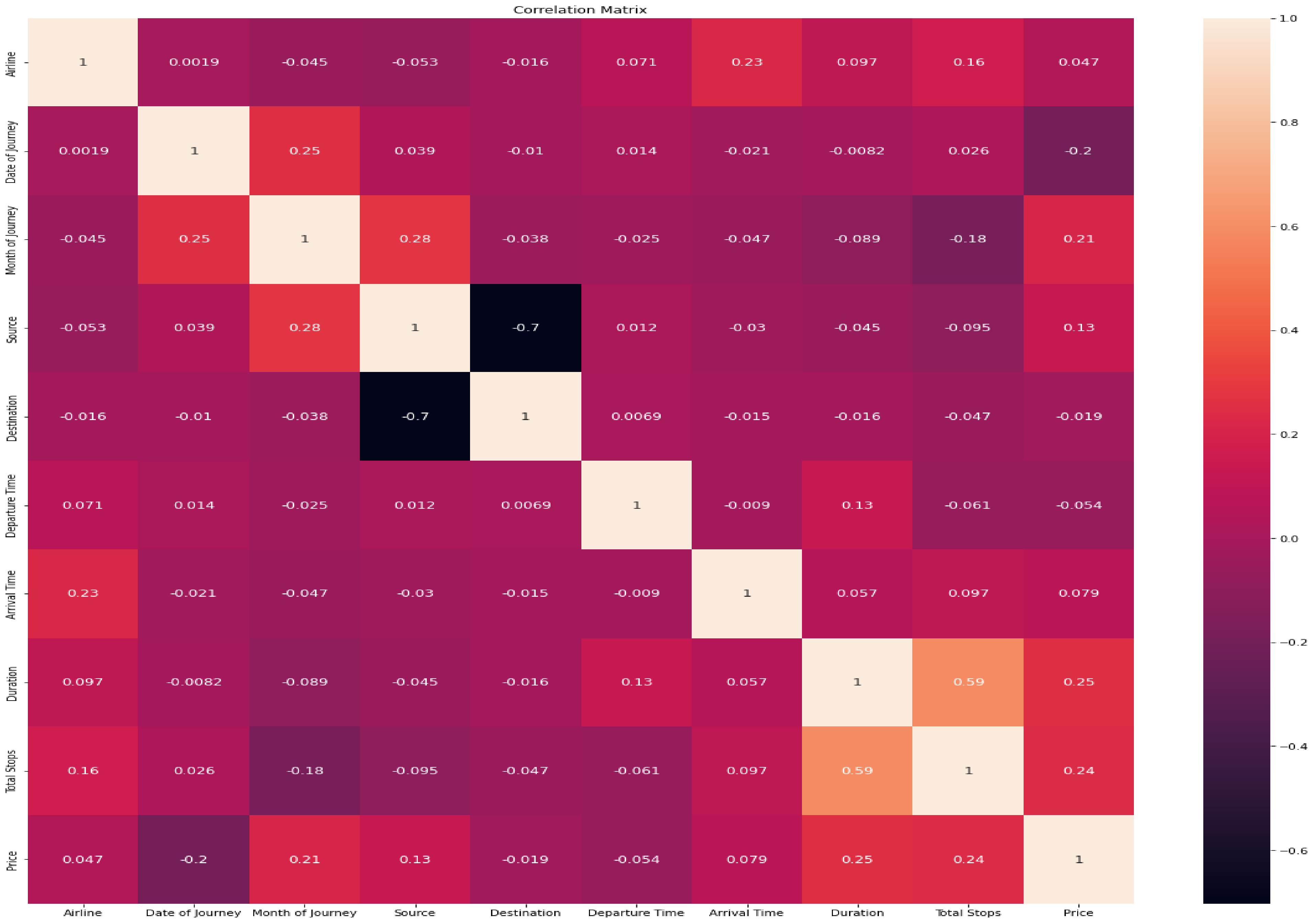
- Statistical analysis using describe method-



```
#Statistical Analysis
data.describe()
```

	Airline	Date of Journey	Month of Journey	Source	Destination	Departure Time	Arrival Time	Duration	Total Stops	Price
count	2270.000000	2270.000000	2270.000000	2270.000000	2270.000000	2270.000000	2270.000000	2270.000000	2270.000000	2270.000000
mean	3.947577	15.970485	0.620705	4.776652	4.951101	1324.852423	1504.612335	767.390749	1.095154	13705.897797
std	2.126950	11.696828	0.695774	3.281139	3.245200	574.623327	649.520421	484.064937	0.570096	6729.656157
min	0.000000	1.000000	0.000000	0.000000	0.000000	5.000000	0.000000	63.000000	0.000000	2126.000000
25%	1.000000	5.000000	0.000000	1.000000	2.000000	800.000000	1011.250000	385.000000	1.000000	9739.250000
50%	4.000000	15.000000	0.000000	3.000000	4.000000	1320.000000	1655.000000	680.000000	1.000000	12225.000000
75%	6.000000	29.000000	1.000000	9.000000	9.000000	1835.000000	2025.000000	1085.000000	1.000000	16062.000000
max	6.000000	31.000000	2.000000	11.000000	11.000000	2355.000000	2355.000000	2460.000000	3.000000	54039.000000

