FLIGHT TICKET PRICE PREDICTION PROJECT

Submitted by:

Anshul Dubey

ACKNOWLEDGMENT

I have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations.

I would like to extend my sincere thanks to all of them. I am highly indebted to Flip Robo Technologies for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project. I want to thank my SME **Khushboo Garg** for providing the Dataset and helping us to solve the problem and addressing out our Query in right time.

I would like to express my gratitude towards DataTrained and my parents & members of Flip Robo for their kind cooperation and encouragement which help me in completion of this project. I would like to express my special gratitude and thanks to industry persons for giving me such attention and time.

INTRODUCTION
Business Problem Framing

The Airline Companies is considered as one of the most enlightened industries using complex methods and complex strategies to allocate airline prices in a dynamic fashion. These industries are trying to keep their all-inclusive revenue as high as possible and boost their profit. Customers are seeking to get the lowest price for their ticket, while airline companies are trying to keep their overall revenue as high as possible and maximize their profit.

However, mismatches between available seats and passenger demand usually leads to either the customer paying more or the airlines company losing revenue. Airlines companies are generally equipped with advanced tools and capabilities that enable them to control the pricing process. However, customers are also becoming more strategic with the development of various online tools to compare prices across various airline companies. In addition, competition between airlines makes the task of determining optimal pricing is hard for everyone. 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 Time of purchase patterns (making sure last-minute purchases are expensive) Keeping the flight as full as they want it (raising prices on a flight which is filling up in order to reduce sales and hold back inventory for those expensive last-minute expensive purchases) So, this project involves collection of data for flight fares with other features and building a model to predict fares of flights.

Conceptual Background of the Domain Problem

A report says India's affable aeronautics industry is on a high development movement. India is the third-biggest avionics showcase in 2020 and the biggest by 2030. Indian air traffic is normal to cross the quantity of 100 million travellers by 2017,whereas there were just 81 million passengers in 2015.

Agreeing to Google, the expression" Cheap Air Tickets" is most sought in India. At the point when the white-collar class of India is presented to air travel, buyers searching at modest costs. Any individual who has booked a flight ticket previously knows how dynamically costs change. Aircraft uses advanced strategies called Revenue Management to execute a distinctive valuing strategy. The least expensive accessible ticket changes over a period thecostofa ticket might be high or low. This valuing method naturally modifies the toll as per the time like morning, afternoon or night. Cost may likewise change with the seasons like winter, summer and celebration seasons. The extreme goal of the carrier is to build its income yet on the opposite side purchaser is searching atthe least expensive cost.

Purchasers generally endeavor to purchase the ticket in advance to the take-off day.

From the customer point of view, determining the minimum price or the best time to buy a ticket is the key issue. The conception of"tickets bought in advance are cheaper" is no longer working (William Groves and Maria Gini, 2013). Itis possible that customers who bought a ticket earlier pay more thanthose who bought the same ticket later. Moreover,early purchasing implies a risk of commitment to a specific schedule that may need tobe changed usually for a fee. Most of the studies performed on the customer side focus on the problem of predicting optimal ticket purchase time using statistical methods. As noted by Y. Chen et al. (2015) [8], predicting the actual ticket price is a more difficult task than predicting an optimal ticket purchase time due to various reasons: absence of enough datasets, external factors influencing ticket prices, dynamic behaviour of ticket pricing, competition among airlines, proprietary nature of airlines ticket pricing policies etc.

Early prediction of the demand along a given route could help an airline company pre-plan the flights and determine appropriate pricing for the route. Existing demand prediction models generally try to predict passenger demand for a single flight/route and market share of an individual airline. Price discrimination allows an airline company to categorize customers based on their willingness to pay and thus charge them different prices. Customers could be categorized into different

groups based on various criteria such as business vs leisure, tourist vs normal traveller, profession etc. For example, business customers are willing to pay more as compared to leisure customers as they rather focus on service quality than price.
In a less competitive market, the market power of a given airline is stronger, and thus, it is more likely to engage in price discrimination. On the other hand, the higher the level of competition, the weaker of the market power of an airline, and then the less likely the chance of the airline fare increases. Because of the availability of other travel options (e.g., bus, train, car etc.). Airlines use price elasticity information to determine when to increase ticket prices or when to launch promotions so that the overall demand is increased

ANALYTICAL PROBLEM FRAMING

Mathematical/ Analytical Modelling of the Problem

We are building a model in Machine Learning to predict the actual value of the flight fares and decide whether to buy them or not. So, this model will help us to determine which variables are important to predict the price of variables & also how do these variables describe the price of the flight ticket. This will help to determine the price of tickets with the available independent variables. They can accordingly manipulate the strategy of the firm and concentrate on areas that will yield high returns.

