FLIP ROBO

Flight Price Prediction Project

Submitted by:

Rakesh Chaudhary

ACKNOWLEDGMENT

This project would not have seen the light of the day without the following people and their priceless support and cooperation. Hence I extend my gratitude to all of them.

As a student of Data Trained Education, I would first of all like to express my gratitude to FlipRobo Team and seniors for granting me permission to undertake the project report in their esteemed organization. I would also like to express my sincere thanks to Miss Khushboo Mam for supporting me and being always there for me whenever I needed.

During the actual research work with FlipRobo team and other IOT that set the ball rolling for my project. They had been a source of inspiration through their constant guidance; personal interest; encouragement and help.

I convey my **sincere thanks** to them. In spite of their busy schedule they always found time to guide me throughout the project. I am also grateful

to them for reposing confidence in my abilities and giving me the freedom to work on my project. Without their invaluable help I would not have been able to do justice to the project.

INTRODUCTION

Business Problem Framing

- In this article, we will be analysing the flight fare prediction using Machine Learning dataset using essential exploratory data analysis techniques then will draw some predictions about the price of the flight based on some features such as what type of airline it is, what is the arrival time, what is the departure time, what is the duration of the flight, source, destination and more.
- Optimal timing for airline ticket purchasing from the consumer's perspective is challenging principally because buyers have insufficient information for reasoning about future price movements. In this project we majorly targeted to uncover underlying trends of flight prices in India using historical data and also to suggest the best time to buy a flight ticket.
- For this project, we have collected data from 15 routes across India while the data of 18 routes were extensively used for the analysis due to the sheer volume of data collected over 4 days resulting in more than 3000 data points each across the Mumbai-Delhi and Delhi-Mumbai route and many more routes. The project implements the validations or contradictions towards myths regarding the airline industry, a comparison study among various models in predicting the optimal time to buy the flight ticket and the amount that can be saved if done so.
- Remarkably, the trends of the prices are highly sensitive to the route, month of departure, day of departure, time of departure, whether the day of departure is a holiday and airline carrier.

Conceptual Background of the Domain Problem

Anyone who has booked a flight ticket knows how unexpectedly the prices vary. Airlines use using sophisticated quasi-academic tactics known as "revenue management" or "yield management". The cheapest available ticket for a given date gets more or 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, if we could inform the travellers with the optimal time to buy their flight tickets based on the historic data and also show them various trends in the airline industry we could help them save money on their travels. This would be a practical implementation of a data analysis, statistics and machine learning techniques to solve a daily problem faced by travellers.

Review of Literature:

❑ Automated Script to Collect Historical Data

For any prediction/classification problem, we need historical data to work with. In this project, past flight prices for each route needs to be collected on a daily basis. Manually collecting data daily is not efficient and thus a python script was run on a remote server which collected prices daily at specific time.

❑ Cleaning & Preparing Data

After we have the data, we need to clean & prepare the data according to the model's requirements. In any machine learning problem, this is the step that is the most important and the most time consuming. We used various statistical techniques & logics and implemented those using built-in python packages.

❑ Analysing & Building Models

Data preparation is followed by analysing the data, uncovering hidden trends and then applying various predictive & classification models on the training set. These included Random Forest, Logistic Regression, Gradient Boosting and combination of these models to increase the accuracy. Further statistical models and trend analyser model have been built to increase the accuracy of the ML algorithms for this task.

❑ Merging Models & Accuracy Calculation

Having built various models, we have to test the models on our testing set and calculate the savings or loss done on each query put by the user. A statistic of the over Savings, Loss and the mean saving per transaction are the measures used to calculate the Accuracy of the model implemented.

Motivation for the Problem Undertaken:-

Travelling through flights has become an integral part of today's lifestyle as more and more people are opting for faster travelling options. The flight ticket prices increase or decrease every now and then depending on various factors like timing of the flights, destination, and duration of flights. Various occasions such as vacations or festive season. Therefore, having some basic idea of the flight fares before planning the trip will surely help many people save money and time. In the proposed system a predictive model will be created by applying machine learning algorithms to the collected historical data of flights. This system will give people the idea about the trends that prices follow and also provide a predicted price value which they can refer to before booking their flight tickets to save money. This kind of system or service can be provided to the customers by flight booking companies which will help the customers to book their tickets accordingly.

Analytical Problem Framing

Mathematical/ Analytical Modelling of the Problem:-

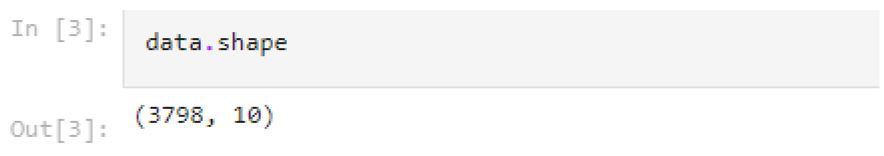
Research methodology is a way to systematically solve the problem. It is a game plan for conducting research. In this we describe various steps that are taken by the researcher.

To build a better machine learning model of predicting the defaulter case from micro credit card data set we encounter with several statistical modelling while data analysis & off course we needed mathematical modelling to build several machine learning models in this project.

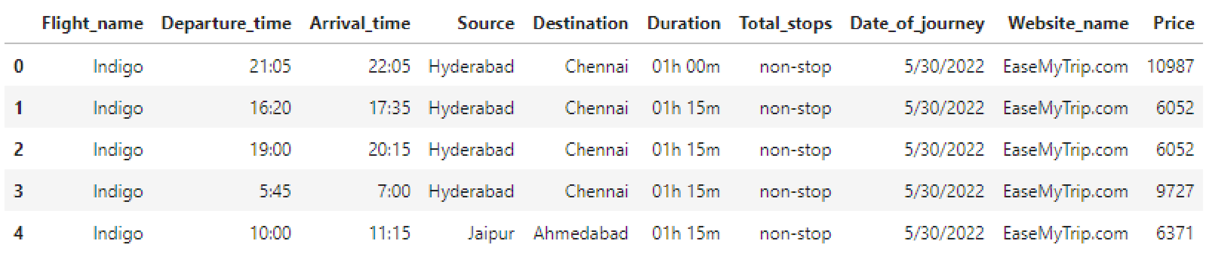
"All progress is born of inquiry. Doubt is often better than overconfidence, for it leads to inquiry and inquiry leads to invention."

Research methodology is a framework for the study and is used as a guide in collecting and analysing the data. It is a strategy specifying which approach will be used for gathering and analysing the data. It also includes time and cost budget since most studies are done under these two constraints. The research methodology includes overall research design, the sampling procedure, the data collection method and analysis procedure.

Some Mathematical analysis shown below screenshots:-



```
In [3]:  data.shape

Out[3]:  (3798, 10)
```



	Flight_name	Departure_time	Arrival_time	Source	Destination	Duration	Total_stops	Date_of_journey	Website_name	Price
0	Indigo	21:05	22:05	Hyderabad	Chennai	01h 00m	non-stop	5/30/2022	EaseMyTrip.com	10987
1	Indigo	16:20	17:35	Hyderabad	Chennai	01h 15m	non-stop	5/30/2022	EaseMyTrip.com	6052
2	Indigo	19:00	20:15	Hyderabad	Chennai	01h 15m	non-stop	5/30/2022	EaseMyTrip.com	6052
3	Indigo	5:45	7:00	Hyderabad	Chennai	01h 15m	non-stop	5/30/2022	EaseMyTrip.com	9727
4	Indigo	10:00	11:15	Jaipur	Ahmedabad	01h 15m	non-stop	5/30/2022	EaseMyTrip.com	6371

### Observations:-

- As we can see that all the features are object data type except our target variable Price.
- Since data is in form of csv file we have to use pandas read csv to load the data.
- After loading it is important to check null values in a column or a row
- If it is present then following can be done,

  Filling NaN values with mean, median and mode using fillna() method

  If Less missing values, we can drop it as well

Data Sources and their formats:

Data selection is the first step where historical data of flight is gathered for the model to predict prices. Our dataset consists of more than 3,000 records of data related to flights and its prices. Some of the features of the dataset are source, destination, departure date, departure time, number of stops, arrival time, prices and few more.

In the exploratory data analysis step, we cleaned the dataset by removing the duplicate values and null values. If these values are not removed it would affect the accuracy of the model. We gained further information such as distribution of data.

Next step is data pre-processing where we observed that most of the data was present in string format. Data from each feature is extracted such as day and month is extracted from date of journey in integer format, hours and minutes is extracted from departure time. Features such as source and destination needed to be converted into values as they were of categorical type. For this One hot-encoding and label encoding techniques are used to convert categorical values to model identifiable values.

Data Description:-

### Target:

Target	Definition
`Price`	The price of the ticket.

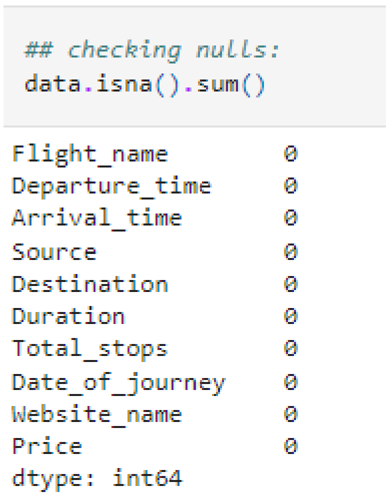
### Features:-

Features	Definition
Flight_name:	The name of the airline.

Date_of_Journey:	The date of the journey
Source:	The source from which the service begins.
Destination:	The destination where the service ends.
Departure_time:	The time when the journey starts from the source.
Arrival_Time:	Time of arrival at the destination.
Duration:	Total duration of the flight.
Total_Stops:	Total stops between the source and destination.
Website_name:	Online website name, where you can book your ticket online.

Data Pre-processing Done:-

## Checking Missing Values:-

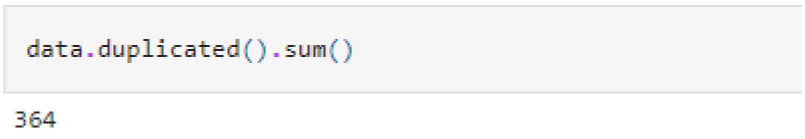


```python
## checking nulls:
data.isna().sum()
```

```
Flight_name       0
Departure_time    0
Arrival_time      0
Source            0
Destination       0
Duration          0
Total_stops       0
Date_of_journey   0
Website_name      0
Price             0
dtype: int64
```

There is no nulls present in my dataset.