The Correlation Matrix using heatmap

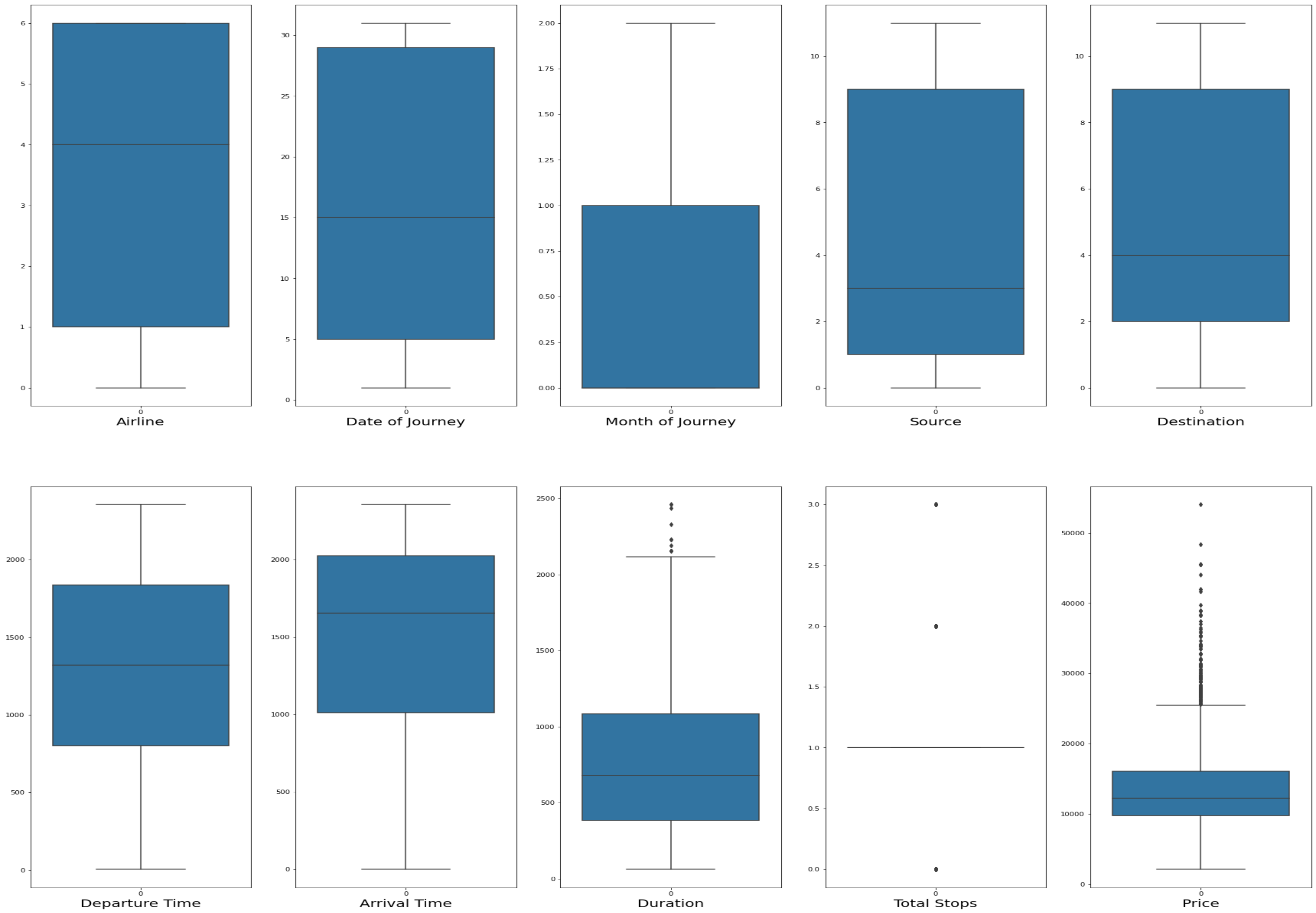


	Airline	Date of Journey	Month of Journey	Source	Destination	Departure Time	Arrival Time	Duration	Total Stops	Price
Airline	1	0.0019	-0.045	-0.053	-0.016	0.071	0.23	0.097	0.16	0.047
Date of Journey	0.0019	1	0.25	0.039	-0.01	0.014	-0.021	-0.0082	0.026	-0.2
Month of Journey	-0.045	0.25	1	0.28	-0.038	-0.025	-0.047	-0.089	-0.18	0.21
Source	-0.053	0.039	0.28	1	-0.7	0.012	-0.03	-0.045	-0.095	0.13
Destination	-0.016	-0.01	-0.038	-0.7	1	0.0069	-0.015	-0.016	-0.047	-0.019
Departure Time	0.071	0.014	-0.025	0.012	0.0069	1	-0.009	0.13	-0.061	-0.054
Arrival Time	0.23	-0.021	-0.047	-0.03	-0.015	-0.009	1	0.057	0.097	0.079
Duration	0.097	-0.0082	-0.089	-0.045	-0.016	0.13	0.057	1	0.59	0.25
Total Stops	0.16	0.026	-0.18	-0.095	-0.047	-0.061	0.097	0.59	1	0.24
Price	0.047	-0.2	0.21	0.13	-0.019	-0.054	0.079	0.25	0.24	1

- Correlation between the columns and the label ‘Price’ using corr method-
    - Duration : 0.248551
    - Total Stops : 0.235351
    - Month of Journey : 0.209234
    - Source : 0.131165
    - Arrival Time : 0.078749
    - Airline : 0.047292
    - Destination : -0.019074
    - Departure Time : -0.054397
    - Date of Journey : -0.197196

- Duration is 24.85% positively correlated to ‘Price’
- Total Stops is 23.53% positively correlated to ‘Price’
- Month of Journey is 20.92% positively correlated to ‘Price’
- Source is 13.11% positively correlated to ‘Price’
- Arrival Time is 7.87% positively correlated to ‘Price’
- Airline is 4.73% positively correlated to ‘Price’
- Destination is 1.91% negatively correlated to ‘Price’
- Departure Time is 5.43% negatively correlated to ‘Price’
- Date of Journey is 19.72% negatively correlated to ‘Price’

