Flight Price Prediction Project

Submitted by:

Alivia Dasgupta

INTRODUCTION

- ## Business Problem Framing

  The tourism industry is changing fast and this is attracting a lot more travellers each year. The airline industry is considered as one of the most sophisticated industry in using complex pricing strategies. Nowadays flight prices are quite unpredictable. The ticket prices change frequently. Customers are seeking to get the lowest prices for their ticket, while airline companies are trying to keep their overall revenue as high as possible. With the use of technology it is actually possible to reduce the uncertainty of flight prices.
  Hence, here we will be predicting the flight prices using different machine learning algorithms.

  Anyone who has booked the flight tickets 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)

- ## Conceptual Background of the Domain Problem

  Flight prices are something unpredictable. It’s more than likely, that we spend hours on the internet searching for good deals. So, airline companies use complex algorithms to calculate flight prices given various conditions present at the particular time. Nowadays, the number of people using flights are increasing significantly. It gets difficult for the airline companies to maintain prices as they change dynamically due to various conditions. For that, we will use machine learning algorithms which can help to solve these problems. This can help airline companies to maintain their prices and also help the customers to predict future flight prices and accordingly plan their journey well in advance.

- ## Review of Literature

It is hard for the client to buy an air ticket at the most reduced cost. The majority of systems are using the modern computerized techniques known as Machine Learning. I have scrapped the data from official online sites and based on that data, did analysis based on which, feature prices are changing and accordingly checked the relationship of flight prices with all the features.

- ## Motivation for the Problem Undertaken

This project helps tourist to find the best flight prices based on their needs and also provides various options and flexibility for travelling. Different features like airline name, source, destination, departure time, arrival time, duration, total stops and date of journey help in understanding and predicting the flight price variations. As per the client requirements, I have worked on this and followed all the steps till model deployment.

Analytical Problem Framing

- Mathematical/ Analytical Modelling of the Problem

In our scrapped dataset, our target variable "Price" is a continuous variable. Therefore, we will be handling this modelling problem as regression.
This project is done in two parts:

Data Collection phase

In this section I scraped the data of Flights from easemytrip.com website where, I have fetched data for different locations and different dates. The features which I scraped are Airline Name, Date of journey, Source, Destination, Departure time, Arrival Time, Duration, No of Stops, and at last target variable Price of the flights.

Model Building phase:

After collecting the data, we need to build a machine learning model. Before model building, will do all data pre-processing steps. Try different models with different hyper parameters and the select the best model. Will include all the below steps mentioned:

1. Data Cleaning
2. Exploratory Data Analysis (EDA)
3. Data Pre-processing and Visualisation
4. Model Building
5. Model Evaluation
6. Selecting the best model

Data Sources and their formats

The dataset is in the form of CSV (Comma Separated Value) format and consists of 10 columns (9features and 1 label) with 1794 number of records.

- Airline Name - This shows the names of the airlines.
- Date of Journey - Gives us the information about the journey date.
- Source – Gives us the information about from where the flight will start(location).
- Destination – Gives us the information about where the flight will land.
- Stops – Shows the number of stops.
- Duration – Shows how much time the flight takes to reach the destination.
- Departure time – Shows the time when the flight will take off from the source location.
- Arrival Time – Shows the time when the flight will reach the destination.
- Price - Lists the price of the flights.

We can see our dataset includes a target label "Price" column and the remaining feature columns can be used to determine or help in predicting the price of the flights.

Data Pre-processing Done

1. Importing the necessary dependencies and libraries.
2. Reading the CSV file and converted into data frame.
3. Checking the data dimensions for the original dataset.
4. Looking for null values and accordingly fill the missing data.
5. Checking the summary of the dataset.
6. Checking unique values.
7. Checking all the categorical columns in the dataset
8. Checking for multi collinearity using VIF.
9. Performed Feature Importance using ExtraTrees Regression

- Data Inputs- Logic- Output Relationships

The input data were all object type , so had to clean the data by initializing the prize column and converting the same into float type and ensuring all the categorical features are converted to numeric form with the help of LabelEncoder Method. Since most of the features were of categorical type, we did not have to worry much about skewness and outliers.

- Hardware and Software Requirements and Tools Used

Hardware technology being used.

RAM : 16 GB

CPU : 11th Gen Intel(R) Core(TM) i5-1135G7@ 2.40GHz 2.42 GHz

GPU : intel Iris Graphics

Software technology being used.

Programming language	: Python
Distribution	: Anaconda Navigator
Browser based language shell	: Jupyter Notebook

Libraries/Packages specifically being used.

Pandas, NumPy, matplotlib, seaborn, scikit-learn

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)
  1. Clean the dataset from unwanted scraped details.
  2. Impute missing values with meaningful information.
  3. Encoding the categorical data to get numerical input data.
  4. Compare different models and identify the suitable model.
  5. R2 score is used as the primary evaluation metric.
  6. MSE and RMSE are used as secondary metrics.
  7. Cross Validation Score was used to ensure there are no overfitting or underfitting models.

- Testing of Identified Approaches (Algorithms)

  All the regression machine learning algorithms used are:

  - Linear Regression Model
  - Random Forest Regression Model
  - Extra Trees Regression Model
  - KNN Regression
  - Decision Tree Regression Model

- Run and Evaluate selected models

  I have used various algorithms for predicting our label like Linear Regression, KNeighbors Regression, Decision Tree Regression, Random Forest Regression, ExtraTreesRegressor, For evaluating the model, I have used Mean Squared Error (MSE), Mean Absolute Error (MAE), training score, testing score and root mean squared root (RMSE).

```
from sklearn.linear_model import LinearRegression

lr=LinearRegression()
lr.fit(X_train,y_train)
print(lr.score(X_train,y_train))
lr_predict=lr.predict(X_test)
```

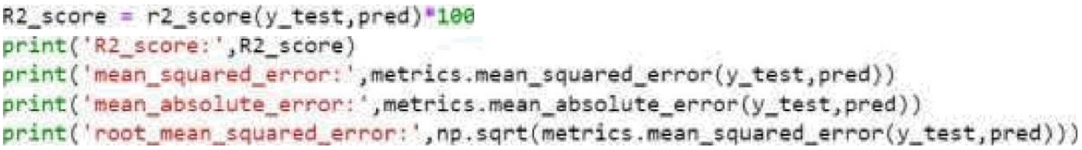
```
0.3007706710369302
```

```
from sklearn.metrics import mean_absolute_error
print('MSE: ',mean_squared_error(lr_predict,y_test))
print('MAE: ',mean_absolute_error(lr_predict,y_test))
print('r2_score:',r2_score(lr_predict,y_test))
```

```
ySE :  12658517 . 585033586
yAE :   2583. 3997168568367
r2_s core : -1.2066481619426543
```

1. Random Forest I4egressor

```
froo sk1aar'n . nsembla 1Inpor t Randonror estn a guess o/
Jrom sk i aar'n 1nporl met r ! C 4
RTR =Ra nd oMFOf“' S7kEgF• S S 0 F (}
CFR.fit(K traim,y tram)
pred=RFR.predltt(x testl
```



```
R2_score = r2_score(y_test,pred)*100
print('R2_score:',R2_score)
print('mean_squared_error:',metrics.mean_squared_error(y_test,pred))
print('mean_absolute_error:',metrics.mean_absolute_error(y_test,pred))
print('root_mean_squared_error:',np.sqrt(metrics.mean_squared_error(y_test,pred)))
```

```
R2_sTo-s- f4-37t375°9675ñ06
mean squared error:  6523296.6046355795
me n_ubsclute_error. 1671.A4166E0288258
root mean sRuared ez sr. 255d.07451a3922827
```

2. EXtra Treos Regrec6or

```
fros sklearn.anse«ola 1M›os’t  GctrazreesRegraesor
£TR- 2 xtr aTrees4agr assor ()
ETR.fit(X_train,y_train)
pred=ETR.predict(X_test)
prLrfi ( ' 2_scére : ',r 2_score(y_test,pred) )
prl nlı ’eean_s   ared_wrar : ' , uetrLcs.eeen  squared_orror(y test,predj)”
ρlrtt (*e ano_a¥no1ut e_er roi• : ' , netr1cs .nagn  absolute  erpor(y  te4c  pred) )
prl ot ( ' r      _ aao_sgugz-ad_wror: * , ap. qrt (zstr1cs.uean_ quarad_error{y_tesc,pr'ed ))}
```

```
Icz_score: a.se”ac1syyssea2ssa
aean_squered_error: 7t69773.G53874486
uean_abso1ute_eroor'• 1581. A2553S71d2g54
t•oot noan 9zjuerad ar'ror : 2677 64330223g889S
```