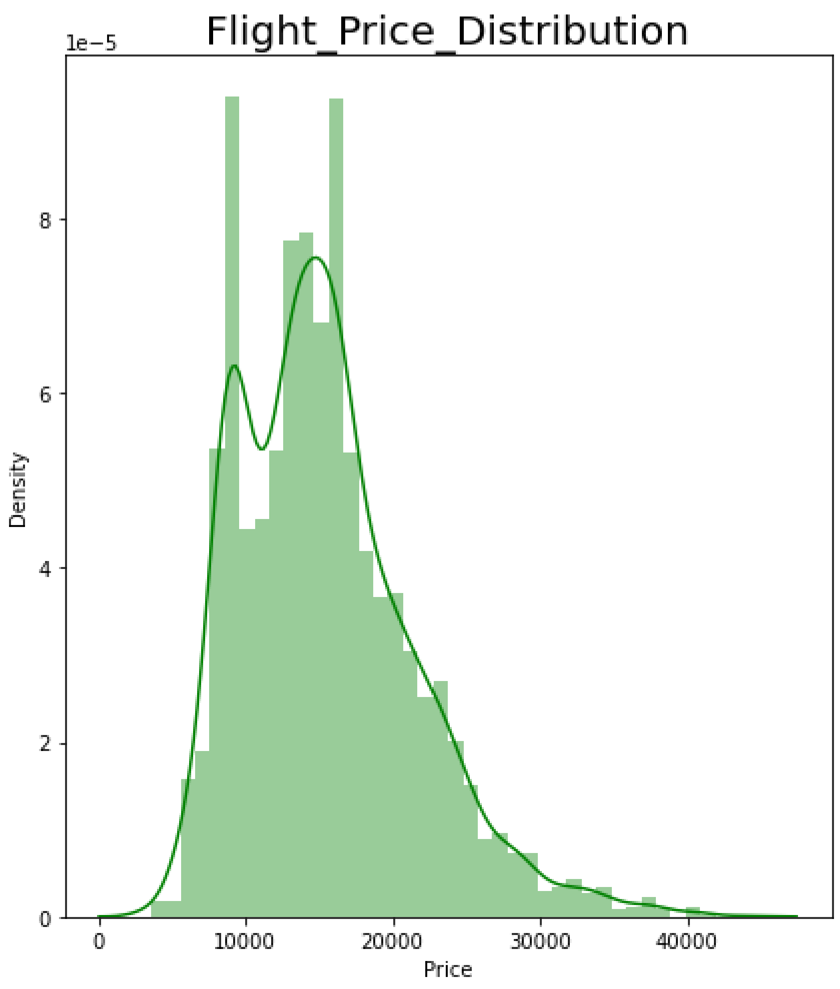
### Checking Duplicates:-



```python
data.duplicated().sum()
```

```
364
```

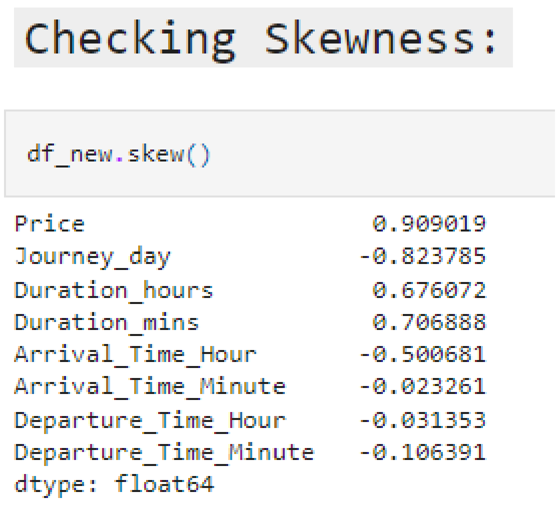
### Checking Target Data Distribution:-



Note:-

- As we can see in the above distribution of our target variable i.e. price, it seems little right skewed. It may be possible that there are some outliers present in my dataset.
- If we ignore them our target variable seems bell curved.

### Checking Skewness:-



Checking Skewness:

```
df_new.skew()
```

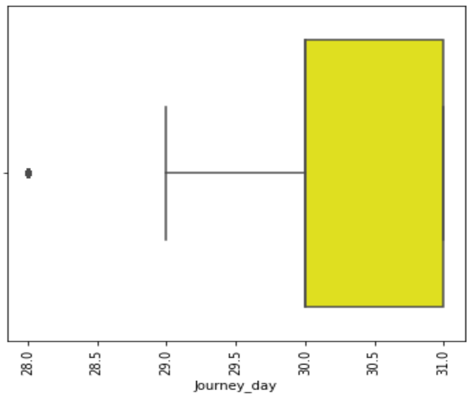
```
Price                   0.909019
Journey_day            -0.823785
Duration_hours          0.676072
Duration_mins           0.706888
Arrival_Time_Hour      -0.500681
Arrival_Time_Minute    -0.023261
Departure_Time_Hour    -0.031353
Departure_Time_Minute  -0.106391
dtype: float64
```

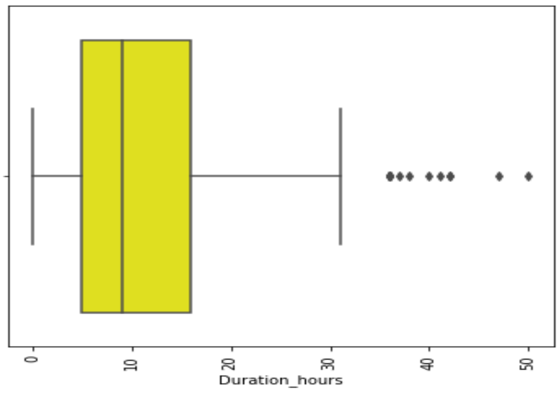
### Handling Skewness:-

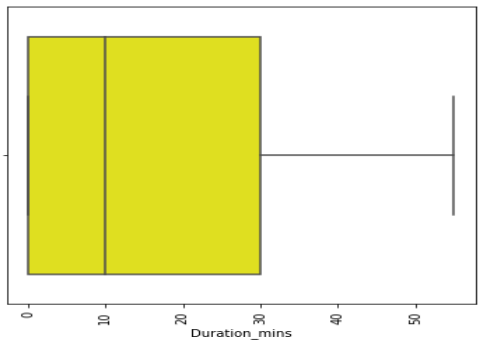
```
## Handling only continous type feature:
df_new.Duration_Thours=np.sqrt(df_new['Duration_hours'])
df_new.Duration_mins=np.sqrt(df_new['Duration_mins'])



print(df_new.skew())
```

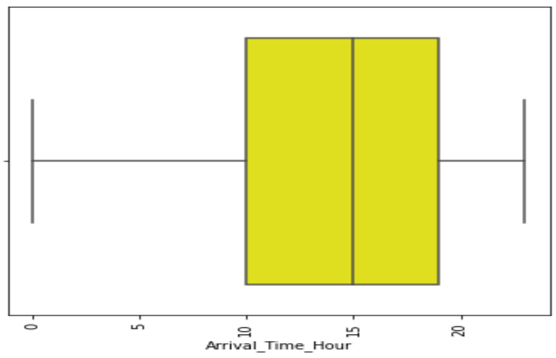
```
Price                    0.909019
Journey_day             -0.823785
Duration_hours           0.676072
Duration_mins            0.177516
Arrival_Time_Hour       -0.500681
Arrival_Time_Minute     -0.023261
Departure_Time_Hour     -0.031353
Departure_Time_Minute   -0.106391
dtype: float64
```

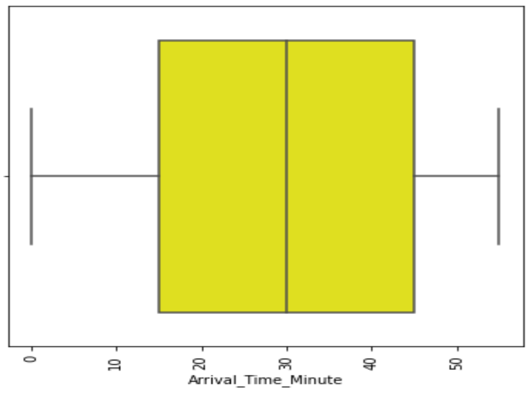
### Checking Outliers:-

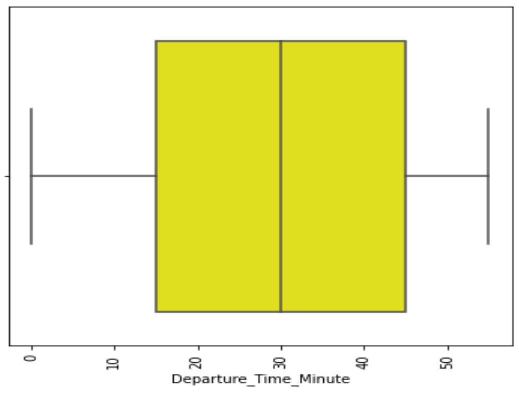












### Handling Outliers:-

```
from scipy.stats import zscore
z_score=zscore(data[['Journey_day','Duration_hours']])
abs_zscore=np.abs(z_score)
```

```
threshold=3
new_entry=(abs_zscore<threshold).all(axis=1)
df_new=data[new_entry]
print("The shape before: ", data.shape)
print("The shape after: ",df_new.shape)
```

```
The shape before:  (3434, 13)
The shape after:  (3420, 13)
```

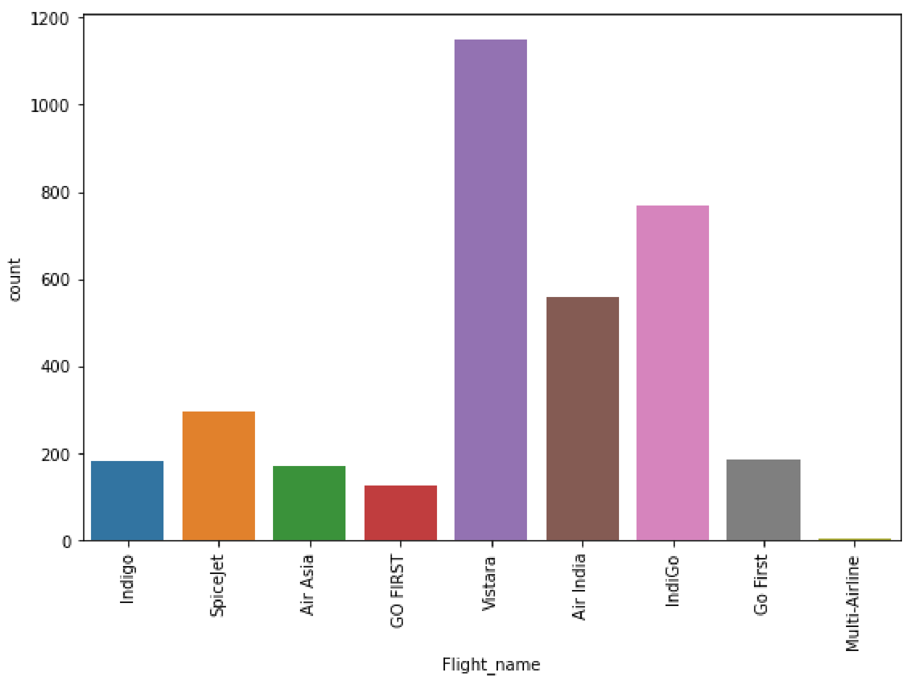
Data Loss:-

```
loss=(3434-3420)/3434*100
print("for removing the  outliers our data loss is : ",round(loss,1),'%')
```

```
for removing the  outliers our data loss is :  0.4 %
```