- Visualizing outliers using boxplot method-



- Removing outliers using zscore method-

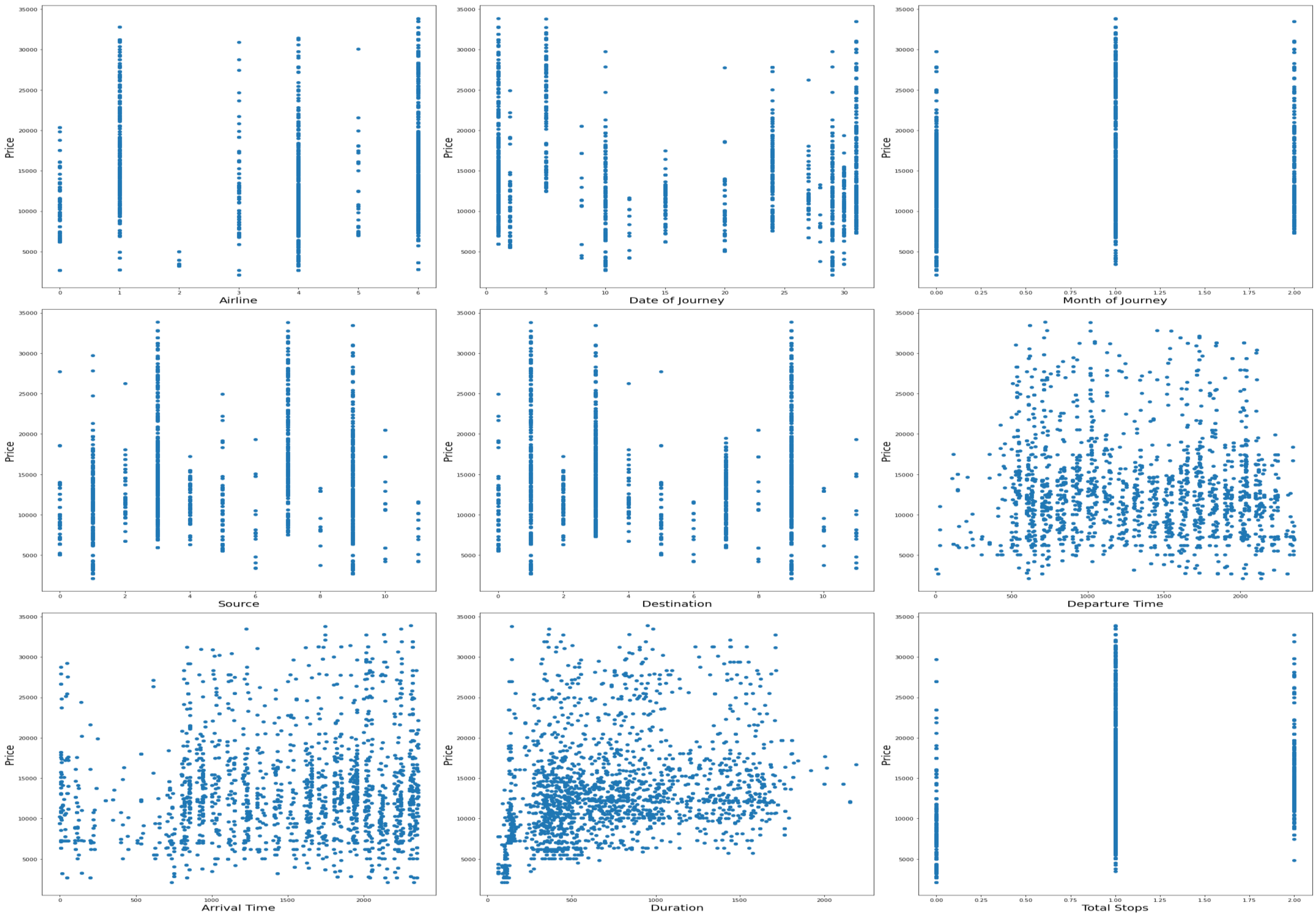
On removing the outliers the data loss is 2.3%, which is acceptable, hence outliers are removed

- The dataset is divided into x(features) and y (label)-

The x contains all the features other than the label ‘Price’

The y contains only the label ‘Price’