Regression analysis is a set of statistical processes for estimating the relationships between a dependent variable (often called the 'outcome variable') and one or more independent variables (often called 'predictors', 'covariates', or 'features'). The most common form of regression analysis is linear regression, in which one finds the line (or a more complex linear combination) that most closely fits the data according to a specific mathematical criterion. For specific mathematical reasons this allows the researcher to estimate the conditional expectation of the dependent variable when the independent variables take on a given set of values.

Regression analysis is also a form of predictive modelling technique which investigates the relationship between a dependent (target) and independent variable (predictor). This technique is used for forecasting, time series modelling and finding the causal effect relationship between the variables.

The different Mathematical/Analytical models that are used in this project are as below:

1. **Linear regression** - is a linear model, e.g., a model that assumes a linear relationship between the input variables (x) and the single output variable (y). More specifically, that y can be calculated from a linear combination of the input variables (x).

2. **Lasso** - In statistics and machine learning, lasso is a regression analysis method that performs both variable selection and regularization in order to enhance the prediction accuracy and interpretability of the resulting statistical model.

3. **Ridge** - regression is a way to create a parsimonious model when the number of predictor variables in a set exceeds the number of observations, or when a data set has multi co linearity (correlations between predictor variables).

4. **K Neighbors Regressor** - KNN algorithm can be used for both classification and regression problems. The KNN algorithm uses 'feature similarity' to predict the values of any new data points. This means that the new point is assigned a value based on how closely it resembles the points in the training set.

5. **Decision Tree** - is one of the most commonly used, practical approaches for supervised learning. It can be used to solve both Regression and Classification tasks with the latter being put more into practical application. It is a tree-structured classifier with three types of nodes.

6. **Random forest** - is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. A Random Forest's nonlinear nature can give it a leg up over linear algorithms, making it a great option.

7. **AdaBoost Regressor** - is a meta-estimator that begins by fitting a regressor on the original dataset and then fits additional copies of the regressor on the same dataset but where the weights of instances are adjusted according to the error of the current prediction.

8. **Gradient Boosting Regressor** - GB builds an additive model in a forward stage-wise fashion; it allows for the optimization of arbitrary differentiable loss functions. In each stage a regression tree is fit on the negative gradient of the given loss function.

9. **Support Vector Regressor** – SVR is a supervised learning algorithm, That is used to predict discrete values. SVR uses the same principle As the SVMs. The basic idea behind SVR is to fit best line. In SVR the Best fit line is the hyperplane that has the maximum number of points.

> First, use the dataset and do the EDA process, fitting the best model and saving the model.

Data Sources and their formats Data is collected from www.paytm.com/flights for timeframe of 10 days using selenium and saved in CSV file. Data is scrapped for flights on different route. Data is scrapped for Economy class.

Let's check the data now. Below I have attached the snapshot below to give an overview

```
## Fetching csv file:
df = pd.read_csv("flight_data_paytm.csv")
df.head()
```

	Unnamed: 0	flight_name	flight_id	start_time	end_time	travel_time	stops	fare	source	destination
0	0	SpiceJet	SG - 534	22:30	01:15	2h 45m	Non Stop	8,160	DEL-Delhi	BLR-Bengaluru
1	1	IndiGo	6E - 2036	22:50	01:45	2h 55m	Non Stop	8,160	DEL-Delhi	BLR-Bengaluru
2	2	Go First	G8 - 275	21:45	03:00	5h 15m	1 stop at Pune	8,160	DEL-Delhi	BLR-Bengaluru
3	3	Go First	G8 - 2511	10:45	16:05	5h 20m	1 stop at Patna	8,160	DEL-Delhi	BLR-Bengaluru
4	4	Go First	G8 - 165	10:00	16:05	6h 5m	1 stop at Patna	8,160	DEL-Delhi	BLR-Bengaluru

Data description

Data contains 4322 entries, each having 10 variables. The details of the features are given below:
1. flight_name :- Airlines company name
2. . flight_id: -flight's unique id
3. End_time:- Flight's arrival time for destination.
4. . travel_time:- Time taken by flight to reach source from destination.
5. . Stops: - route via flight goes from source to destination, 0 means nonstop, greater than 0 means taking stops.
6. . fare: - flight's ticket price
7. . source: - From where flight takes off. 9. Destination: - Place, where flight lands.