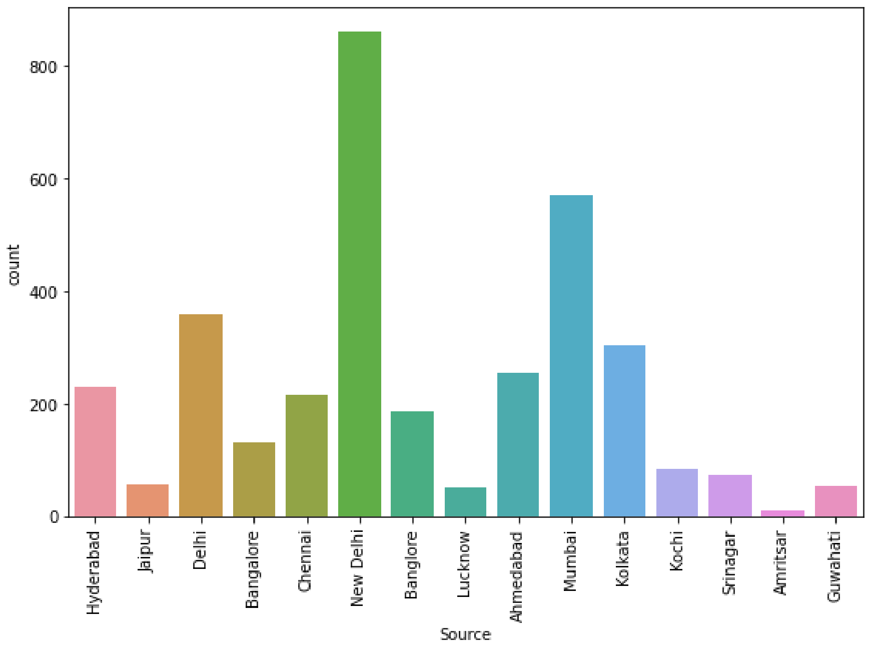
The data loss is very less, Let's move ahead.

Data Inputs- Logic- Output Relationships:-



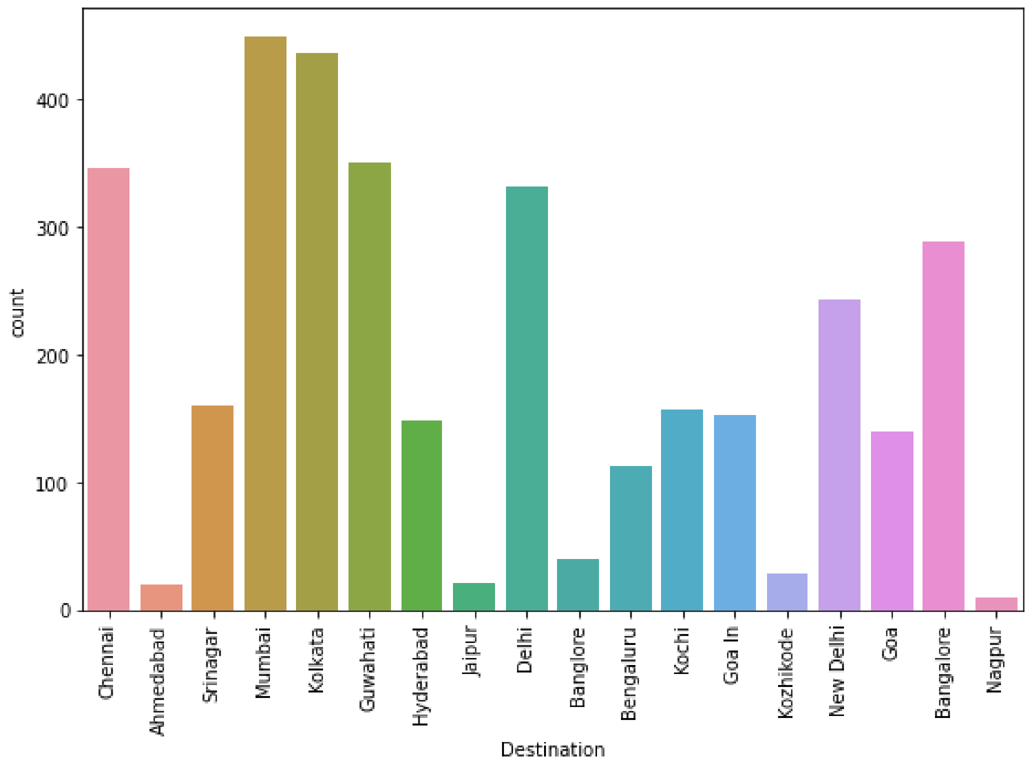
### Observations:-

- There are highest count of flights for Vistara Airways Airlines.
- Air_Asia, Air_Asia_I, Airasia Airlines is actually one Airline. so We include all of them and will convert them with AirAsia.



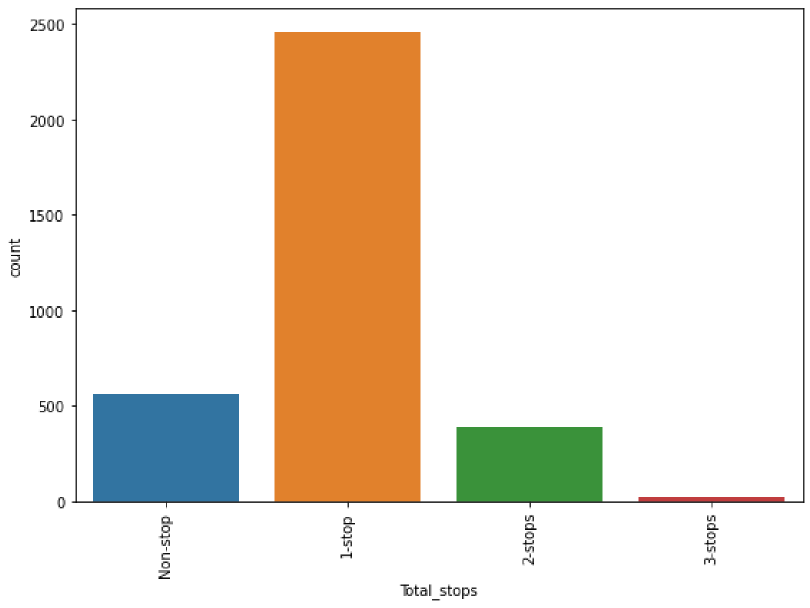
### Observations:-

- As we can see that most of the flights are taking off from New delhi.
- there are least number of flights are taking off from Amritsar.



### Observations:-

- Mumbai & Kolkata are most of the flight's destination.
- Ahmedabad & Nagpur are least flight's destination.

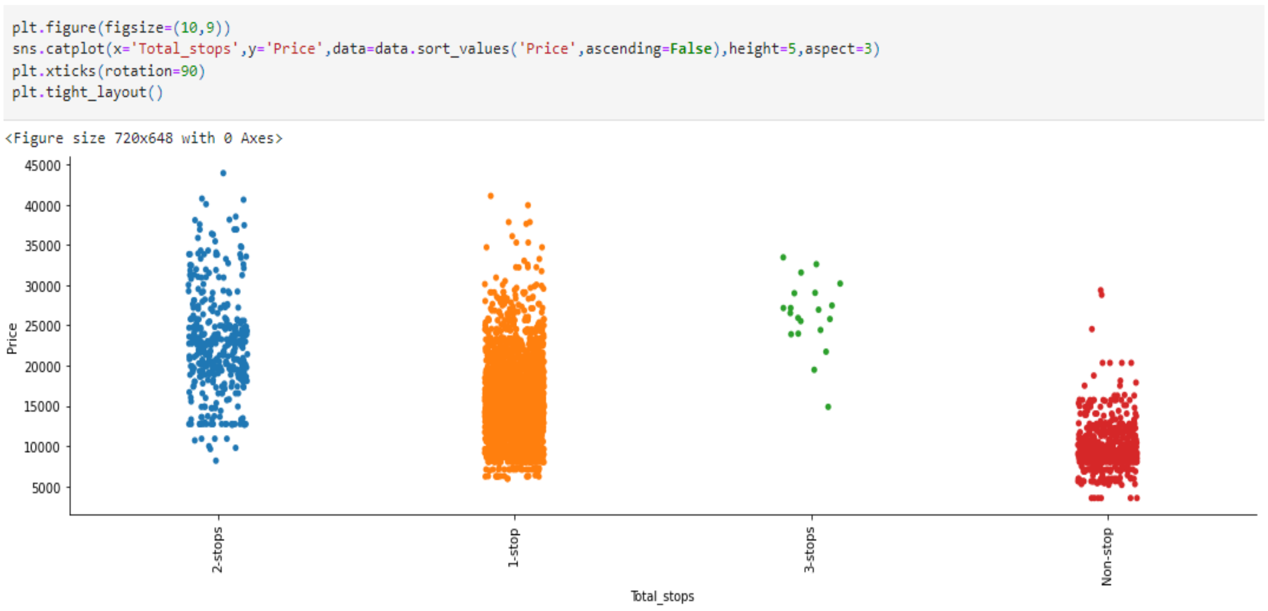


## Observation:-

- Most of the flights are taking at least 1 stops during the journey.
- There is minimum number of flights are taking 3-stops.