- Visualizing relationship between features and label-

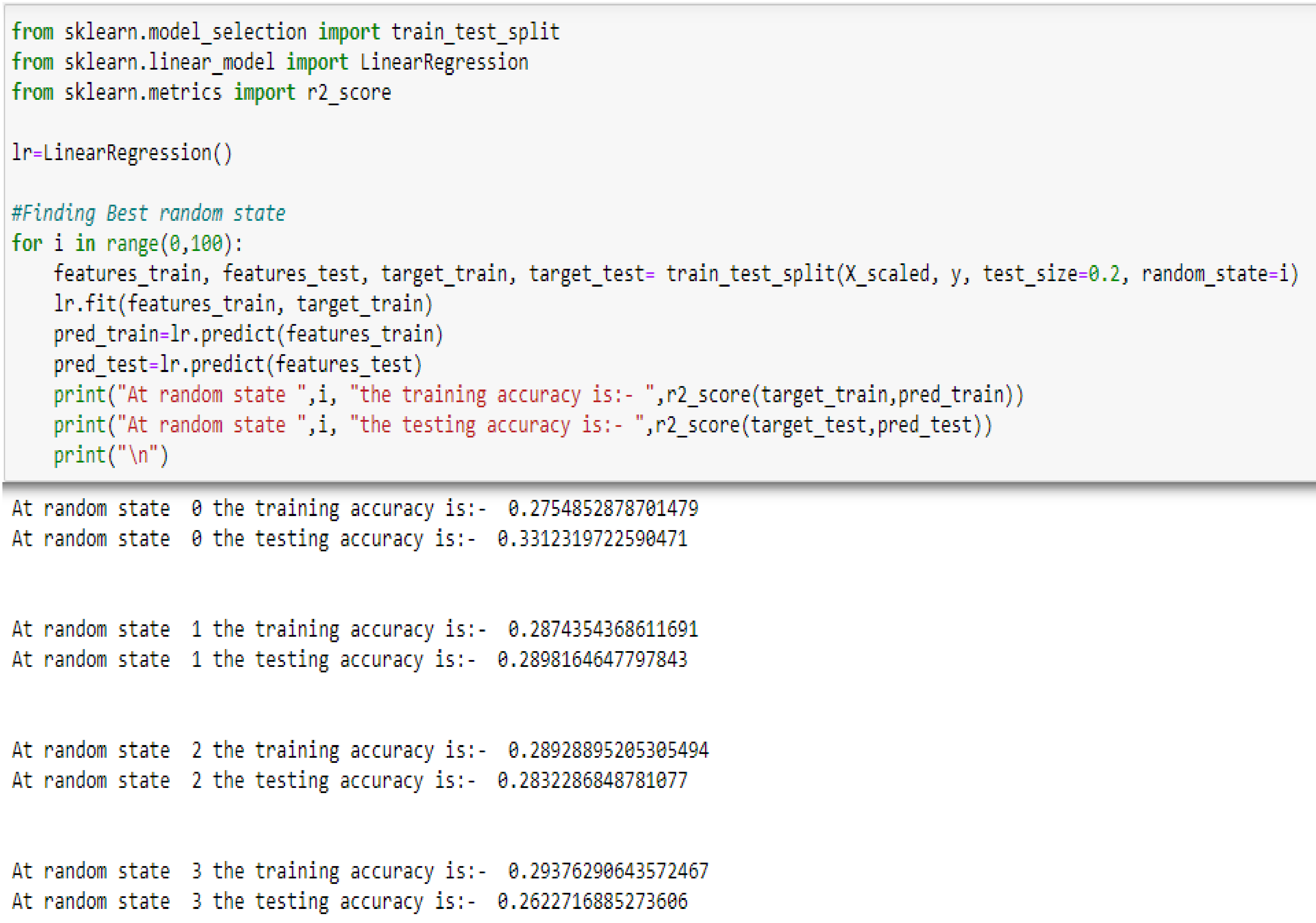


- The skewness observed in graphical analysis was confirmed by using the skew method-
    - Month of Journey : 0.693280
    - Duration : 0.548951
    - Source : 0.145235
    - Destination : 0.051229
    - Total Stops : 0.030756
    - Departure Time : -0.004913
    - Date of Journey : -0.013968
    - Airline : -0.524608
    - Arrival Time : -0.610788
- This skewness was removed using the power transformer
- The x(features) were scaled using the Standard Scaler
- The dataset was divided into train and test set using train test split and the best random state was found to be 1

Sofware Requirements-

- Jupyter Notebook – Interface for the program
- Pandas – for datafram working
- Numpy – to deal with null data
- matplotlib.pyplot – for data visualization
- Seaborn - for data visualization
- Warnings- to omit warnings
- Selenium- to scrape the flight data from different websites
- sklearn.preprocessing – to import LabelEncoder, powertransform
- Label Encoder- to encode object data to numeric data
- scipy.stats – to import zscore
- Zscore- to remove outliers
- sklearn.feature_selection- to import SelectPercentile and chi2
- SelectPercentile- to select best features
- Chi2-chi2 retrives p-values which help in filtering the best features
- power_transform- to remove the skewness in the data
- sklearn.model_selection- to import train_test_split
- train_test_split- to spit dataset into train and test samples
- sklearn.linear_model- to import LinearRegression
- LinearRegression- to use LinearRegression model
- sklearn.metrics- to import r2_score
- R2 score- to check for the efficiency of the model
- sklearn.model_selection- to import cross_val_score
- cross_val_score- to check for overfitting and underfitting
- sklearn.model_selection- to import GridSearchCV
- GridSearchCV to enhance the working of the model by manipulating the parameters
- from sklearn.linear_model-to import Lasso
- Lasso- to regularize the linear regression model
- sklearn.ensemble – to import RandomForestRegressor, AdaBoostRegressor
- RandomForestRegressor- to use Random Forest Regressor model
- AdaBoostRegressor- to use AdaBoostRegressor