3. Gradient Boosting Regressor

```
]: from sklearn.ensemble import GradientBoostingRegressor
```

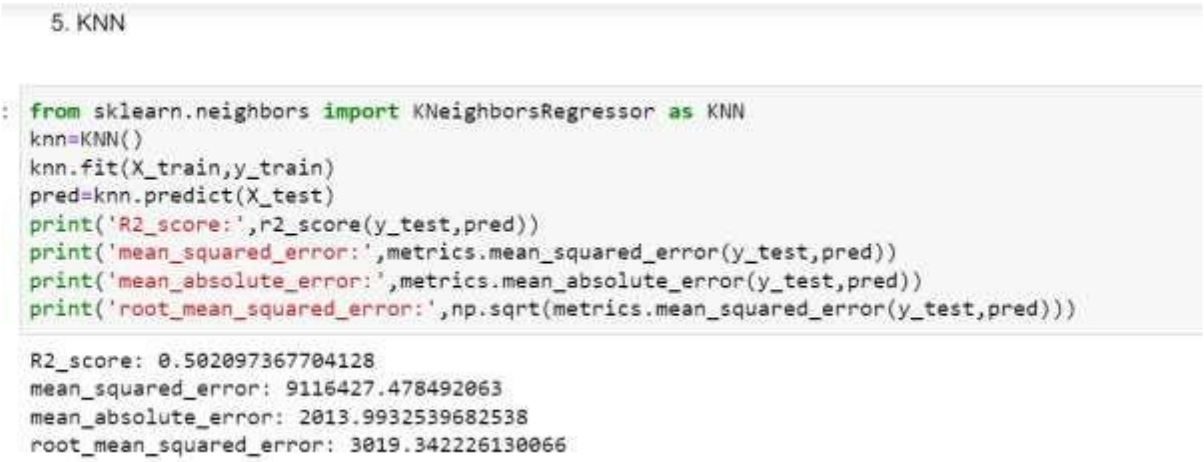
```
p/•1ut{ ' kz scar‘a: ' , z•2_¿ cooe {y_ta st pred) )
pr  zzt( '       _az{unr'eâ._aa r•or: ' .ea tf•1cs.etaan_squarad_cr”roc'{y_C«st,ppad) )
pr1nt( '      n agao1uca_arnor: ' , eetr”Scs.eaan_absolut• 4r r•on{y_tcst,pfiad›)
priot{ ' rzxzt        _squar•ad_•rr'or: ' .np.sqrt(”aa€r1cs .aaan_sqvared wror'{y test. pra4}})
```

```
k2 s<oro. 6. 5988763752565782

uean_abso1ute_error : z029.9S76g960I709
ooot eaan_xqua rad_arpor: 2716. 6I2azSzS27176
```

4, oaetaan Trea R•gr•••or

```
[:  ft       sklaam ..waa 1■porc Oec Ss1onTreaR•gr acsor
    DTR•DacJs1onTreeRegr’essor( )
    DTR.fit(X_train,y_train)
    pred=DTR.predict(X_test)
    print('R2_score:',r2_score(y_test,pred))
    print('mean_squared_error:',metrics.mean_squared_error(y_test,pred))
    print('mean_absolute_error:',metrics.mean_absolute_error(y_test,pred))
    print('root_mean_squared_error:',np.sqrt(metrics.mean_squared_error(y_test,pred)))
```

```
mean_squared_error: 10577375.662533067
      x6s«1vt«_er wr. 1s1z. /s2sssyse1szs
root eean_squared_ eroor . s252. 2B775B”Z6eBg1
```

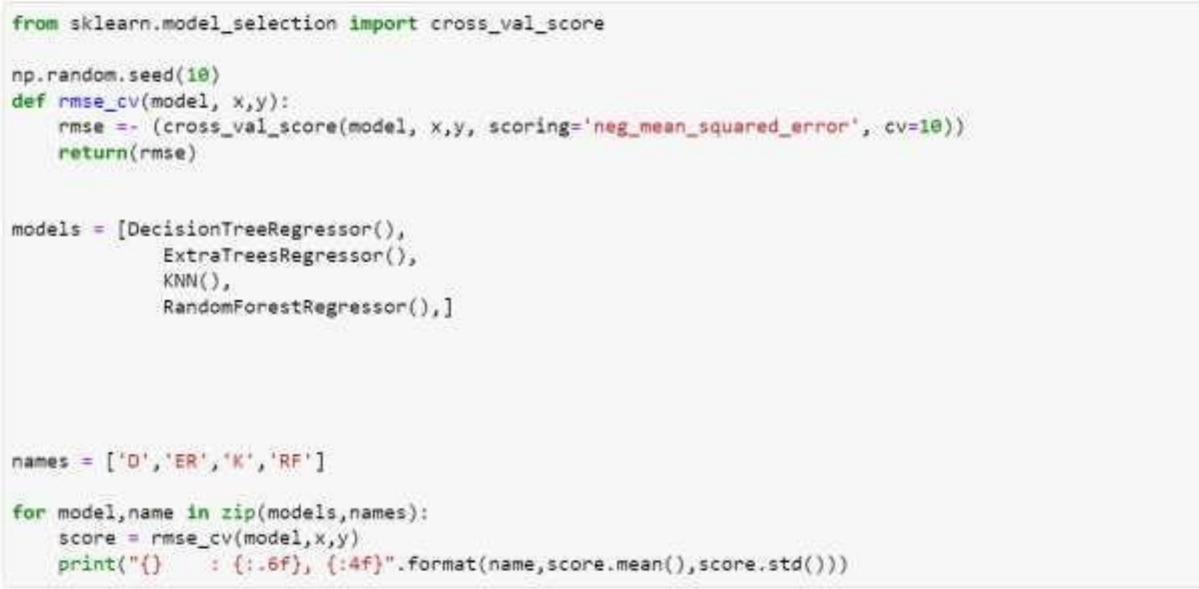


5. KNN

```
from sklearn.neighbors import KNeighborsRegressor as KNN
knn=KNN()
knn.fit(X_train,y_train)
pred=knn.predict(X_test)
print('R2_score:',r2_score(y_test,pred))
print('mean_squared_error:',metrics.mean_squared_error(y_test,pred))
print('mean_absolute_error:',metrics.mean_absolute_error(y_test,pred))
print('root_mean_squared_error:',np.sqrt(metrics.mean_squared_error(y_test,pred)))
```

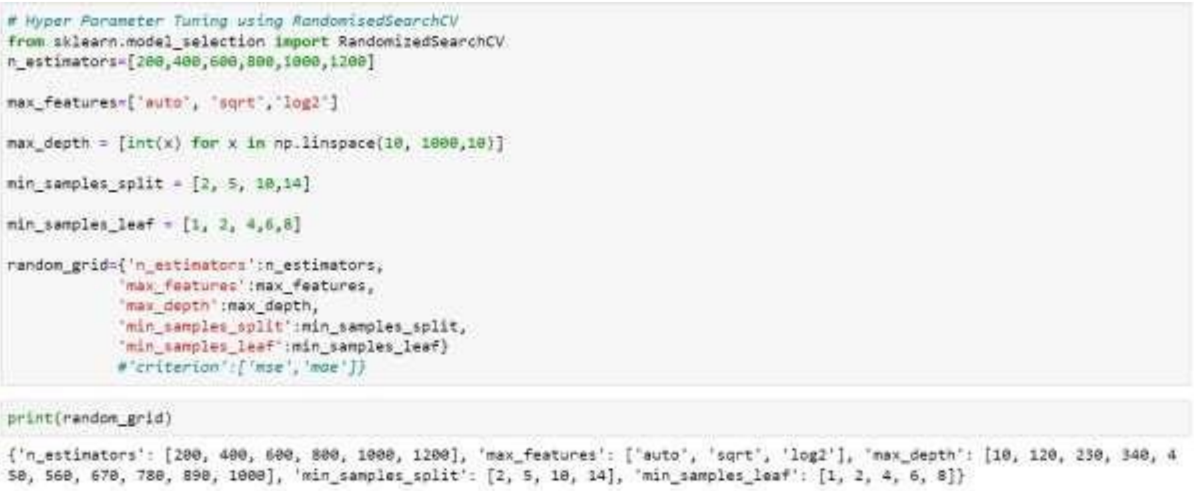
```
R2_score: 0.502097367704128
mean_squared_error: 9116427.478492063
mean_absolute_error: 2013.9932539682538
root_mean_squared_error: 3019.342226130066
```

Cross Validation



```
from sklearn.model_selection import cross_val_score

np.random.seed(10)
def rmse_cv(model, x,y):
    rmse =- (cross_val_score(model, x,y, scoring='neg_mean_squared_error', cv=10))
    return(rmse)


models = [DecisionTreeRegressor(),
             ExtraTreesRegressor(),
             KNN(),
             RandomForestRegressor(),]






names = ['D','ER','K','RF']

for model,name in zip(models,names):
    score = rmse_cv(model,x,y)
    print("{}    : {:.6f}, {:4f}".format(name,score.mean(),score.std()))
```