Feature VS Target:-

Relation between Total_Stops and Price



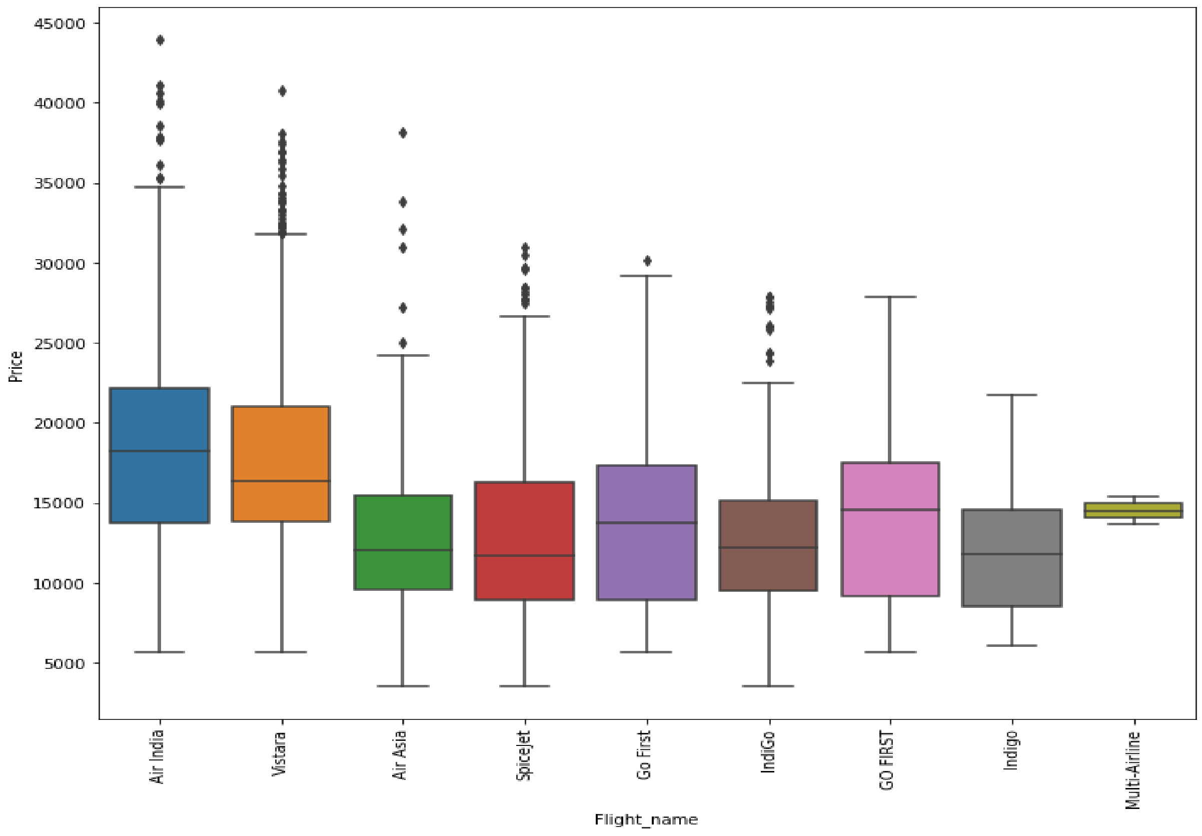
```
plt.figure(figsize=(10,9))
sns.catplot(x='Total_stops',y='Price',data=data.sort_values('Price',ascending=False),height=5,aspect=3)
plt.xticks(rotation=90)
plt.tight_layout()
```

```
<Figure size 720x648 with 0 Axes>
```

Observation:-

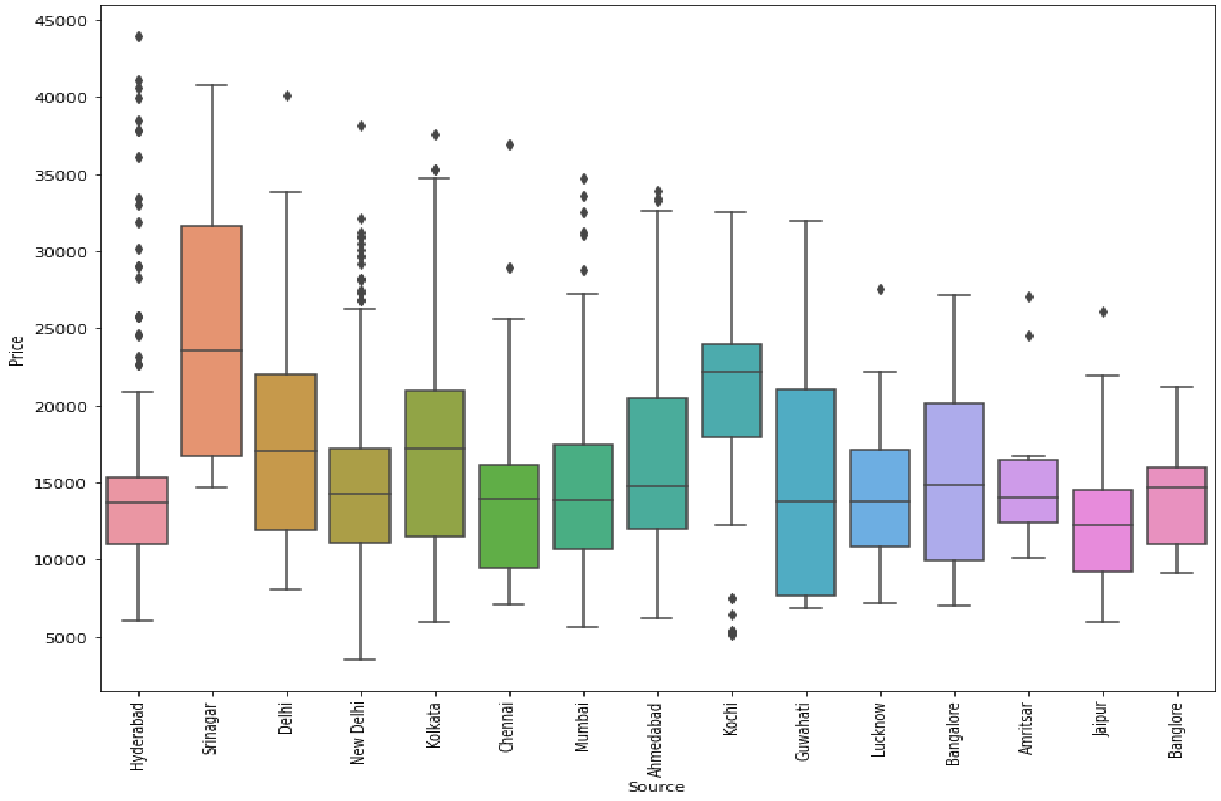
- We can see that Non-Stop flights price are less as compare to others.
- If a flight takes 2-stops during the journey, having highest price of flights.
- If flight takes 1-stop during the journey, the price of the flights falling between median range.

`Flights_name vs Price:`



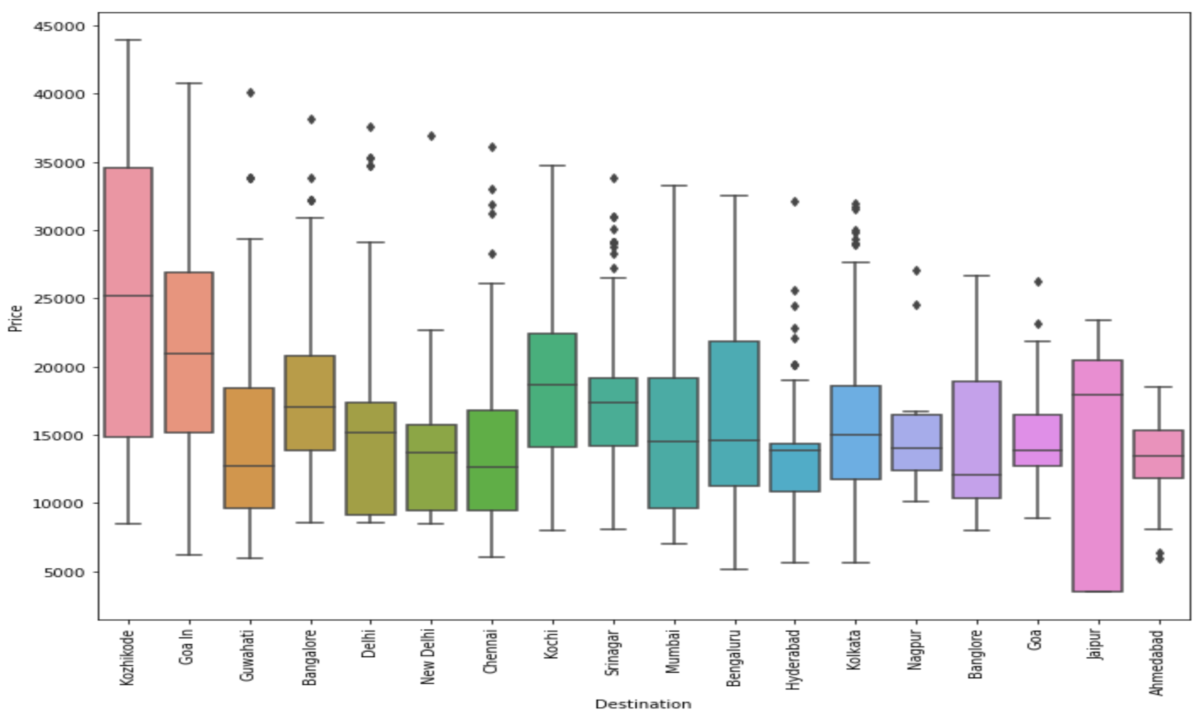
- Air India airline having highest prices among all of them.
- Indigo,Spicejet,Air Asia airline having least prices among all of them.

Source vs Price:

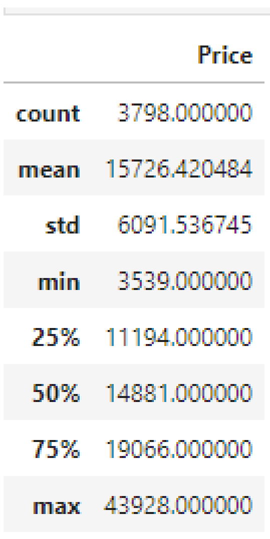


Here, we can see that the up-trend. It mean there is some relation between these features.

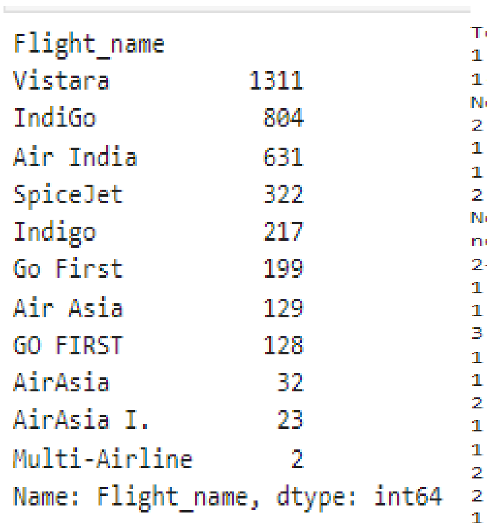
Destination Vs Price:



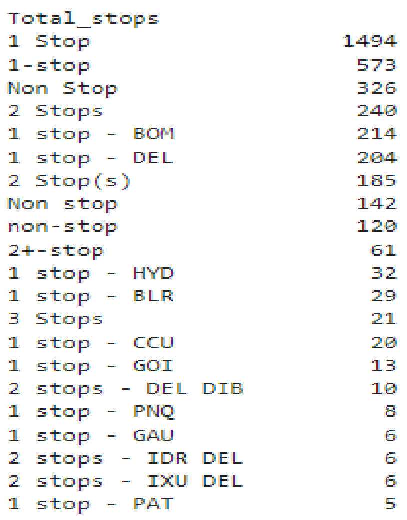
State the set of assumptions (if any) related to the problem under consideration:-



	Price
count	3798.000000
mean	15726.420484
std	6091.536745
min	3539.000000
25%	11194.000000
50%	14881.000000
75%	19066.000000
max	43928.000000



```
Flight_name
Vistara          1311
IndiGo            804
Air India         631
SpiceJet          322
Indigo            217
Go First          199
Air Asia          129
GO FIRST          128
AirAsia            32
AirAsia I.         23
Multi-Airline       2
Name: Flight_name, dtype: int64
```



```
Total_stops
1 Stop                  1494
1-stop                   573
Non Stop                 326
2 Stops                  240
1 stop - BOM             214
1 stop - DEL             204
2 Stop(s)                185
Non stop                 142
non-stop                 120
2+-stop                   61
1 stop - HYD              32
1 stop - BLR              29
3 Stops                   21
1 stop - CCU              20
1 stop - GOI              13
2 stops - DEL DIB         10
1 stop - PNQ               8
1 stop - GAU               6
2 stops - IDR DEL          6
2 stops - IXU DEL          6
1 stop - PAT               5
```

## Assumptions:-

- As we could see that our dataset having all object data type data. So that's why we are unable to get their statistical summary.