Train Test Split



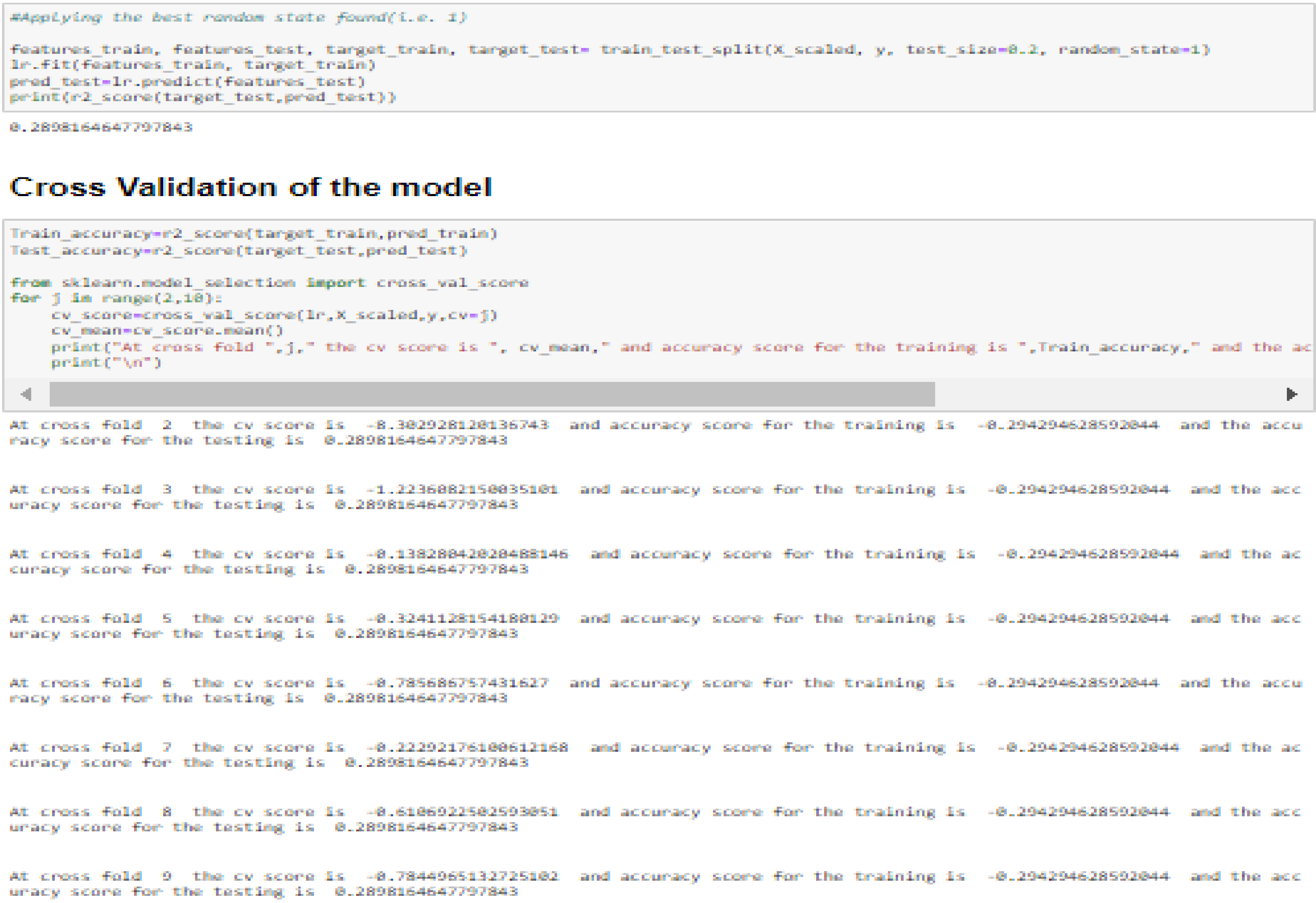
```python
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score

lr=LinearRegression()

#Finding Best random state
for i in range(0,100):
    features_train, features_test, target_train, target_test= train_test_split(X_scaled, y, test_size=0.2, random_state=i)
    lr.fit(features_train, target_train)
    pred_train=lr.predict(features_train)
    pred_test=lr.predict(features_test)
    print("At random state ",i, "the training accuracy is:- ",r2_score(target_train,pred_train))
    print("At random state ",i, "the testing accuracy is:- ",r2_score(target_test,pred_test))
    print("\n")
```

```
At random state  0 the training accuracy is:-  0.2754852878701479
At random state  0 the testing accuracy is:-  0.3312319722590471


At random state  1 the training accuracy is:-  0.2874354368611691
At random state  1 the testing accuracy is:-  0.2898164647797843


At random state  2 the training accuracy is:-  0.28928895205305494
At random state  2 the testing accuracy is:-  0.2832286848781077


At random state  3 the training accuracy is:-  0.29376290643572467
At random state  3 the testing accuracy is:-  0.2622716885273606
```

Model/s Development and Evaluation

The train and test data were applied on different models as follows

Linear Regression Model



```
#Applying the best random state found(i.e. 1)

features_train, features_test, target_train, target_test= train_test_split(X_scaled, y, test_size=0.2, random_state=1)
lr.fit(features_train, target_train)
pred_test=lr.predict(features_test)
print(r2_score(target_test,pred_test))
```

```
0.2898164647797843
```