Hyper Parameter Tuning



```
# Hyper Parameter Tuning using RandomisedSearchCV
from sklearn.model_selection import RandomizedSearchCV
n_estimators=[200,400,600,800,1000,1200]

max_features=['auto', 'sqrt','log2']

max_depth = [int(x) for x in np.linspace(10, 1000,10)]

min_samples_split = [2, 5, 10,14]

min_samples_leaf = [1, 2, 4,6,8]

random_grid={'n_estimators':n_estimators,
             'max_features':max_features,
             'max_depth':max_depth,
             'min_samples_split':min_samples_split,
             'min_samples_leaf':min_samples_leaf}
             #'criterion':['mse','mae']}
```

```
print(random_grid)
```

```
{'n_estimators': [200, 400, 600, 800, 1000, 1200], 'max_features': ['auto', 'sqrt', 'log2'], 'max_depth': [10, 120, 230, 340, 4
50, 560, 670, 780, 890, 1000], 'min_samples_split': [2, 5, 10, 14], 'min_samples_leaf': [1, 2, 4, 6, 8]}
```

```
rf2 •Banaoffif or eatre$ees sa rt )
T_r anda19 •d-R8ndoaJ e•dS-aarehCV (a stSm•1•o<-=r*1.p •r•Jm_d1 str £ but Lan s -nandam r•Cd, r_1e ar-- UI', cv-7.varho¥• - 2.
```

```
rf_randomized.fit(X_train,y_train)
```

```
Rardomi za•Sea rch€ v ( c u= 3 , a s tTmaEer=RandomForgst gggr e F.her ( ) , next gn- 188,
                              par am_d1s I rfibution sa { ' na x deo th ' ' t be, zze, z se, z<B, 45B,
                                                                     ssh,  876,   78B,  are.
                                               'na 6oatu nea ' : [ ' aucs ' , “ sqrt ',
                                               '**zn samples s p* it ” - [2,  5, 2B , I4 1 •
                                                                  1000, 1200]},
                              random_state=100, verbose=2)
```

```
rf_randomized.best_params_
```

```
{'n_estimators': 600,
 'min_samples_split': 5,
 'min_samples_leaf': 1,
 'max_features': 'auto',
 'max_depth’: IB}
```

```
rf_final=rf_randomized.best_estimator_
```

```
y_pred_random=rf_final.predict(X_test)
```

```
B•st_nod= k ando•F or• stRcgrassor(naz_feat ur-•s = ' auto , mtn_s ainpJas_1•afi- 2.min_s rig1•s_s plot- z, n_•> u «a Koi s =ea , n-joo s -a )
eel «•sa. 7t< ( x_ t rJJn, y_T rg in ›
pred=Best_mod.predict(X_test)
pr- n I ' R*_5 co=e: ' , r2_s cora ( y_€e st- , pr-edF 1BO}
pre nc‹ 'nean_oqugr o4_err-or :  .<eCrJc5 .rnaan sCu-seed error (y flest .pred))
pni n< '-•=._ahEoz u Le_or ra •-  ,—*•$cs ,e*ean_abs.sJ.u re_<rrsr(›'_t9st,1u'gd) }print
f r:'1s E aa ue:, np . ñcirt (ztgt; r jcc .mqan_s guarad_error [ y_te st. Prod j ) )
```

```
R2_Score: 67.04099107553559
mean_squared_error: [illegible]2.187[illegible]565
mean absolute error: 1517.299818512685
RMSE value: 2456.559013642163
```

Hence. are cBn 6ee Ihat Random Pofas. 9ives us w bast accwacy~~—a~~

- ## Key Metrics for success in solving problem under consideration

Will go with Random Forest Regressor as it gives us the best accuracy score.

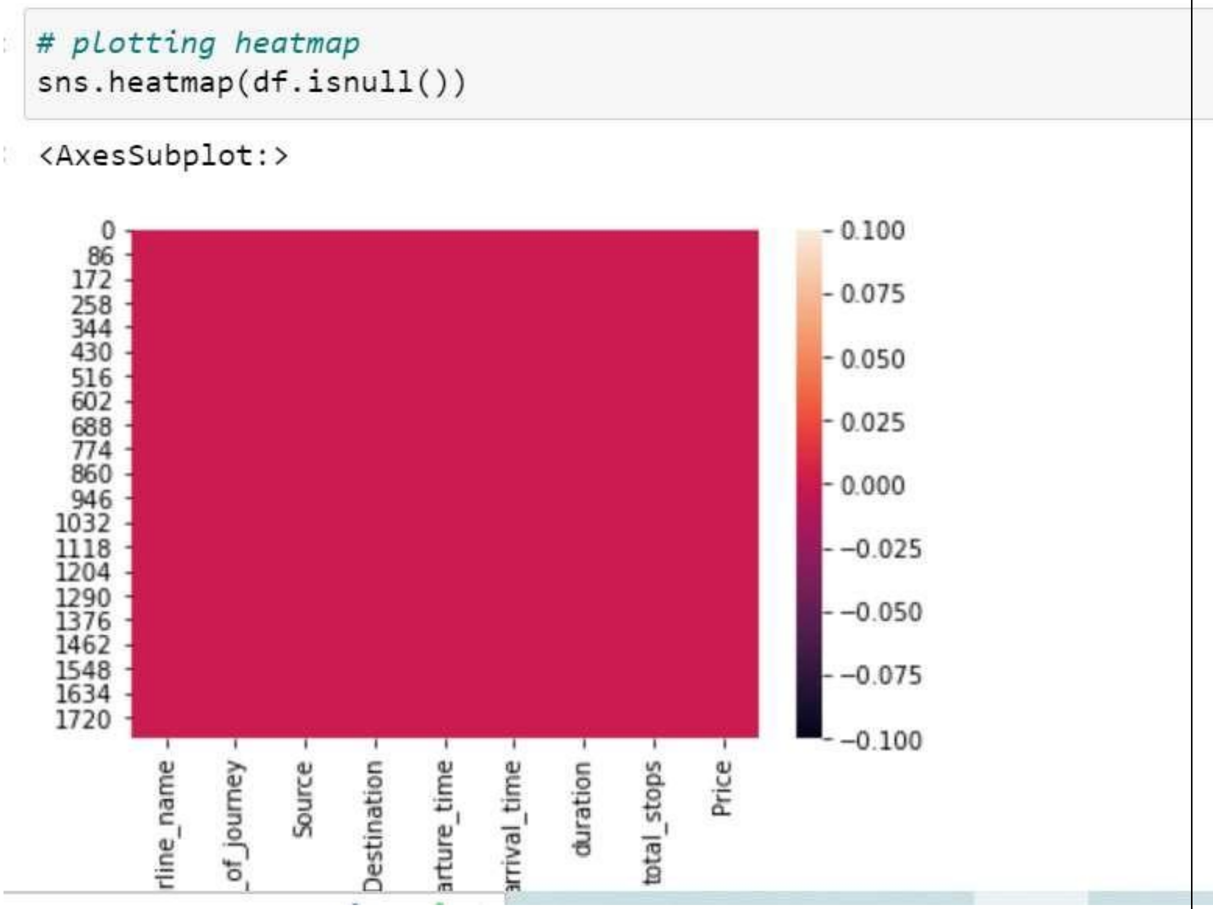
Reasons:

1. Random Forest reduces overfitting in decision tree and helps to improve accuracy.
2. It is flexible for both classification and regression tasks.
3. It also works well with both continuous and categorical variables.
4. It is a rule based approach.
5. It automates missing values present in the data.

- ## Visualizations

I have compared by plotting plots on different features with the label and compared the variation of fares with different classes in the features and tried to know how label is varying with the features and even plotted hist plots for all the columns and checked the variation.

1. Null values



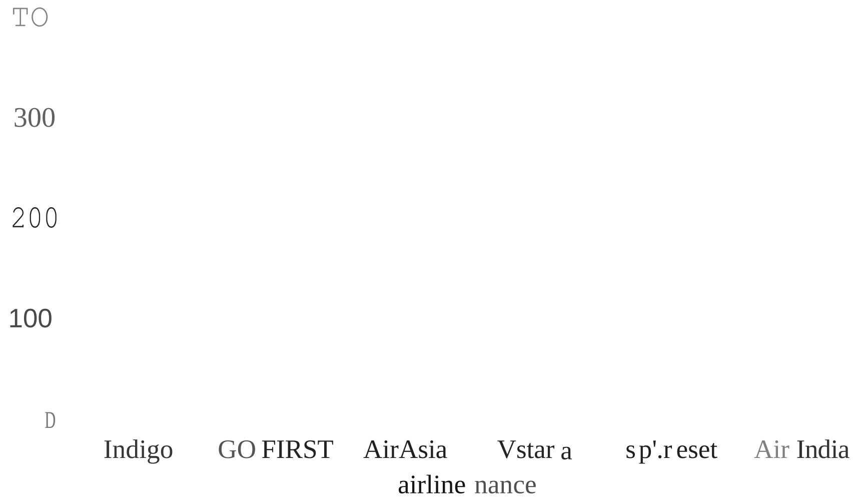
```
# plotting heatmap
sns.heatmap(df.isnull())
```

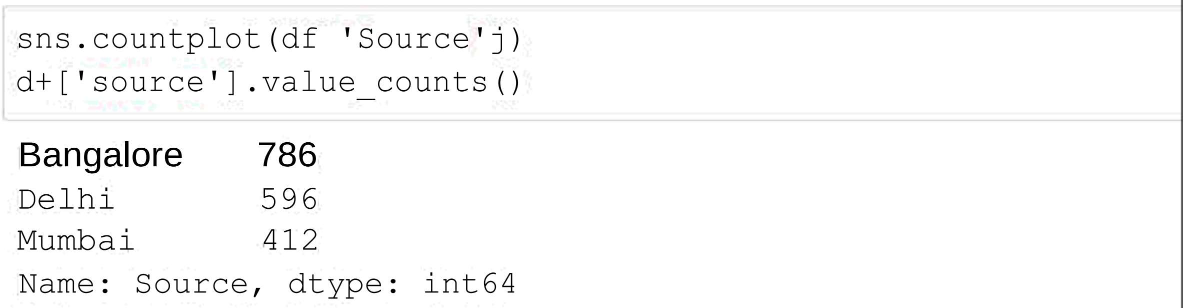
```
<AxesSubplot:>
```

2. Checking for value counts of all the features and target variable

```
sns.countolot(df['airline_name'])
df['airline_name'].value_counts()
```