- As we can see that the minimum price of any Airline is 3549 inr and Max price of any airline ticket is 43928. There is big difference we can see. We will take care of it. It may be possible that our dataset contains outliers. We will see that in further steps.
- We see that in Airline name feature, Air Asia, AirAsia, AirAsia I. flights are different name of Air Asia flight. So we will convert them in one as Air Asia.
- Also we see that Total stops feature are showing different name in similar stops. We have to handle them also.

Hardware and Software Requirements and Tools Used:-

The Flight Prediction Project is about built a machine learning model that could predict the default case so that the company could to whom they should provide the loan amount and whom they should not.

So for that we require lots of libraries and packages to work upon this project.

Hardware Required:-

- Processor:- Core i5 or above
- Ram:- 8GB or above
- SSD:- 256GB or Above

Software Required:-

- Anaconda Promt/Nevigator

Libraries Required:-

- Pandas
- Numpy
- Matplotlib
- Seaborn
- Sklearn.preprocessing.StandardScaler
- Sklearn.preprocessing.LabelEncoder
- Sklearn.preprocessing.StandardScaler
- from sklearn.model_selection import train_test_split,cross_val_score
- sklearn.linear_model.LinearRegression
- sklearn.neighbors.KNeighborsRegressor
- Sklearn.ensemble.RandomForestRegressor,AdaBoostRegressor,GradientBoostingRegressor
- XGB Reressor

Model/s Development and Evaluation

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

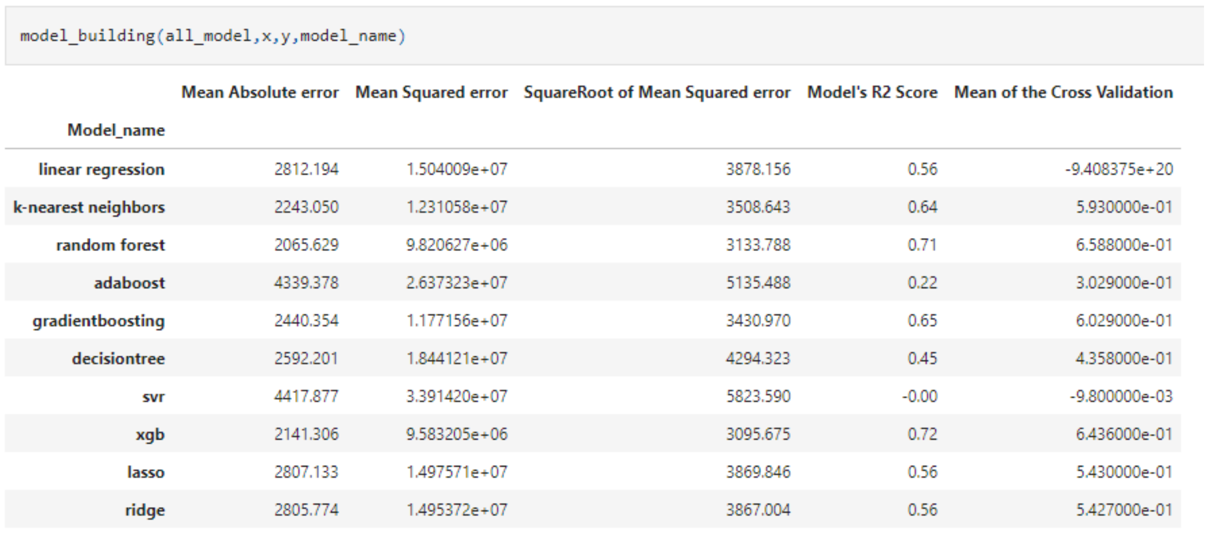
During the statistical analyzation of the data distribution we found data was little bit skewed data and had some of outliers and we solved it by applying power transformation technique of Scikit learn on the data and we were succeeded significantly in doing so. We also found some not useful column in the data set and we removed them with the help of pandas. We had sorted the Date column with the help of pandas. We had used ROC plot to evaluate the best model and its selection. To Hyper tune the model we used scikit learn's GridsearchCV method.

Testing of Identified Approaches (Algorithms):-

As per the Flight price prediction project use case demanded the prediction of the default case, we analysed the data and found that the problem is of Supervised Machine Learning Classification problem. Hence we decided to use the following algorithms to build the model for the use case:

- ➢ Linear Regression
- ➢ Lasso Regression.
- ➢ Ridge Regression.
- ➢ Random Forest Regressor.
- ➢ Select Vector Machine Reggressor.
- ➢ XGB Regressor
- ➢ Ada Boost Regressor
- ➢ Gradient Boosting Regressor.

Run and Evaluate selected models:-



```
model_building(all_model,x,y,model_name)
```

Model_name	Mean Absolute error	Mean Squared error	SquareRoot of Mean Squared error	Model's R2 Score	Mean of the Cross Validation
linear regression	2812.194	1.504009e+07	3878.156	0.56	-9.408375e+20
k-nearest neighbors	2243.050	1.231058e+07	3508.643	0.64	5.930000e-01
random forest	2065.629	9.820627e+06	3133.788	0.71	6.588000e-01
adaboost	4339.378	2.637323e+07	5135.488	0.22	3.029000e-01
gradientboosting	2440.354	1.177156e+07	3430.970	0.65	6.029000e-01
decisiontree	2592.201	1.844121e+07	4294.323	0.45	4.358000e-01
svr	4417.877	3.391420e+07	5823.590	-0.00	-9.800000e-03
xgb	2141.306	9.583205e+06	3095.675	0.72	6.436000e-01
lasso	2807.133	1.497571e+07	3869.846	0.56	5.430000e-01
ridge	2805.774	1.495372e+07	3867.004	0.56	5.427000e-01

<u>Key Metrics for success in solving problem under consideration:-</u>

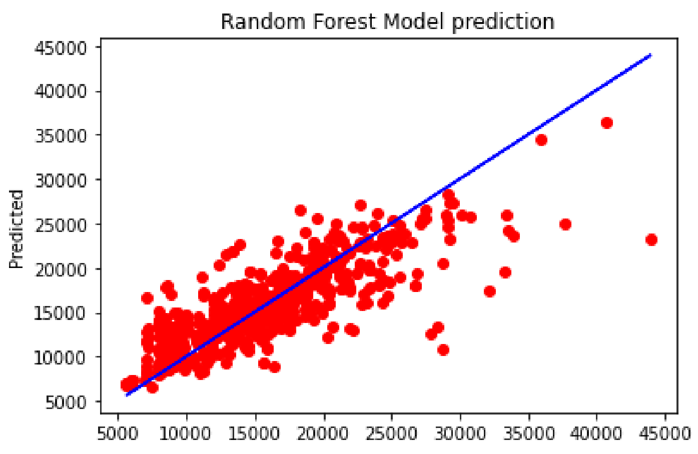
Model Evaluation For Random Forest

```
rf=RandomForestRegressor(n_estimators=100,min_samples_split=3,min_samples_leaf=2,,max_depth=8)
rf.fit(x_train,y_train)
rf.predict(x_train)
y_pred=rf.predict(x_test)
score=r2_score(y_test,y_pred)
print("\nRandom Forest Regressor R2 Score is : ",score)
print("\nMean Squared Erros is : ",MSE(y_test,y_pred))
print("\nMean Abosute Erros is : ",MAE(y_test,y_pred))
print("\nRoot Mean Squared Error is: ",np.sqrt(MSE(y_test,y_pred)))
```

```
Random Forest Regressor R2 Score is :  0.7073356666094661

Mean Squared Erros is :  9727760.456666946

Mean Abosute Erros is :  2011.3886012611088

Root Mean Squared Error is:  3118.9357891221402
```

```
## plotting graph
plt.scatter(x=y_test,y=y_pred,color='r')
plt.plot(y_test,y_test,color='b')
plt.xlabel('Acutal ')
plt.ylabel('Predicted')
plt.title('Random Forest Model prediction')
```

```
Text(0.5, 1.0, 'Random Forest Model prediction')
```



Hyperparameter tunning for XGB Regressor

```
x_train,x_test,y_train,y_test=tts(x,y, test_size=0.25,random_state=100)
grid_param={'learning_rate':[0.0001,0.001,0.01,0.1,1],
            'gamma':[0.1,0.001,0.2,0.3,0.4,0.5,0.6,0.7],
            'colsample_bytree':[0.3,0.5,0.7]}
```

```
Gcv=GridSearchCV(XGBRegressor(),grid_param,cv=5,n_jobs=-1)
Gcv.fit(x_train,y_train)
```

```
Gcv.best_params_
```

```
{'colsample_bytree': 0.7, 'gamma': 0.1, 'learning_rate': 0.1}
```

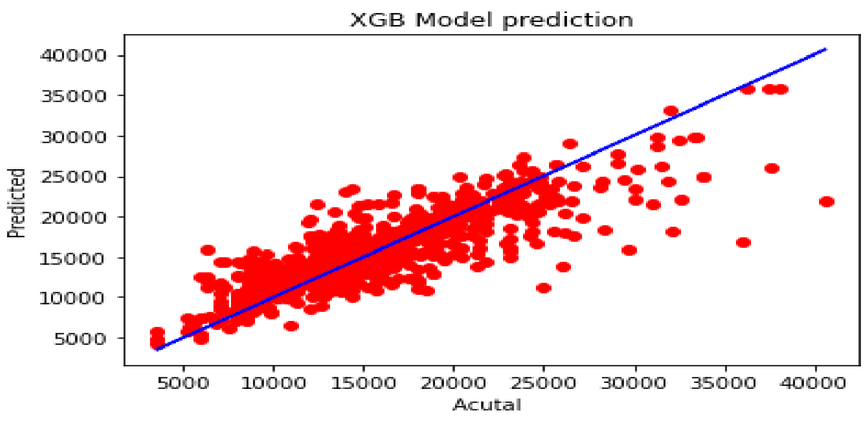
Model Evaluation of XGB Regressor

```
xgb=XGBRegressor(colsample_bytree=0.6,gamma=0.1,learning_rate=0.1)
xgb.fit(x_train,y_train)
xgb.predict(x_train)
pred_y=xgb.predict(x_test)
score=r2_score(y_test,pred_y)
print("\nRandom Forest Regressor R2 Score is : ",score)
print("\nMean Squared Erros is : ",MSE(y_test,pred_y))
print("\nMean Abosute Erros is : ",MAE(y_test,pred_y))
print("\nRoot Mean Squared Error is: ",np.sqrt(MSE(y_test,pred_y)))
```

```
Random Forest Regressor R2 Score is :  0.7218376742432178

Mean Squared Erros is :  9534698.315718545

Mean Abosute Erros is :  2109.7378917671786

Root Mean Squared Error is:  3087.8306811932785
```

```
## plotting graph
plt.scatter(x=y_test,y=pred_y,color='r')
plt.plot(y_test,y_test,color='b')
plt.xlabel('Acutal ')
plt.ylabel('Predicted')
plt.title('XGB Model prediction')
```

```
Text(0.5, 1.0, 'XGB Model prediction')
```



1. Cross Validation:

Cross-validation helps to find out the overfitting 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. R Squared :

The R2 score is a very important metric that is used to evaluate the performance of a regression-based machine learning model. It is pronounced as R squared and is also known as the coefficient of determination. It works by measuring the amount of variance in the predictions explained by the dataset.