Cross Validation of the model

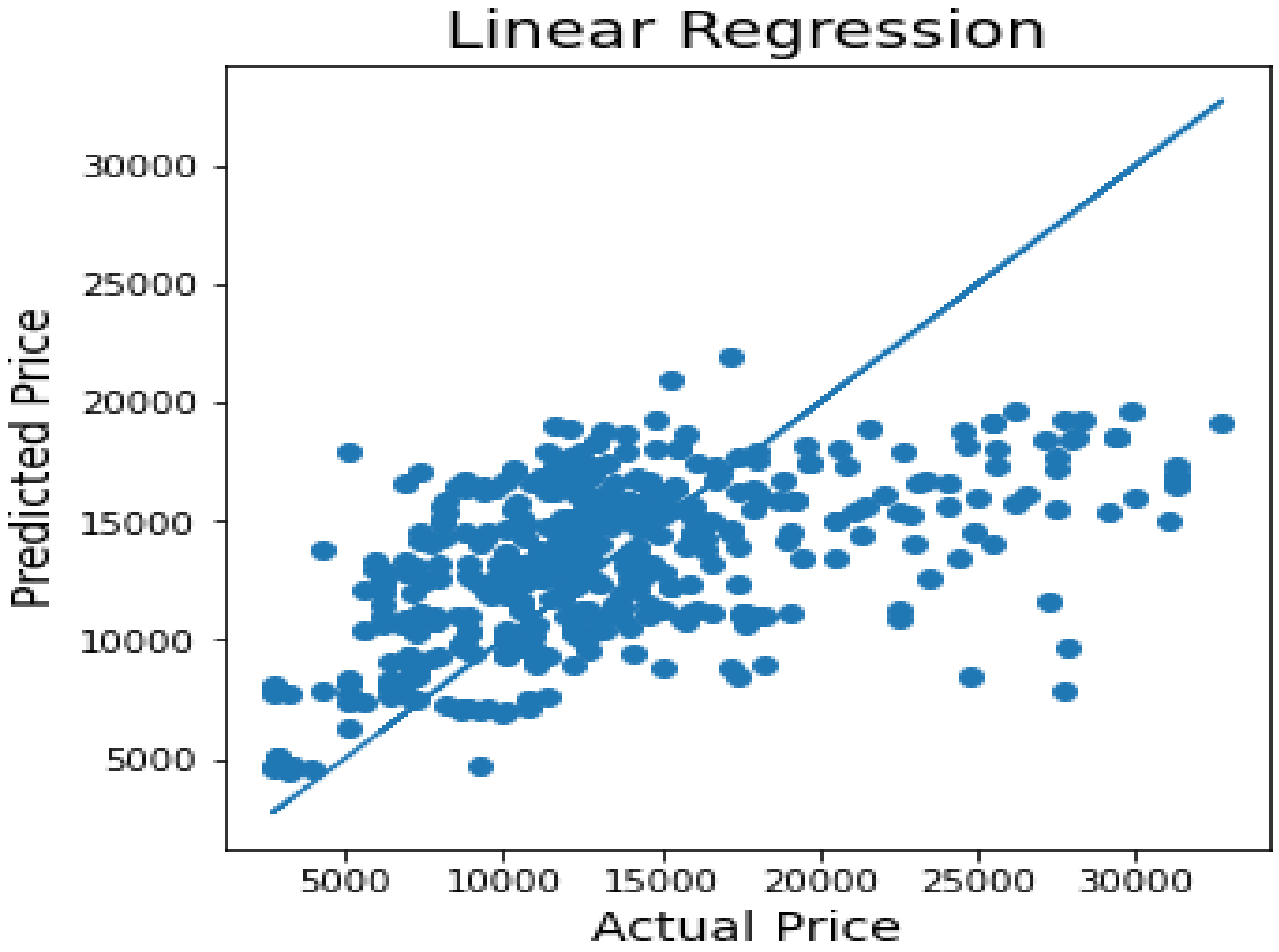
```
Train_accuracy=r2_score(target_train,pred_train)
Test_accuracy=r2_score(target_test,pred_test)

from sklearn.model_selection import cross_val_score
for j in range(2,10):
    cv_score=cross_val_score(lr,X_scaled,y,cv=j)
    cv_mean=cv_score.mean()
    print("At cross fold ",j," the cv score is ", cv_mean," and accuracy score for the training is ",Train_accuracy," and the ac
    print("\n")
```

```
At cross fold  2  the cv score is  -8.302928120136743  and accuracy score for the training is  -0.2942946285920044  and the accu
racy score for the testing is  0.2898164647797843


At cross fold  3  the cv score is  -1.2236082150835101  and accuracy score for the training is  -0.2942946285920044  and the acc
uracy score for the testing is  0.2898164647797843


At cross fold  4  the cv score is  -0.13828842028488146  and accuracy score for the training is  -0.2942946285920044  and the ac
curacy score for the testing is  0.2898164647797843


At cross fold  5  the cv score is  -0.3241128154188129  and accuracy score for the training is  -0.2942946285920044  and the acc
uracy score for the testing is  0.2898164647797843


At cross fold  6  the cv score is  -0.785686757431627  and accuracy score for the training is  -0.2942946285920044  and the accu
racy score for the testing is  0.2898164647797843


At cross fold  7  the cv score is  -0.22292176108612168  and accuracy score for the training is  -0.2942946285920044  and the ac
curacy score for the testing is  0.2898164647797843


At cross fold  8  the cv score is  -0.6106922502593051  and accuracy score for the training is  -0.2942946285920044  and the acc
uracy score for the testing is  0.2898164647797843


At cross fold  9  the cv score is  -0.7844965132725102  and accuracy score for the training is  -0.2942946285920044  and the acc
uracy score for the testing is  0.2898164647797843
```

- The **R2 score** for target test and pred_test(data predicted on features_test) is **28.98%**

- Upon cross-validation it was observed that the number of folds did not have such impact on the accuracy and cv score. So cv=9 is selected. Here we have handled the problem of the overfitting and the underfitting by checking the training and testing score

- The graph between Actual Price and Predicted Price depicts the best fit line which passes through maximum of the points, hence suggesting that the model works well-


Linear Regression
Predicted Price
Actual Price
30000
25000
20000
15000
10000
5000
5000
10000
15000
20000
25000
30000

Regularization

```
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import Lasso
parameters={'alpha':[.0001, .001, .01,.1, 1, 10], 'random_state':list(range(0,10))}
ls=Lasso()
clf=GridSearchCV(ls,parameters)
clf.fit(features_train,target_train)
print(clf.best_params_)
```

```
{'alpha': 10, 'random_state': 0}
```

```
ls=Lasso(alpha=10, random_state=0)
ls.fit(features_train,target_train)
ls.score(features_train, target_train)
pred_ls=ls.predict(features_test)

lss=r2_score(target_test,pred_ls)
lss
```

```
0.290501272908951
```

```
cv_score=cross_val_score(ls,X_scaled,y,cv=4)
cv_mean=cv_score.mean()
cv_mean
```

```
-0.1312253501136659
```