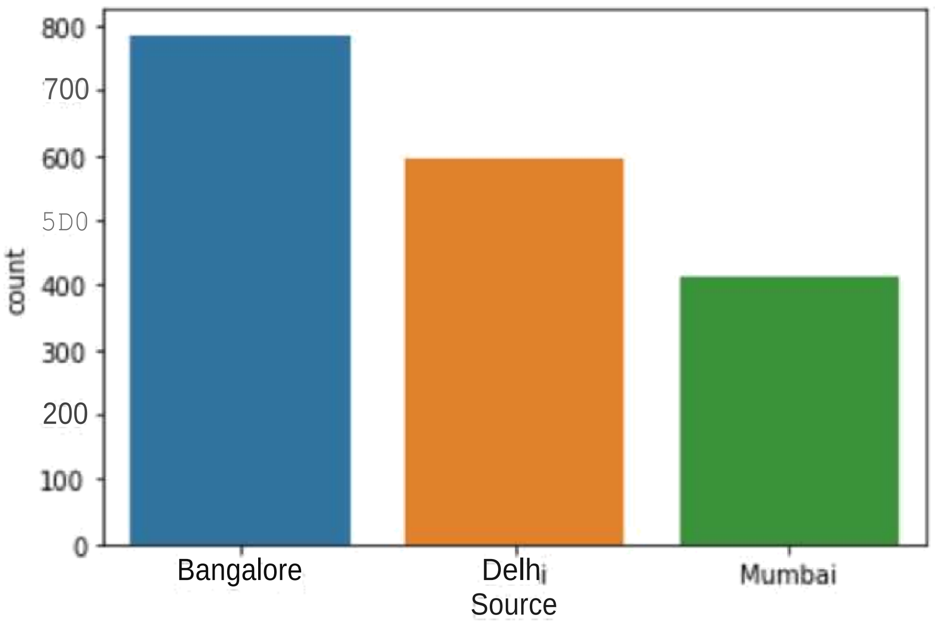
```
Vistara       490
Air   Ind ia  363
Indigo        361
GO FIRST      229
SpiceJet      183
AirAsia       168
Name: airline_name, dtype: int64
```



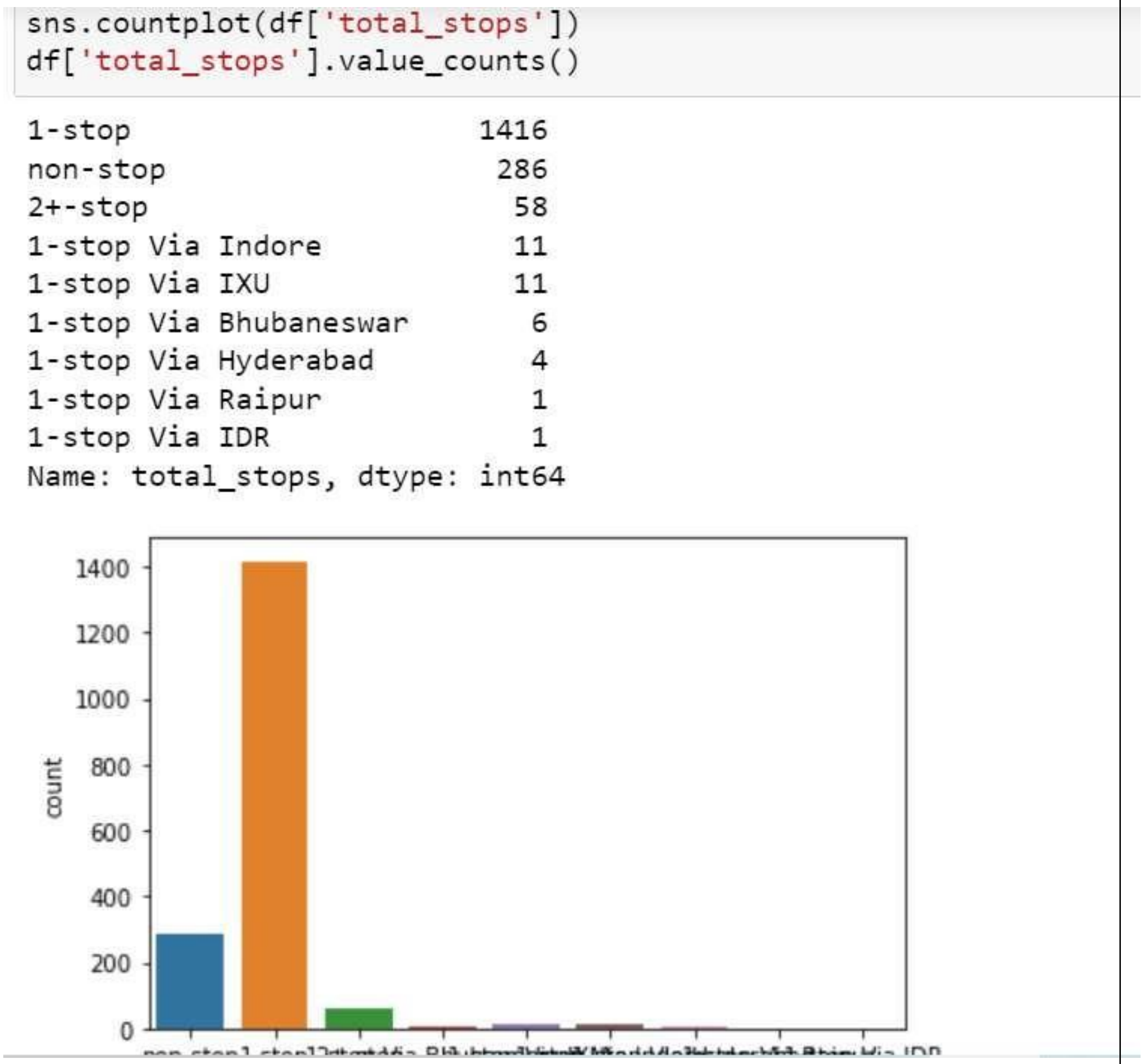


```
sns.countplot(df 'Source'j)
d+['source'].value_counts()
```

```
Bangalore    786
Delhi        596
Mumbai       412
Name: Source, dtype: int64
```



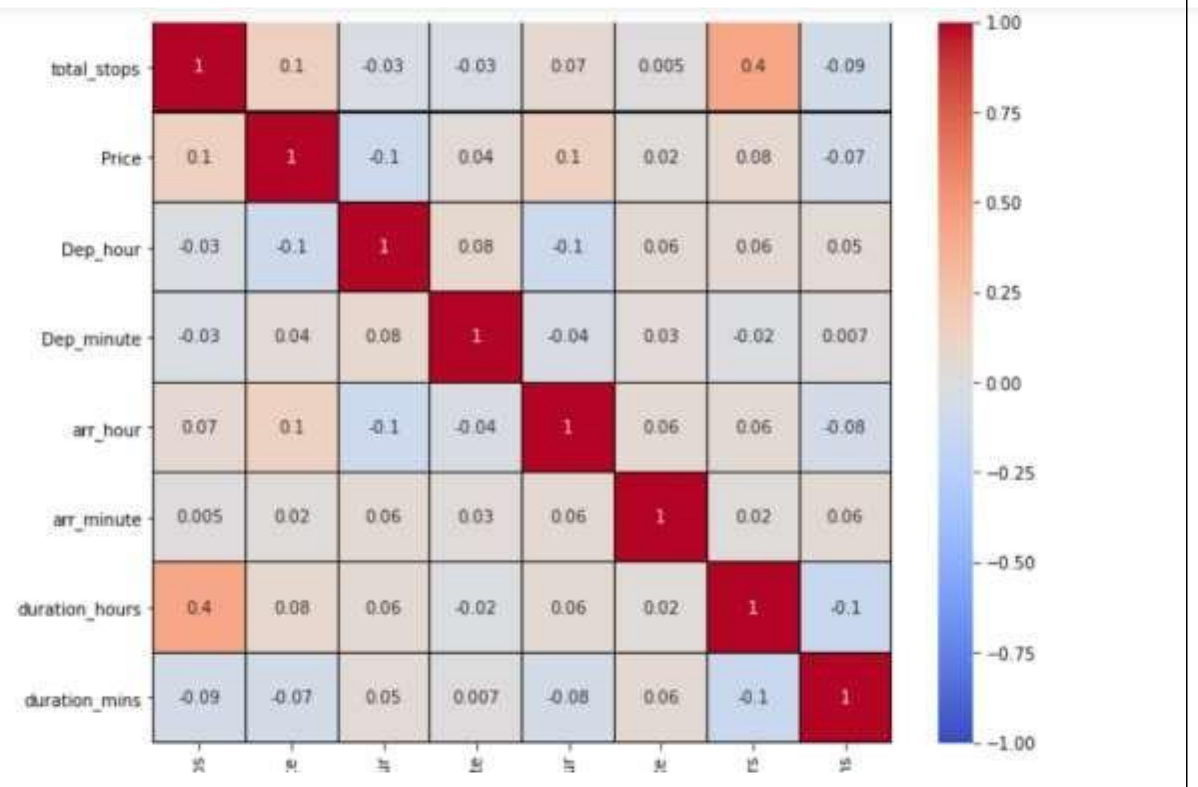
```
sns.countplot(df('Destination'])
df['Destination'].value_counts()

Mumbai       786
8angalore    448
Kolkata      327
Delhi        233
Name: Destination, dtype: int64
```