In other fields, the standards for a good R-Squared reading can be much higher, such as 0.9 or above. In finance, an R-Squared above 0.7 would generally be seen as showing a high level of correlation, whereas a measure below 0.4 would show a low correlation.

R-squared measures the strength of the relationship between your model and the dependent variable on a convenient 0 – 100% scale. After fitting a linear regression model, you need to determine how well the model fits the data.

Best Fit Model

As we can see that XGB Regressor Model gives us the best accuracy. So we will finalized this as a best fit model.

finding the best random state for xgb:

```
def random_state(feature,target):
    max_r2=0
    for i in range(1,101):
        x_train,x_test,y_train,y_test=tts(feature,target,test_size=0.20,random_state=i)
        lr=XGBRegressor()
        lr.fit(x_train,y_train)
        pred=lr.predict(x_test)
        score=r2_score(y_test,pred)
        if score>max_r2:
            max_r2=score

    return i
```

```
random_state(x,y)
```

```
100
```

```
x_train,x_test,y_train,y_test=tts(x,y, test_size=0.25,random_state=100)
```

```
xgb=XGBRegressor(colsample_bytree=0.6,gamma=0.1,learning_rate=0.1)
xgb.fit(x_train,y_train)
xgb.predict(x_train)
pred_y=xgb.predict(x_test)
score=r2_score(y_test,pred_y)
```

```
## printing the scores
print("\nXGB Regressor R2 Score is : ",score)
print("\nMean Squared Erros is : ",MSE(y_test,pred_y))
print("\nMean Abosute Erros is : ",MAE(y_test,pred_y))
print("\nRoot Mean Squared Error is: ",np.sqrt(MSE(y_test,pred_y)))
```

```
XGB Regressor R2 Score is :  0.7218376742432178

Mean Squared Erros is :  9534698.315718545

Mean Abosute Erros is :  2109.7378917671786

Root Mean Squared Error is:  3087.8306811932785
```

3.PCA:-

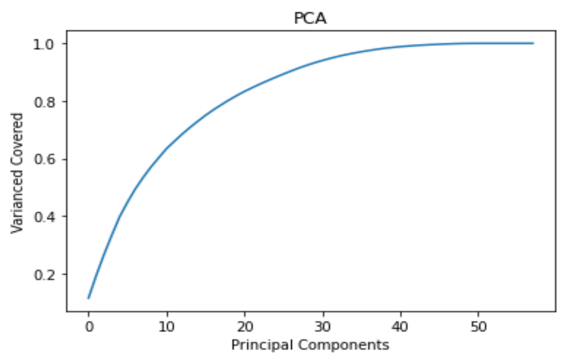
```
from sklearn.decomposition import PCA
pca=PCA()
```

```
pca.fit_transform(x)
```

```
array([[-4.17440748e-01,  8.28338140e-01,  9.48665054e-01, ...,
         1.46987179e-16, -3.98703634e-17, -3.40154765e-17],
       [-2.67280865e-01,  9.01165781e-01,  8.77353520e-01, ...,
         1.36476521e-16, -5.77204359e-16,  2.11008172e-16],
       [-2.52896486e-01,  9.21805496e-01,  9.07153835e-01, ...,
         4.68441792e-17, -1.82467547e-16,  1.91363561e-17],
       ...,
       [ 3.57964156e-01, -4.81310442e-01, -2.62951119e-01, ...,
        -2.55298668e-17, -1.29928760e-17,  2.28882837e-17],
       [-4.34343492e-01,  6.51445309e-01, -6.89940038e-01, ...,
        -1.45004444e-17,  4.24514047e-18,  3.28938794e-18],
       [-2.99663177e-01,  5.03868816e-01, -6.77427933e-01, ...,
        -7.17127888e-18, -3.56104713e-18,  8.24929347e-18]])
```

Plotting the scree plot:

```
plt.figure()
plt.plot(np.cumsum(pca.explained_variance_ratio_))
plt.xlabel('Principal Components')
plt.ylabel('Varianced Covered')
plt.title('PCA')
plt.show()
```



we can see 35 Principal components are able to explain our data more than 95%. Hence we will take only 35 components.

```
pca=PCA(n_components=35)
new_comp=pca.fit_transform(x)
x_comp=pd.DataFrame(new_comp,columns=['PC1','PC2','PC3','PC4','PC5','PC6','PC7','PC8','PC9','PC10',
                                      'PC11','PC12','PC13','PC14','PC15','PC16','PC17','PC18','PC19','PC20',
                                       'PC21','PC22','PC23','PC24','PC25','PC26','PC27','PC28','PC29','PC30',
                                      'PC31','PC32','PC33','PC34','PC35'])

x_comp
```

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10	PC11	PC12	PC13	PC14	PC15	PC16
0	-0.417441	0.828338	0.948665	-0.320994	0.637881	0.163929	0.393662	0.277286	0.501755	-0.672958	0.310441	-0.635682	0.718432	0.007805	0.588190	0.527067
1	-0.267281	0.901166	0.877354	-0.285828	0.583356	-0.030310	0.455518	0.237172	0.601156	-0.606756	-0.003949	-0.653642	0.952800	-0.010552	0.081518	0.393236
2	-0.252896	0.921805	0.907154	-0.314162	0.561347	0.060662	0.416086	0.267342	0.641168	-0.653235	0.065096	-0.707803	0.719617	0.173254	0.409728	0.517554
3	-0.254694	0.942441	0.915812	-0.339527	0.574247	-0.131842	0.415521	0.245239	0.520737	-0.634787	0.130565	-0.378392	0.783681	-0.333199	0.235649	0.808719
4	-0.253515	0.979382	0.901244	-0.367183	0.529781	-0.083953	0.189649	0.315742	0.347504	-0.125415	-0.100205	-0.416383	0.148987	0.131540	0.445082	-0.041497
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
3415	0.554985	-0.642429	0.073455	0.523519	-0.364278	-0.332081	0.599483	0.501236	0.665908	-0.384966	-0.309169	0.141432	-0.084401	0.615374	0.003392	-0.624910
3416	0.702110	0.107423	-0.479759	-0.387900	-0.511344	-0.079420	1.054224	-0.469224	-0.060447	0.536340	0.156348	0.237091	-0.177709	0.065828	-0.220417	0.110672
3417	0.357964	-0.481310	-0.262951	0.578915	-0.316309	-0.153614	0.852674	-0.413866	0.325950	0.092444	-0.439431	-0.023585	-0.444289	-0.361554	-0.337462	0.098623
3418	-0.434343	0.651445	-0.689940	-0.204181	0.498954	0.169771	0.219205	0.670505	-0.780079	-0.190093	0.500750	0.127469	-0.055832	0.043662	0.227943	-0.195787
3419	-0.299663	0.503869	-0.677428	-0.277119	0.734376	0.179303	0.169401	0.894570	-0.903864	0.156480	0.475399	-0.020058	0.000648	0.088065	0.184574	-0.102564

3420 rows × 35 columns

Now we will give this to model and check the scores.

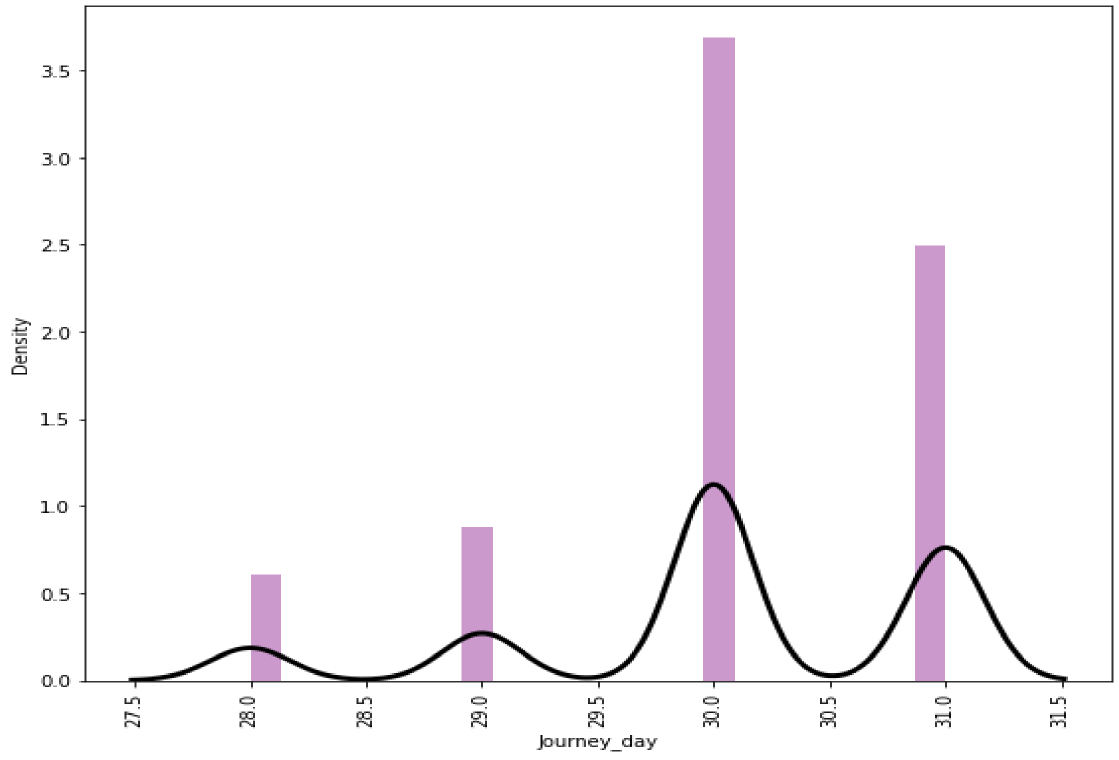
With Using PCA the All Model's Performance