Checking the data type & info of dataset.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4322 entries, 0 to 4321
Data columns (total 10 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Unnamed: 0   4322 non-null   int64
 1   flight_name  4322 non-null   object
 2   flight_id    4322 non-null   object
 3   start_time   4322 non-null   object
 4   end_time     4322 non-null   object
 5   travel_time  4322 non-null   object
 6   stops        4322 non-null   object
 7   fare         4322 non-null   object
 8   source       4322 non-null   object
 9   destination  4322 non-null   object
dtypes: int64(1), object(9)
memory usage: 337.8+ KB
```

Checking the no. of null values in the dataset.

```
df.isnull().sum()
```

```
[9]: Unnamed: 0     0
     flight_name    0
     flight_id      0
     start_time     0
     end_time       0
     travel_time    0
     stops          0
     fare           0
     source         0
     destination    0
     dtype: int64
```

There is not any null values present in dataset.

Data Pre-processing

Data pre-processing in Machine Learning refers to the technique of preparing (cleaning and organizing) the raw data to make it suitable for a building and training Machine Learning models. In other words, whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis. Data pre-processing is an integral step in Machine Learning as the quality of data and the useful information that can be derived from it directly affects the ability of our model to learn;

therefore, it is extremely important that we pre- process our data before feeding it into our model.

Checking the value counts of categorical data

```
df['flight_name'].value_counts()
```

```
Vistara         2009
IndiGo          1495
Air India        447
Air Asia         163
SpiceJet         103
Go First          92
Alliance Air      11
FlyBig             2
Name: flight_name, dtype: int64
```

t number of flights are from Vistara followed by Indigo and Air India.

```
df['source'].value_counts()
```

```
DEL-Delhi               873
CU-Kolkata              587
BLR-Bengaluru           497
BOM-Mumbai              412
HYD-Hyderabad           357
MAA-Chennai             238
```

Observations:

1. Airlines company name -> Vistara, Air India, Indigo, Go First, SpiceJet, Air Asia are present in flight_name column.

2. There are 29 different route pair values present in source -destination.

3. Minimum stops are zero and maximum stops 4

.

Putting data into proper format:

```
## Let us convert travel_time column into Hour and minute format:-
## Creating empty lists to contain hours and minute data
travel_hours =[]
travel_minutes =[]

duration = list(df["travel_time"])

for i in range(len(duration)):
    if len(duration[i].split()) != 2:
        if "h" in duration[i]:
            duration[i] = duration[i].strip()+ " 0m"
        else:
            duration[i] = "0h " +duration[i]

for i in range(len(duration)):
    travel_hours.append(int(duration[i].split(sep="h")[0]))
    travel_minutes.append(int(duration[i].split(sep="m")[0].split()[-1]))

#print(travel_hours)
#print(travel_minutes)
```

```
df['travel_hours'] = travel_hours
df['travel_minutes'] = travel_minutes
```

```
## let's take only city name from source :-
df['source'] = [i.split("-")[1] for i in df['source']]
```

```
## let's take only city name from destination :-

df['destination'] = [i.split("-")[1] for i in df['destination']]
```

```
df.head()
```

]:

	flight_name	flight_id	travel_time	stops	fare	source	destination	dep_hour	dep_minute	arr_hour	arr_minute	travel_hours	travel_minutes
0	Go First	G8 - 713	5h 5m	1 stop at Ahmedabad	6,592	Delhi	Mumbai	19	15	0	20	5	5
1	Go First	G8 - 713	13h 50m	1 stop at Ahmedabad	6,906	Delhi	Mumbai	19	15	9	5	13	50
2	Go First	G8 - 336	2h 5m	Non Stop	7,319	Delhi	Mumbai	15	15	17	20	2	5
3	Go First	G8 - 323	2h 15m	Non Stop	7,319	Delhi	Mumbai	18	20	20	35	2	15
4	Go First	G8 - 330	2h 15m	Non Stop	7,319	Delhi	Mumbai	20	50	23	5	2	15

Dropping some unnecessary columns

```
## Unnamed: 0 column is nothing but an index so dropping this column.
df.drop('Unnamed: 0',axis=1,inplace = True)
```

Checking the statistical summary of the dataset

In descriptive statistics, summary statistics are used to summarize a set of observations, in order to communicate the largest amount of information as simply as possible. Summary statistics summarize and provide information about your sample data.

It tells something about the values in data set. This includes where the average lies and whether the data is skewed. The describe() function computes a summary of statistics pertaining to the Data Frame columns. This function gives the mean, count, max, standard deviation and IQR values of the dataset in a simple understandable way

Statistical Description