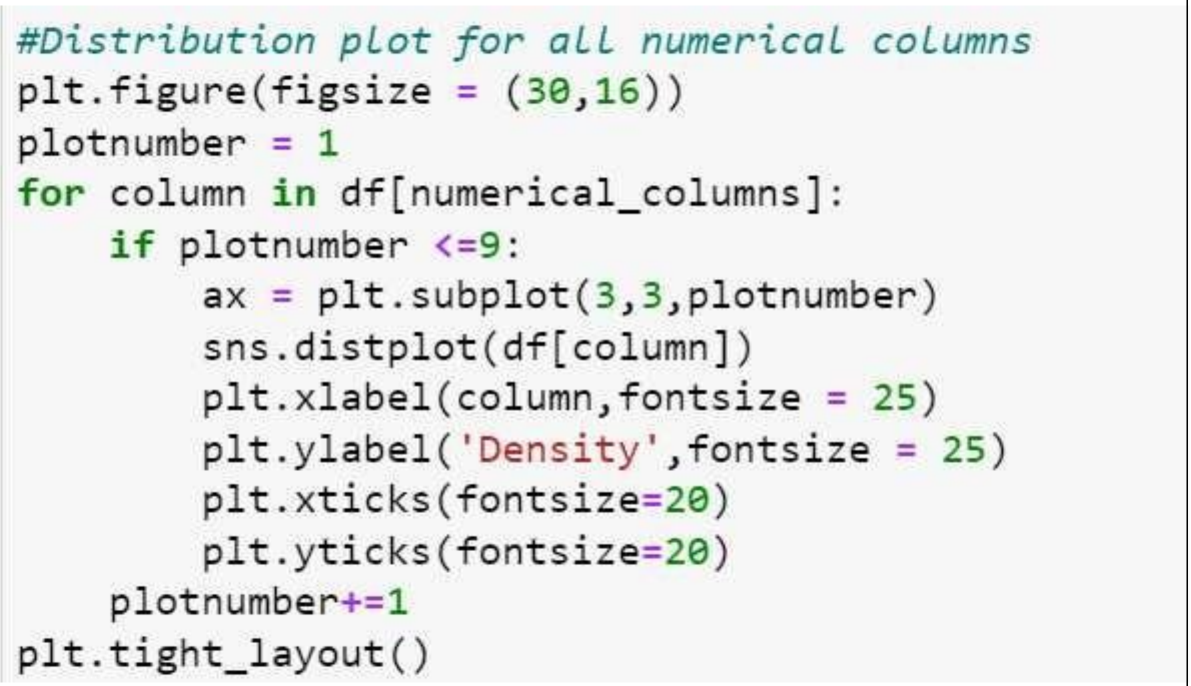
```
sns.countplot(df['total_stops'])
df['total_stops'].value_counts()
```

```
1-stop                       1416
non-stop                      286
2+-stop                        58
1-stop Via Indore              11
1-stop Via IXU                 11
1-stop Via Bhubaneswar          6
1-stop Via Hyderabad            4
1-stop Via Raipur               1
1-stop Via IDR                  1
Name: total_stops, dtype: int64
```

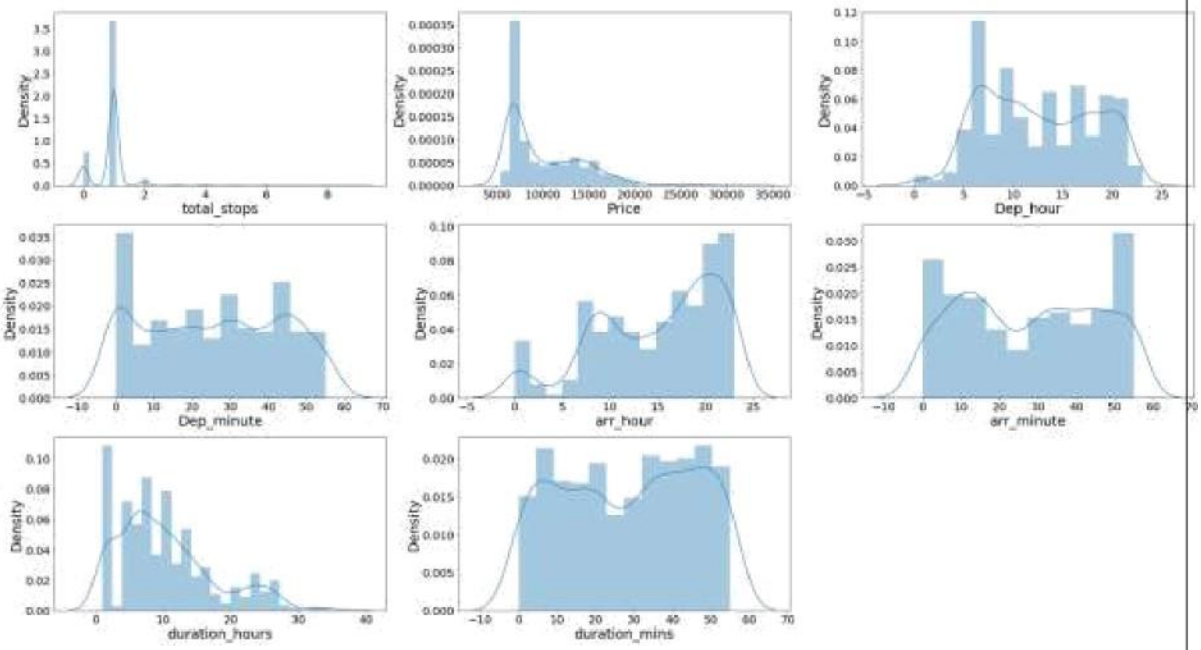
3. Visualizing the correlation matrix by plotting heat map.