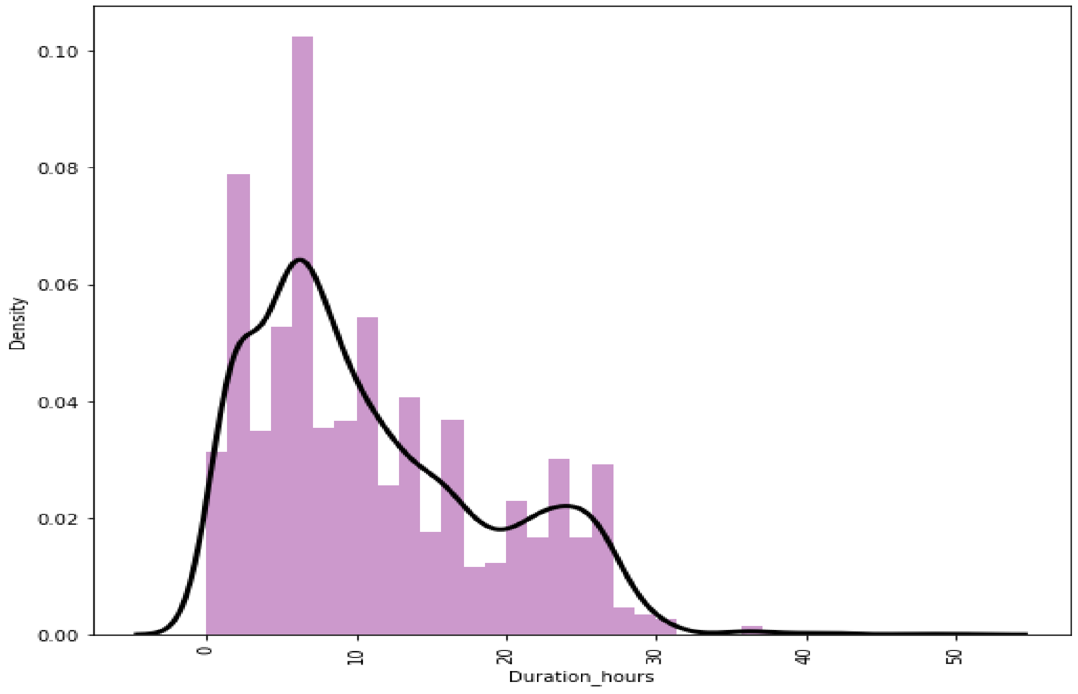
```
model_building(all_model,x_comp,y,model_name)
```

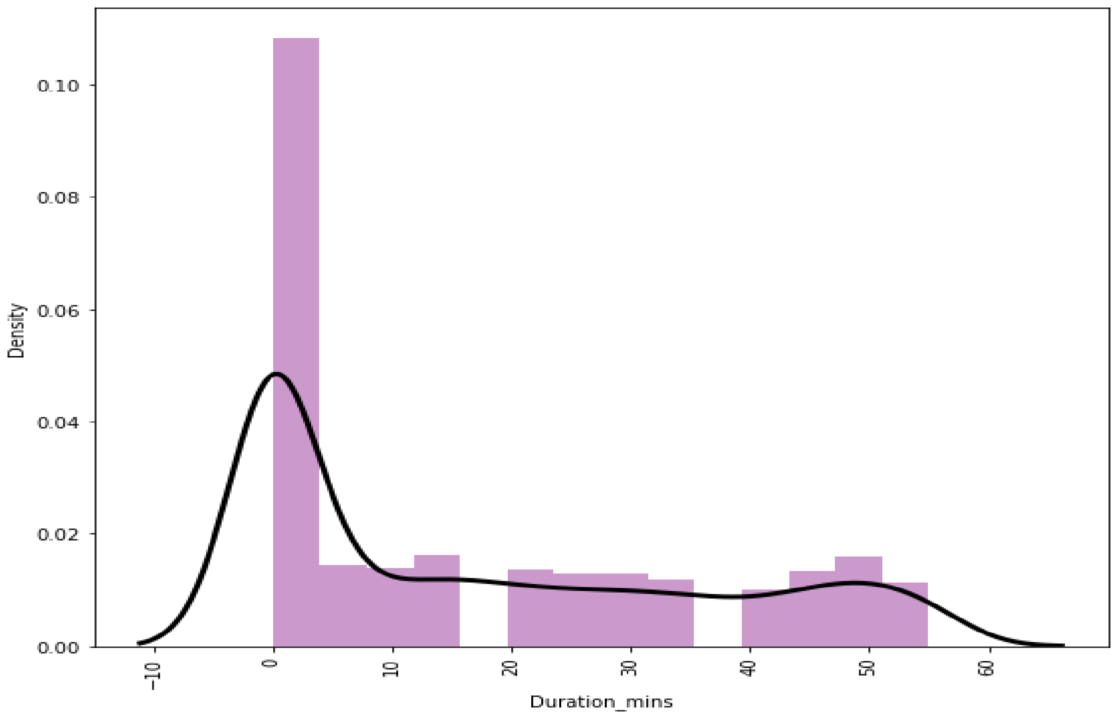
	Mean Absolute error	Mean Squared error	SquareRoot of Mean Squared error	Model's R2 Score	Mean of the Cross Validation
Model_name					
linear regression	2927.369	1.527880e+07	3908.810	0.54	0.5084
k-nearest neighbors	2203.189	1.189835e+07	3449.399	0.64	0.6086
random forest	2175.424	1.076267e+07	3280.651	0.68	0.6336
adaboost	3872.207	2.229439e+07	4721.694	0.33	0.3367
gradientboosting	2580.509	1.272830e+07	3567.674	0.62	0.5860
decisiontree	2776.181	2.009894e+07	4483.184	0.40	0.2570
svr	4434.646	3.345370e+07	5783.917	-0.01	-0.0138
xgb	2297.610	1.199743e+07	3463.731	0.64	0.5797
lasso	2926.783	1.527081e+07	3907.788	0.54	0.5085
ridge	2926.654	1.526958e+07	3907.631	0.54	0.5085

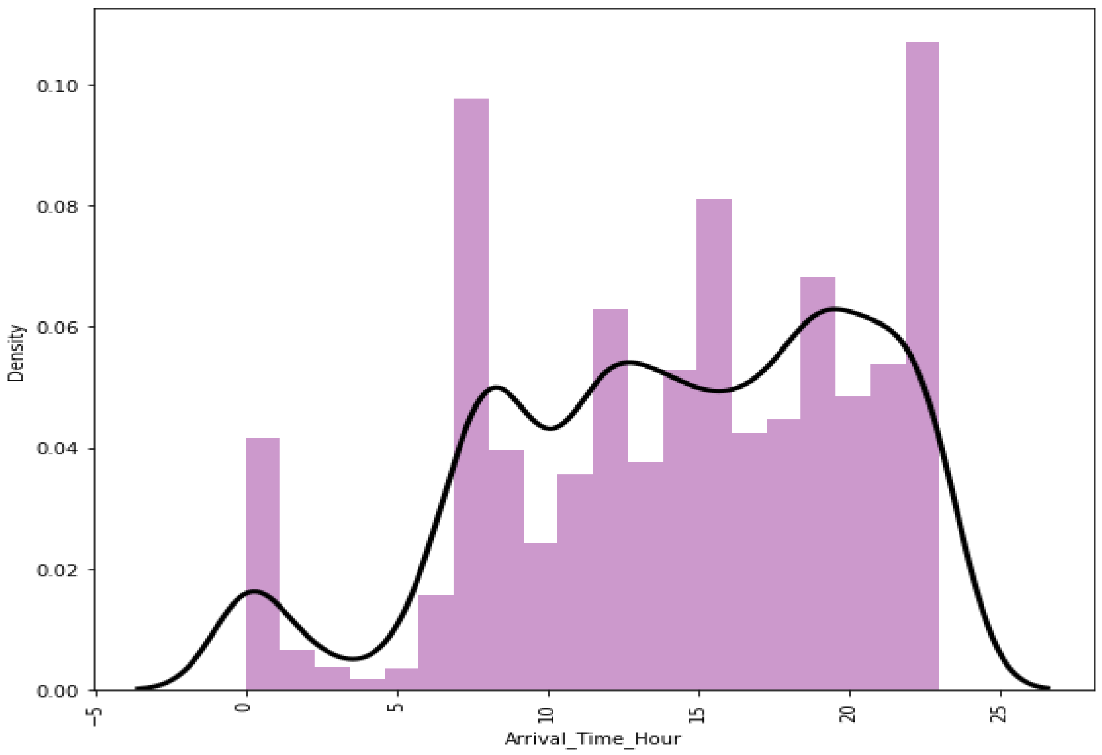
Visualizations:-

Checking Distribution Through Graph:-









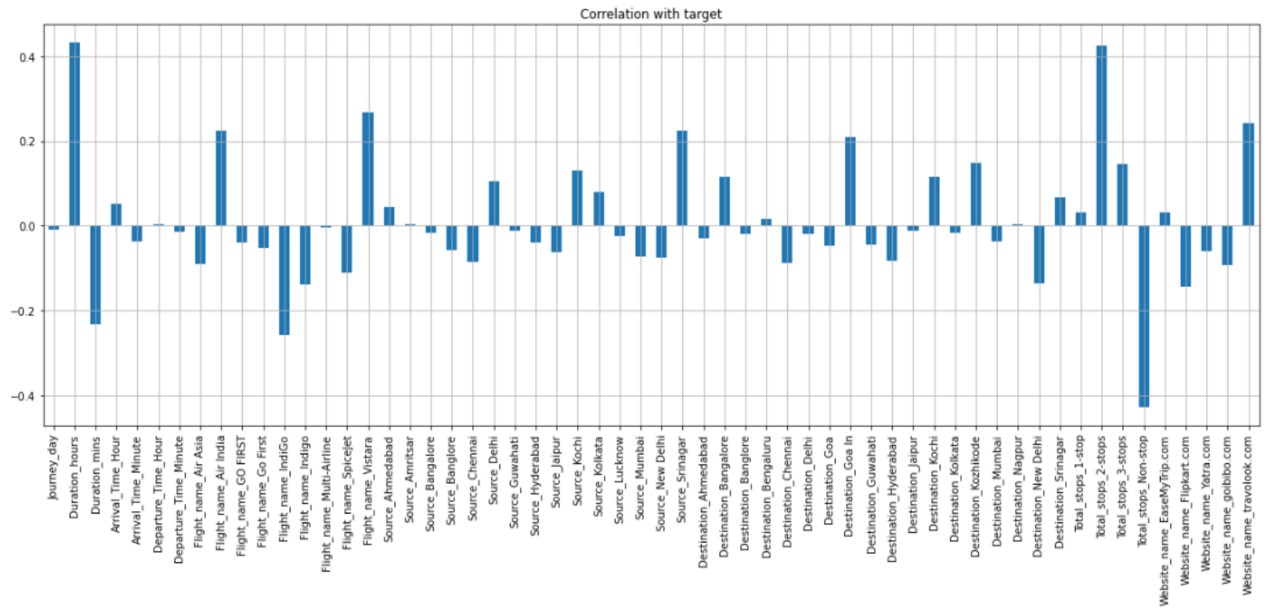
### Observations:-

- Maximum flights take 1 to 10 hours during the journey.
- There are least number of flights take more the 30 hours during the journey.
- As we can see that as number of duration hours increases the price are also increases. There are tight relation duration hours and Price.
- Most of the flights are takeoff between 8 AM to 24 PM.
- There are least number of flights are takeoff between 1 to 7 AM.