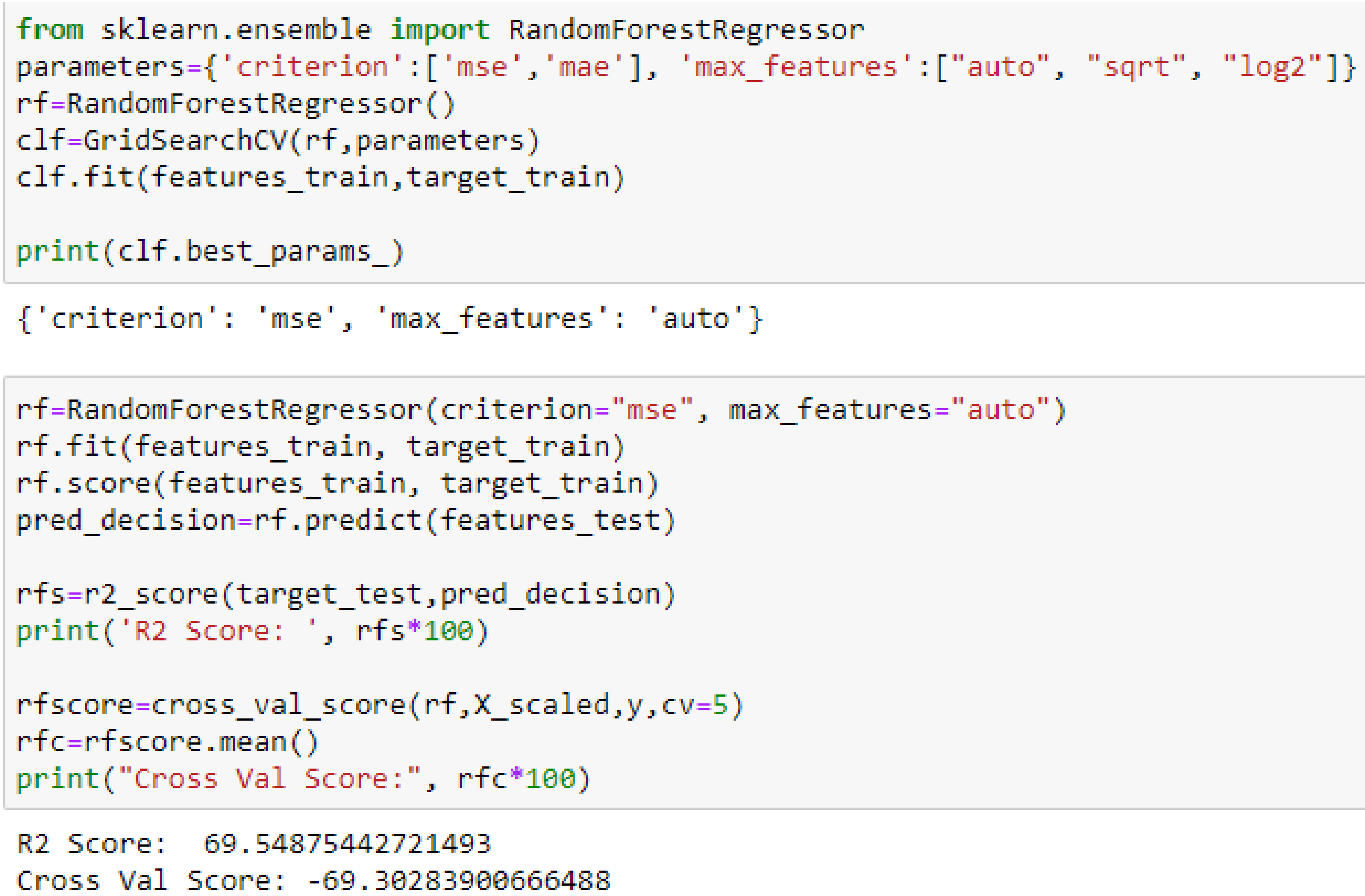
- The Linear Regression Model is Lasso regularized with the aid of GridSearchCV

- The best parameters for alpha and random_state are found as follows-
    - **alpha: 10**
    - **random_state: 0**

- Applying the above found best parameters on Lasso regularized Linear Regression Model, the following was obtained-

    - **R2 score for target_test and pred_test(data predicted on features_test) : 29.05%**
    - **CV score : -13.12%**

Random Forest Regressor Model



```
from sklearn.ensemble import RandomForestRegressor
parameters={'criterion':['mse','mae'], 'max_features':["auto", "sqrt", "log2"]}
rf=RandomForestRegressor()
clf=GridSearchCV(rf,parameters)
clf.fit(features_train,target_train)

print(clf.best_params_)
```

```
{'criterion': 'mse', 'max_features': 'auto'}
```

```
rf=RandomForestRegressor(criterion="mse", max_features="auto")
rf.fit(features_train, target_train)
rf.score(features_train, target_train)
pred_decision=rf.predict(features_test)

rfs=r2_score(target_test,pred_decision)
print('R2 Score: ', rfs*100)

rfscore=cross_val_score(rf,X_scaled,y,cv=5)
rfc=rfscore.mean()
print("Cross Val Score:", rfc*100)
```

```
R2 Score:  69.54875442721493
Cross Val Score: -69.30283900666488
```