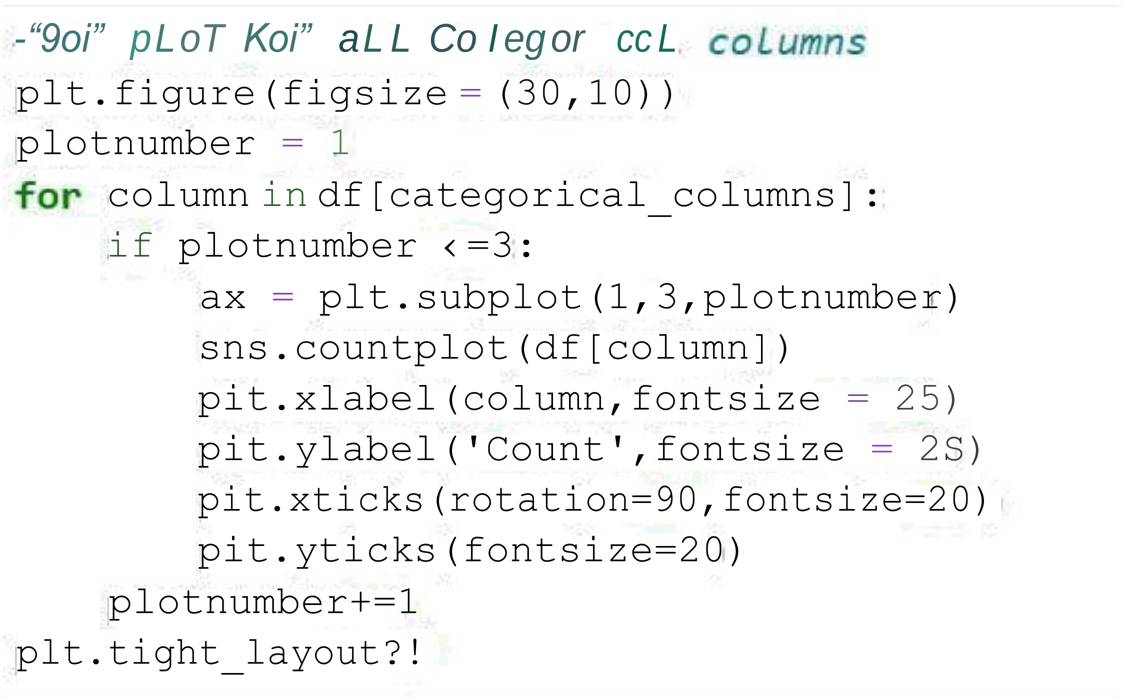


4. Univariate Analysis

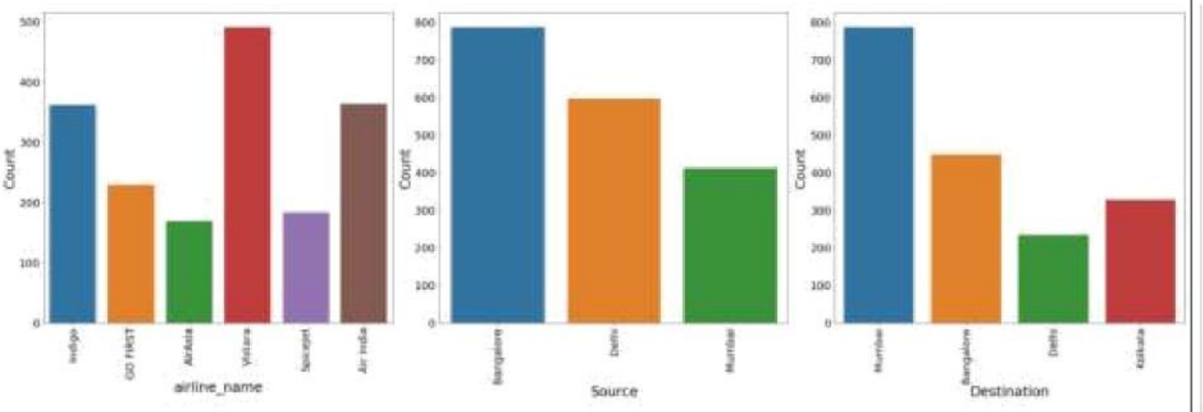


```
#Distribution plot for all numerical columns
plt.figure(figsize = (30,16))
plotnumber = 1
for column in df[numerical_columns]:
    if plotnumber <=9:
        ax = plt.subplot(3,3,plotnumber)
        sns.distplot(df[column])
        plt.xlabel(column,fontsize = 25)
        plt.ylabel('Density',fontsize = 25)
        plt.xticks(fontsize=20)
        plt.yticks(fontsize=20)
    plotnumber+=1
plt.tight_layout()
```





```
-“9oi” pLoT Koi” aLL Co legor ccL coLumns
plt.figure(figsize = (30,10))
plotnumber = 1
for column in df[categorical_columns]:
    if plotnumber <=3:
        ax = plt.subplot(1,3,plotnumber)
        sns.countplot(df[column])
        pit.xlabel(column,fontsize = 25)
        pit.ylabel('Count',fontsize = 2S)
        pit.xticks(rotation=90,fontsize=20)
        pit.yticks(fontsize=20)
    plotnumber+=1
plt.tight_layout?!
```



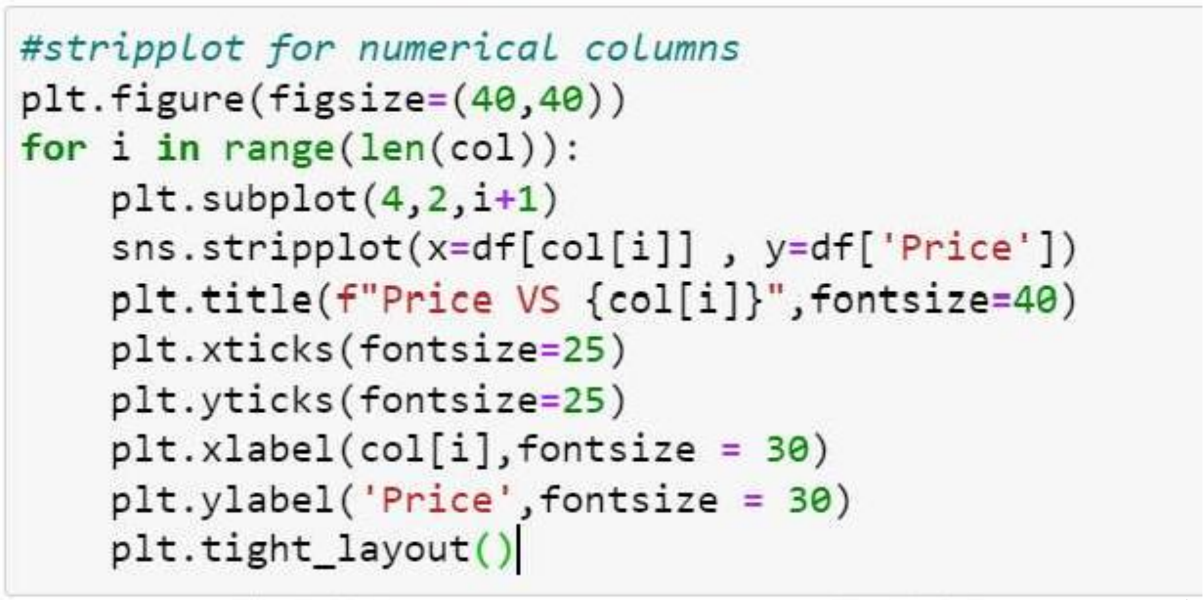
Observations:

Vistara has maximum count which means most of the passengers preferred Vistara for there travelling.

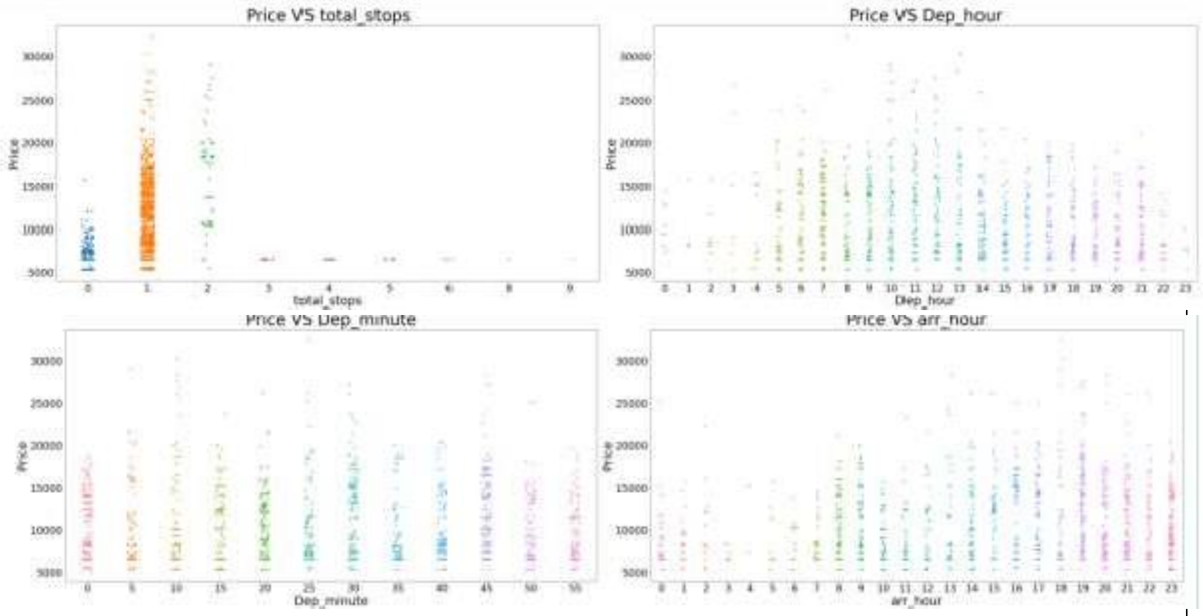
Bangalore has maximum count for source which means maximum passengers are choosing Bangalore as there source.

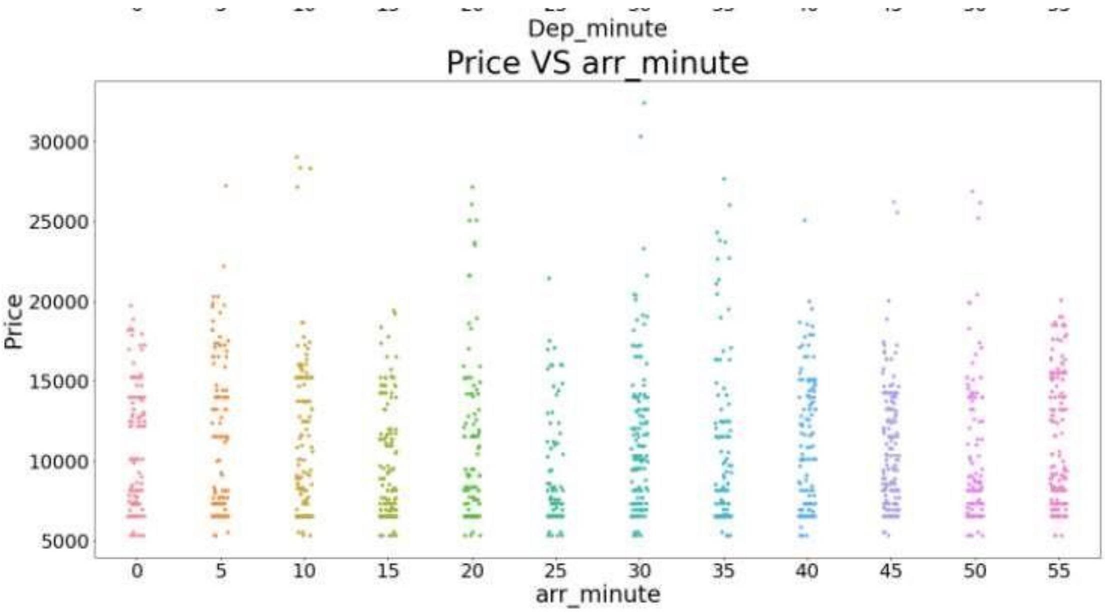
Mumbai has maximum count for Destination which means maximum passengers are choosing Mumbai as there Destination.