```
# Statistical Description
df.describe()
```

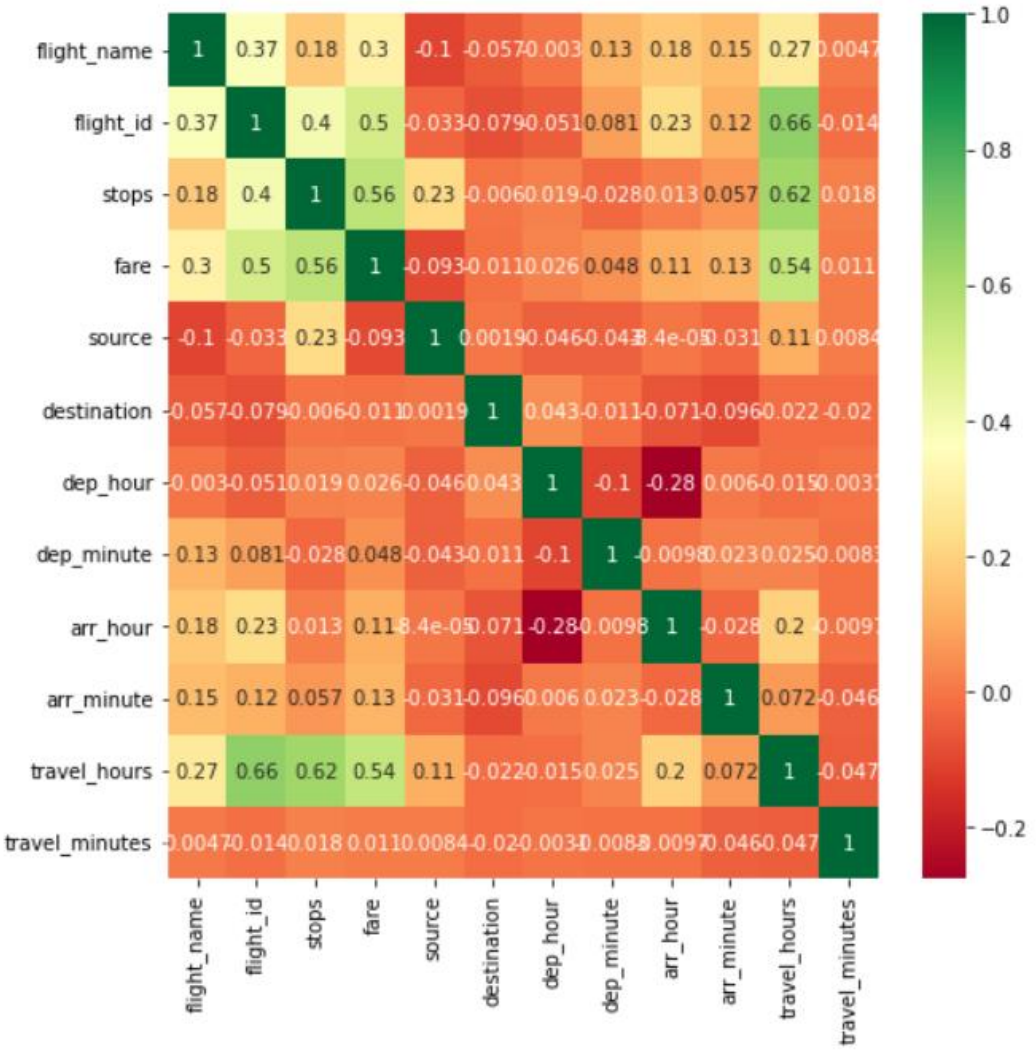
	flight_name	flight_id	stops	fare	source	destination	dep_hour	dep_minute	arr_hour	arr_minute	travel_hours	trave
count	7927.000000	7927.000000	7927.000000	7927.000000	7927.000000	7927.000000	7927.000000	7927.000000	7927.000000	7927.000000	7927.000000	79
mean	3.008831	716.379967	1.251924	17363.793238	12.190110	12.841302	16.838400	27.367857	14.216475	29.330768	14.106219	
std	1.643674	329.640447	0.630608	6795.318861	7.278052	8.007175	3.880501	17.551357	5.746633	17.145254	7.781415	
min	0.000000	0.000000	0.000000	3435.000000	0.000000	0.000000	9.000000	0.000000	0.000000	0.000000	0.000000	
25%	1.000000	529.000000	1.000000	12394.500000	6.000000	6.000000	14.000000	10.000000	9.000000	15.000000	8.000000	
50%	3.000000	702.000000	1.000000	16971.000000	11.000000	12.000000	17.000000	30.000000	14.000000	30.000000	14.000000	
75%	5.000000	1045.000000	2.000000	21615.000000	19.000000	20.000000	20.000000	40.000000	19.000000	45.000000	21.000000	