Relation With Target:-

`Correlation With Target:-`

```
df.drop('Price',axis=1).corrwith(df.Price).plot(kind='bar',grid=True,figsize=(17,8),title='Correlation with target')
plt.tight_layout()
```



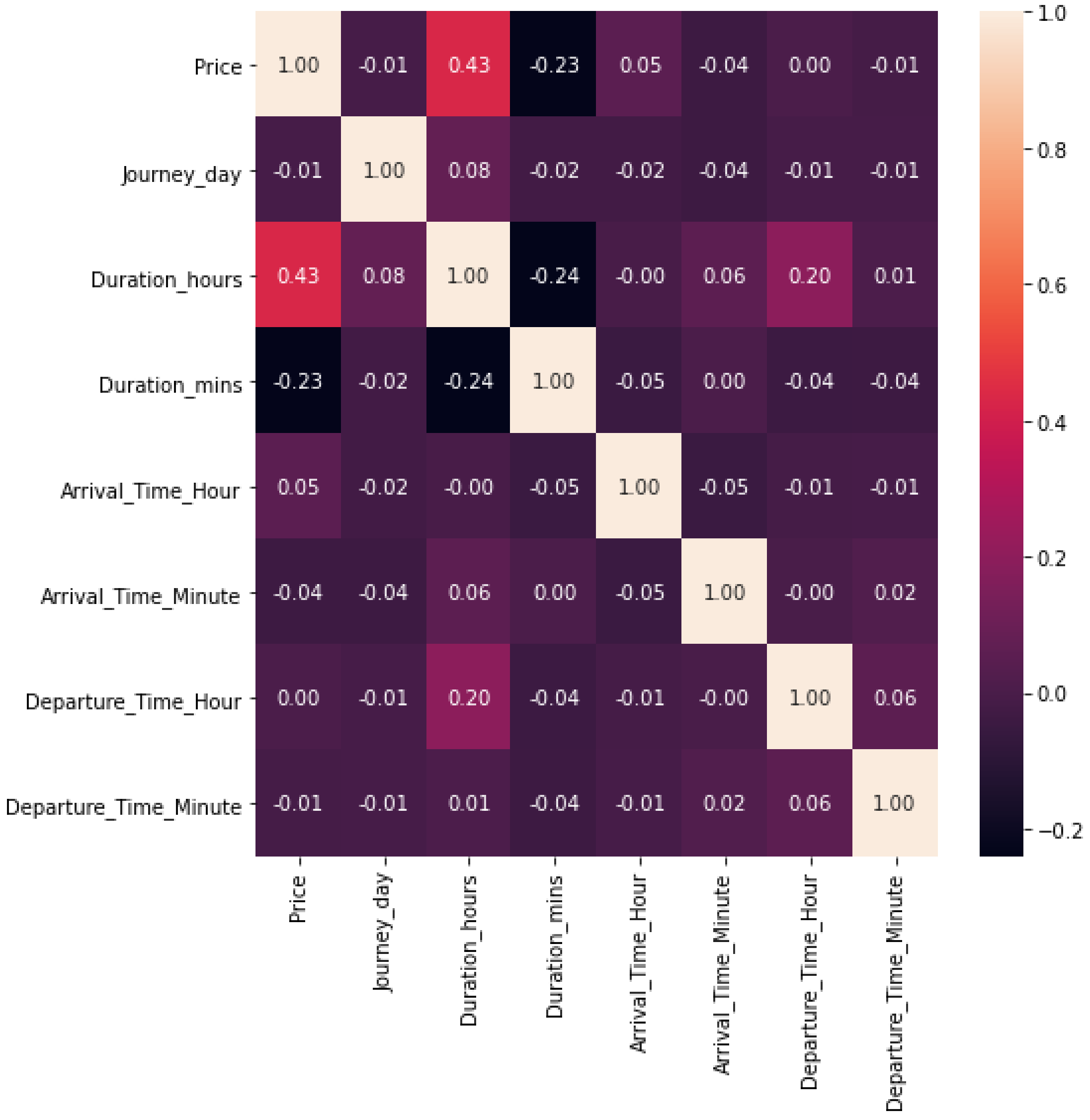
Observations of above correlation:-

- 'Duration_hours','Duration_mins','Flight_name_Air_India','Flight_name_Vistara','Source_Srinagar','Total-stops_non_stops',and 'Website_name_travaolook.com'are highly correlated with target variable.
- 'Journey-day','Departure-Time_Hour','Flight_name_multi-Airline','Source_Amritsar', and 'Destination_nagpur' are very less correlated with target variable. We will use feature selection technique to select best feature:-

Interpretation of the Results

Heatmap:-

```
plt.figure(figsize=(8,8))
sns.heatmap(df_new.corr(),annot=True,
            linecolor='b',fmt='.2f')
plt.tight_layout()
```



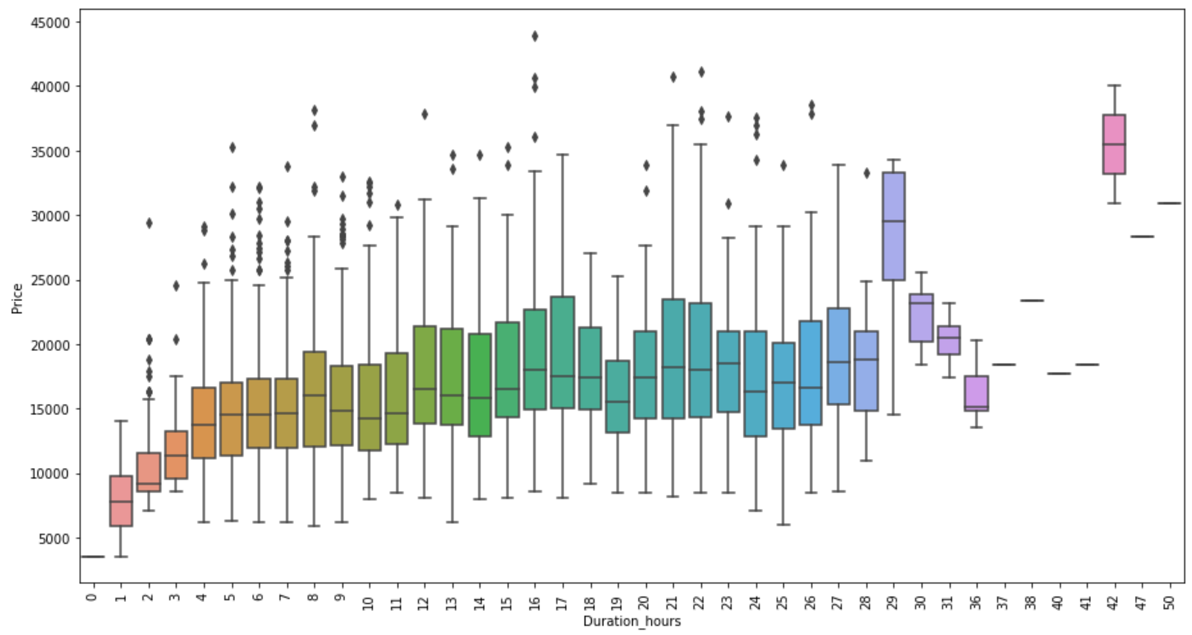
	Price	Journey_day	Duration_hours	Duration_mins	Arrival_Time_Hour	Arrival_Time_Minute	Departure_Time_Hour	Departure_Time_Minute
Price	1.00	-0.01	0.43	-0.23	0.05	-0.04	0.00	-0.01
Journey_day	-0.01	1.00	0.08	-0.02	-0.02	-0.04	-0.01	-0.01
Duration_hours	0.43	0.08	1.00	-0.24	-0.00	0.06	0.20	0.01
Duration_mins	-0.23	-0.02	-0.24	1.00	-0.05	0.00	-0.04	-0.04
Arrival_Time_Hour	0.05	-0.02	-0.00	-0.05	1.00	-0.05	-0.01	-0.01
Arrival_Time_Minute	-0.04	-0.04	0.06	0.00	-0.05	1.00	-0.00	0.02
Departure_Time_Hour	0.00	-0.01	0.20	-0.04	-0.01	-0.00	1.00	0.06
Departure_Time_Minute	-0.01	-0.01	0.01	-0.04	-0.01	0.02	0.06	1.00

Observations:-

- Duration hours are highest correlated with target variable.
- Duration_mins are second highest correlated with target variable.
- Rest of the features are very least correlated with target variable.

```
Duration_hours Vs Price:
```

```
plt.figure(figsize=(13,7))
sns.boxplot(x='Duration_hours',y='Price',data=data.sort_values('Price',ascending=True))
plt.xticks(rotation=90)
plt.tight_layout()
```



- As we can see that as number of duration hours increases the price are also increases. There are tight relation duration hours and Price.

A proper implementation of this project can result in saving money of inexperienced people by providing them the information related to trends that flight prices follow and also give them a predicted value of the price which they use to decide whether to book ticket now or later. In conclusion this type of service can be implemented with good accuracy of prediction. As the predicted value is not fully accurate there is huge scope for improvement of these kind of service.

CONCLUSION

Key Findings and Conclusions of the Study

From Whole Dataset and Analysis we find some conclusions:

- Dataset had 3798 records of telecom users with 10 types of the services related data were present in the data.

- Dataset has 9 object datatype columns and rest of the features are float datatype which is our target variable.
- There are no null values in the dataset.
- There may be some feature may be converted in some meaningful manner.
- The Target variable is little bit right skewed.
- For some features, there may be values which might not be realistic. You may have to observe them and treat them with a suitable explanation.
- You might come across outliers in some features which you need to handle as per your understanding. Keep in mind that data is expensive and we cannot lose more than 7-8% of the data.

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

From the data collected and through exploratory data analysis, we can determine the following:

● The trend of flight prices vary over various day and across the may month.

● There are two groups of airlines: the economical group and the luxurious group. Spicejet, AirAsia, IndiGo, Go Air are in the economical class, whereas Jet Airways and Air India in the other. Vistara has a more spread out trend.

● The airfare varies depending on the time of departure, making timeslot used in analysis is an important parameter.

The work is a product of the intellectual environment of the whole team; and that all members have contributed in various degrees to the analytical methods used, to the research concept, and to the experiment design along with writing the manuscript.

We declare that we have no significant competing financial, professional or personal interests that might have influenced the performance or presentation of the work described in this manuscript.

Limitations of this work and Scope for Future Work

● More routes can be added and the same analysis can be expanded to major airports and travel routes in India.

● The analysis can be done by increasing the data points and increasing the historical data used. That will train the model better giving better accuracies and more savings.

● More rules can be added in the Rule based learning based on our understanding of the industry, also incorporating the offer periods given by the airlines.

● Developing a more user friendly interface for various routes giving more flexibility to the users.In future may be the information about target variable increase then we will get some more accuracy . And accuracy will increase we will get better prediction.