5. Bi variate Analysis



```
#stripplot for numerical columns
plt.figure(figsize=(40,40))
for i in range(len(col)):
    plt.subplot(4,2,i+1)
    sns.stripplot(x=df[col[i]] , y=df['Price'])
    plt.title(f"Price VS {col[i]}",fontsize=40)
    plt.xticks(fontsize=25)
    plt.yticks(fontsize=25)
    plt.xlabel(col[i],fontsize = 30)
    plt.ylabel('Price',fontsize = 30)
    plt.tight_layout()
```

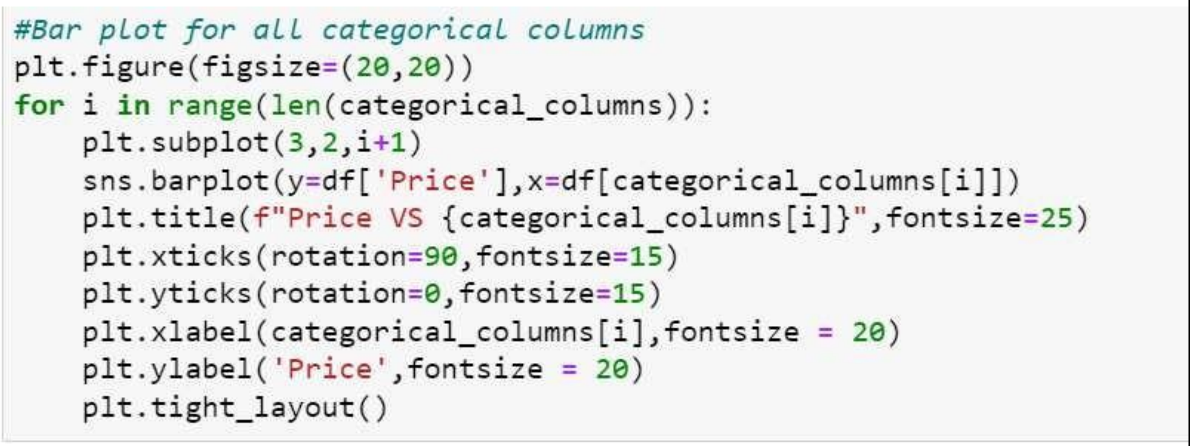




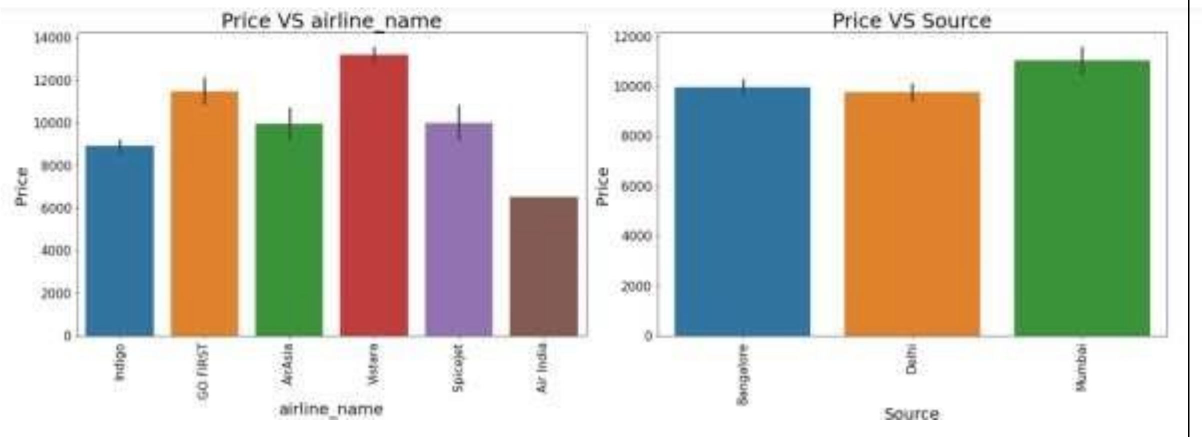
Price VS arr_minute

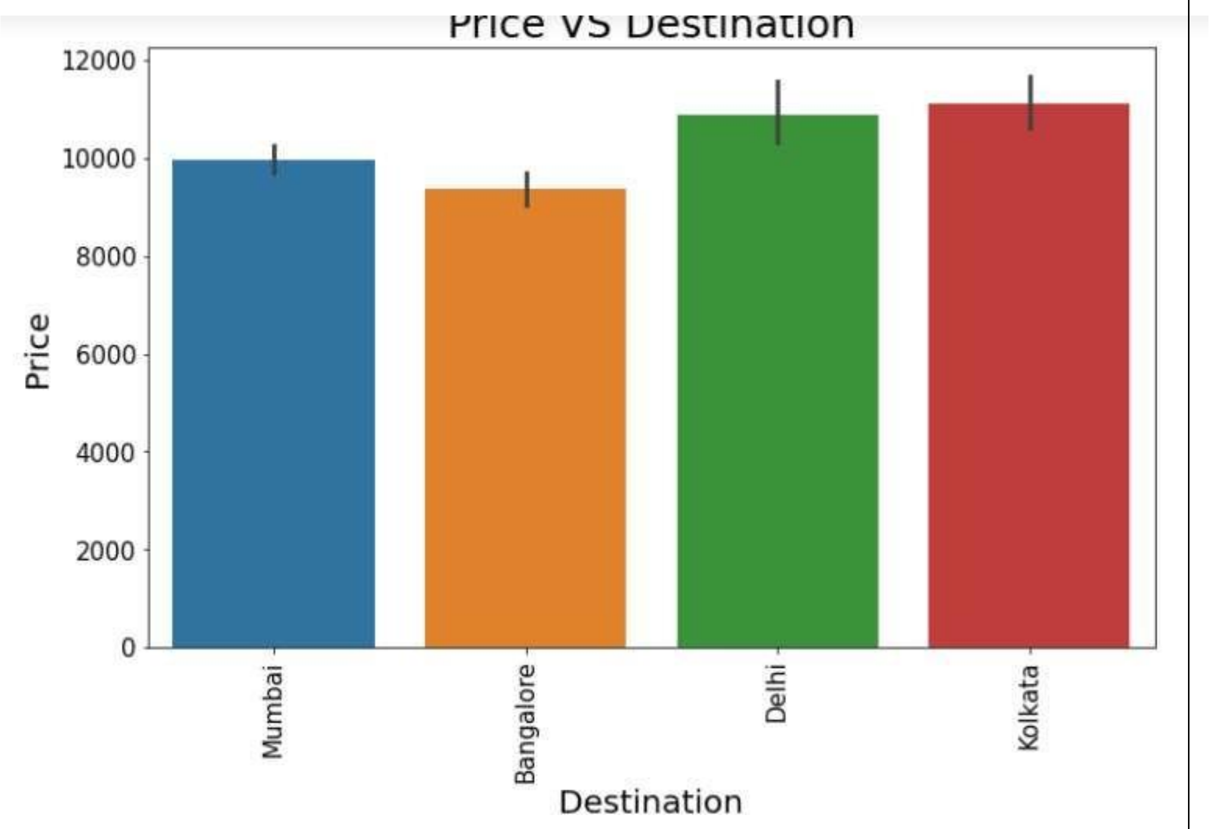
Observations:

1. Flights with 1 stop costs more price compared to other flights.
2. At noon time of every day the flight Prices are high so it looks good to book flights rather than noon.
3. And Departure minute has less relation with target Price.
4. At 7AM to 1PM Arrival time of every day the flight Prices are high so it looks good to book flights rather than this arrival time.
5. And Arrival minute has less relation with target Price.



```
#Bar plot for all categorical columns
plt.figure(figsize=(20,20))
for i in range(len(categorical_columns)):
    plt.subplot(3,2,i+1)
    sns.barplot(y=df['Price'],x=df[categorical_columns[i]])
    plt.title(f"Price VS {categorical_columns[i]}",fontsize=25)
    plt.xticks(rotation=90,fontsize=15)
    plt.yticks(rotation=0,fontsize=15)
    plt.xlabel(categorical_columns[i],fontsize = 20)
    plt.ylabel('Price',fontsize = 20)
    plt.tight_layout()
```