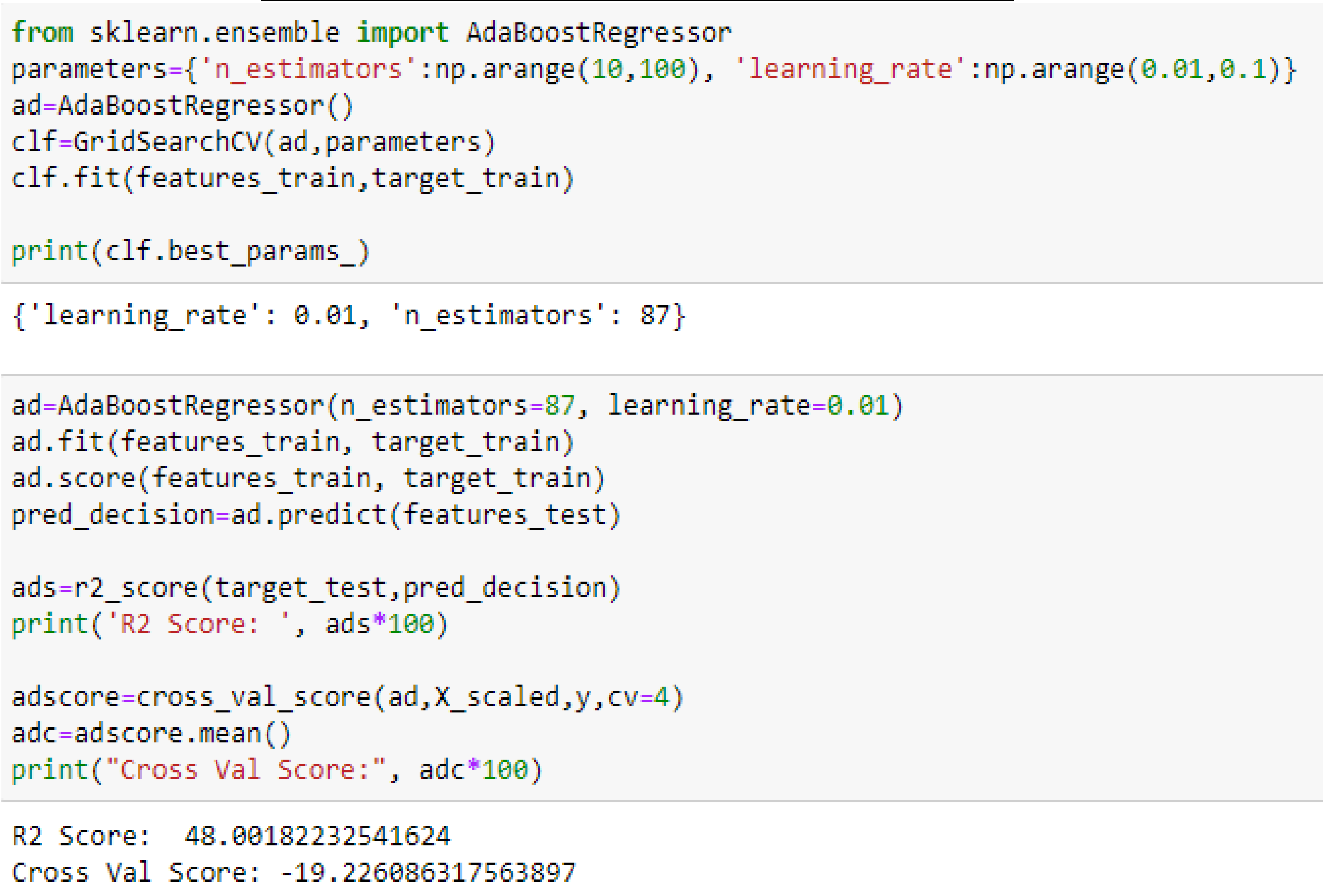
- Random Forest Regressor Model is hyperparameter tuned using GridSearchCV

- The best parameters for criterion and max_features are found as follows-
  - **criterion: mse**
  - **max_features: auto**

- Applying the above found best parameters on Random Forest Regressor Model, the following was obtained-

  - **R2 score for target_test and pred_test (data predicted on features_test) : 69.55%**
  - **CV score : -69.30%**

Ada Boost Regressor Model



```python
from sklearn.ensemble import AdaBoostRegressor
parameters={'n_estimators':np.arange(10,100), 'learning_rate':np.arange(0.01,0.1)}
ad=AdaBoostRegressor()
clf=GridSearchCV(ad,parameters)
clf.fit(features_train,target_train)

print(clf.best_params_)
```

```
{'learning_rate': 0.01, 'n_estimators': 87}
```

```python
ad=AdaBoostRegressor(n_estimators=87, learning_rate=0.01)
ad.fit(features_train, target_train)
ad.score(features_train, target_train)
pred_decision=ad.predict(features_test)

ads=r2_score(target_test,pred_decision)
print('R2 Score: ', ads*100)

adscore=cross_val_score(ad,X_scaled,y,cv=4)
adc=adscore.mean()
print("Cross Val Score:", adc*100)
```

```
R2 Score:  48.00182232541624
Cross Val Score: -19.22608631756389 7
```

- Ada Boost Regressor Model is hyperparameter tuned using GridSearchCV

- The best parameters for n_estimators and learning_rate are found as follows-
    - **learning_rate: 0.01**
    - **n_estimators: 87**

- Applying the above found best parameters on Ada Boost Regressor Model, the following was obtained-

    - **R2 score for target_test and pred_test(data predicted on features_test) : 48%**
    - **CV score : 19.22%**

Model	R2 score for target_test and pred_test (data predicted on features_test)	CV score
Linear Regression Model	29.05%	-13.12%
Random Forest Regressor Model	69.55%	-69.30%
AdaBoost Regressor Model	48%	19.22%

- The R2 score of Random Forest Regressor is 69.55% and CV score of Random Forest Regressor is -69.30%
- This is the best working model and is finalized
- This model can be used to predict the price of the flights

CONCLUSION

- It is essential to scrape diverse data to allow the model to be applicable for versatile inputs

- The EDA analysis of the data is essential as it helps to understand the relationship between the target and features as well as omit out unwanted columns, thereby taking care of overfitting scenario

- It is necessary to have the data encoded properly in order to avoid mis-interpretation of data

- The models should be used properly, as their regularization /hyperparamter tuning is highly advisable for the best outcome.

- The project imparted key knowledge about the aviation market and how beneficial it is to estimate a proper price of the flight ticket in order to enahnce one's own benefit.

- The limitation of the solution is that it predicts the price of the flight ticket but does not classify if it is on sale or if the ticket offers any discount or benefits. The limitation can be handled, that post the regression model, the findings can be subjected to further classification and regression models to ascertain if the ticket offers any discount or benefits.