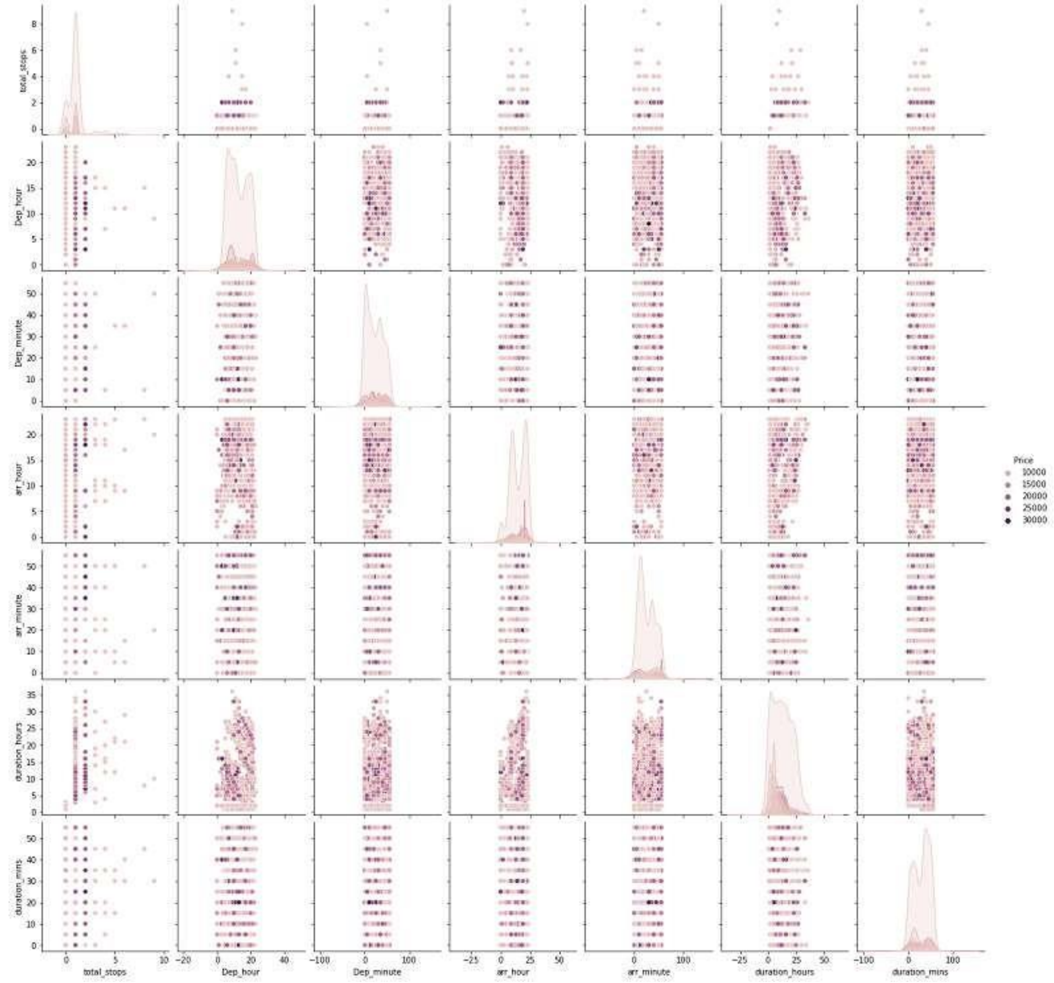
Observations:

For Go First Airlines the Price is high compared to other Airlines.

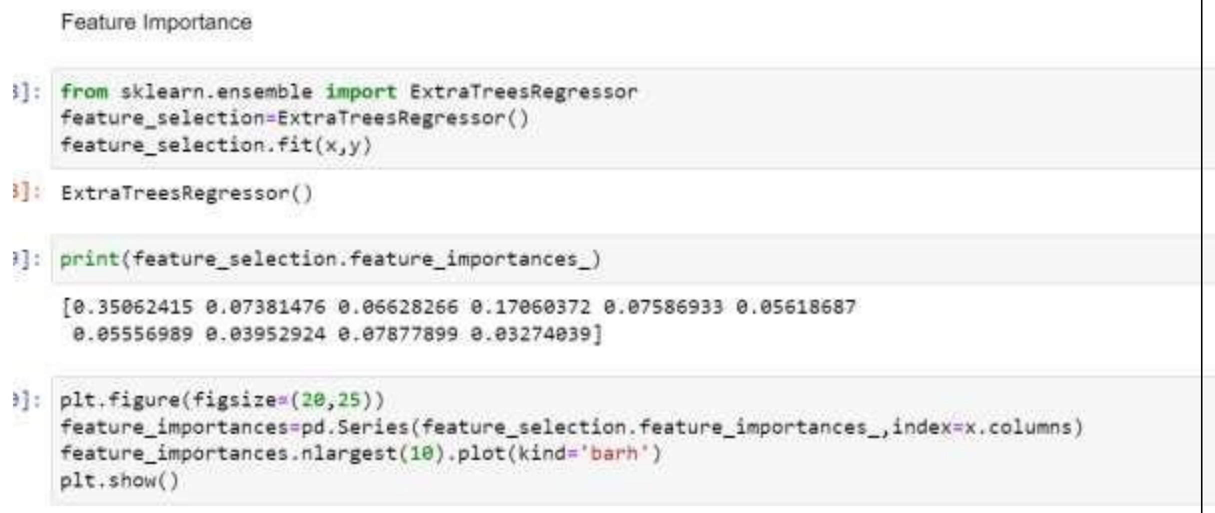
All the Sources has approximately same prices.

Destination also has the approximately same prices.

6. Multi-Variate Analysis

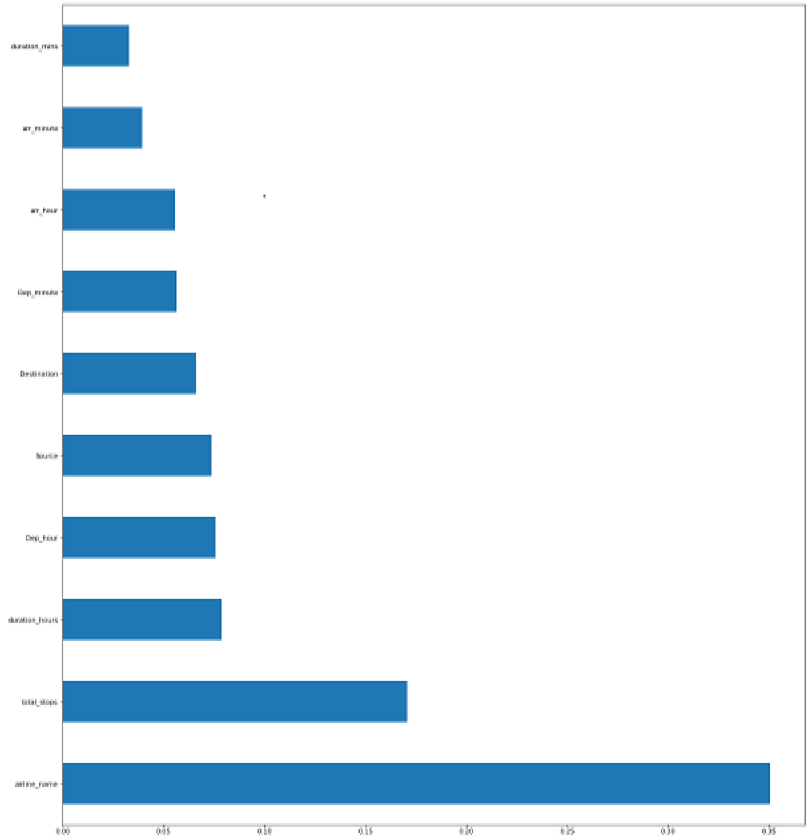


7. Feature Importance



Feature Importance

```
3]: from sklearn.ensemble import ExtraTreesRegressor
    feature_selection=ExtraTreesRegressor()
    feature_selection.fit(x,y)

3]: ExtraTreesRegressor()

]: print(feature_selection.feature_importances_)

    [0.35062415 0.07381476 0.06628266 0.17060372 0.07586933 0.05618687
     0.05556989 0.03952924 0.07877899 0.03274039]

]: plt.figure(figsize=(20,25))
    feature_importances=pd.Series(feature_selection.feature_importances_,index=x.columns)
    feature_importances.nlargest(10).plot(kind='barh')
    plt.show()
```



- Interpretation of the Results

From the above EDA we can easily understand the relationship between features and can also determine which features are affecting the price of the flights.

In UNIVARIATE Analysis, I have used count plots to visualize the counts in categorical variables and distribution plots to visualize the numerical variables.

In BIVARIATE Analysis, I have used bar plots, to check the relation between label and the features.

Used pair plots to check the pairwise relation between the features.

The heat map and bar plot helped in understanding the correlation between dependent and independent variables.

Detected outliers and skewness with the help of box plots and distribution plots respectively.

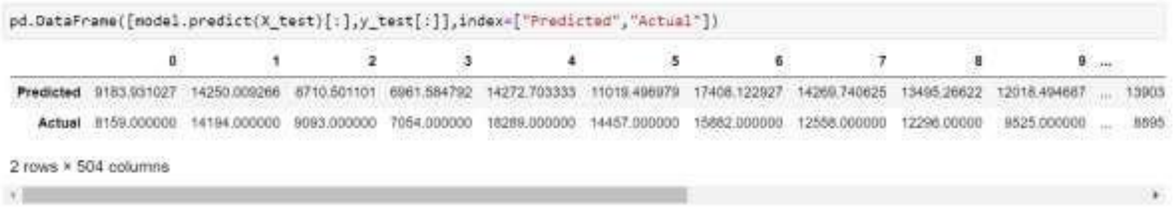
CONCLUSION

- Key Findings and Conclusions of the Study

I have used various models for predicting the price of flights and used various evaluation metrics for evaluating the model like finding the R2 Score, Mean Squared error (MSE), Mean Absolute error (MAE), Root Mean squared error (RMSE). So, after evaluating on different models, Random Forest Regressor is giving high score and low RMSE Value. So I finalised the model and saved the model using job-lib library.

- Learning Outcomes of the Study in respect of Data Science

After finalising the model Random Forest Regressor, I have taken the values of prices which are predicted by the model and compared with the actual Price values.



```
pd.DataFrame([model.predict(X_test)[:],y_test[:]],index=["Predicted","Actual"])
```

	0	1	2	3	4	5	6	7	8	9	...	
Predicted	9183.931027	14250.009266	8710.601101	6961.584792	14272.703333	11019.496979	17408.122927	14269.740625	13495.26622	12018.494667	...	13903
Actual	8159.000000	14194.000000	9093.000000	7054.000000	18289.000000	14457.000000	15882.000000	12558.000000	12296.00000	9525.000000	...	8895

2 rows × 504 columns