max	5.000000	1127.000000	4.000000	45633.000000	28.000000	27.000000	23.000000	55.000000	23.000000	55.000000	36.000000	

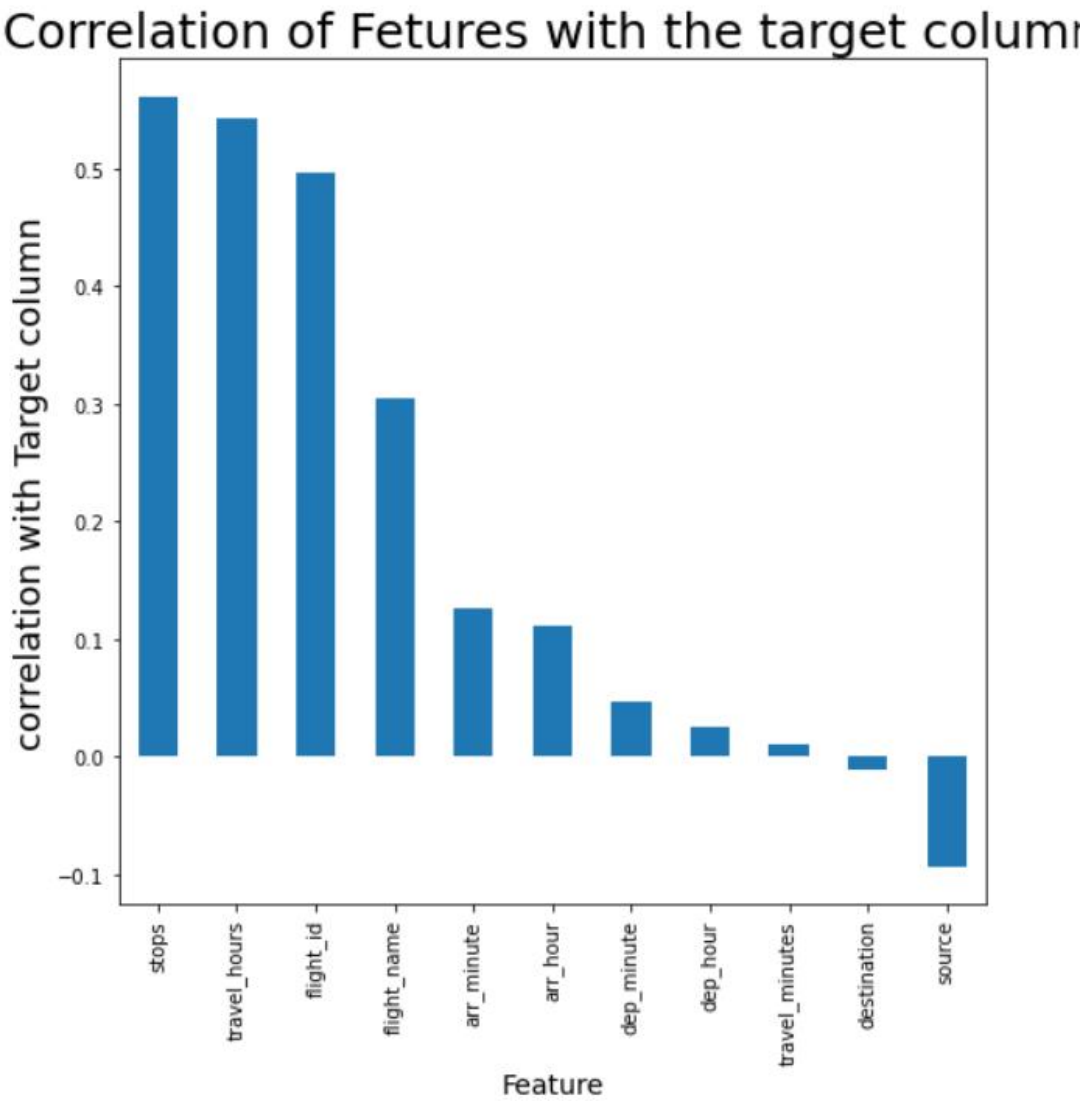
Correlation Factor

The statistical relationship between two variables is referred to as their correlation. The correlation factor represents the relation between columns in a given dataset. A correlation can be positive, meaning both variables are moving in the same direction or it can be negative, meaning that when one variable's value increasing, the other variable's value is decreasing.

Correlation matrix and its visualization

A correlation matrix is a tabular data representing the ‘correlations’ between pairs of variables in a given dataset. It is also a very important pre-processing step in Machine Learning pipelines. The Correlation matrix is a data analysis representation that is used to summarize data to understand the relationship between various different variables of the given dataset.





Observations:

−> Stops and travel_hours has good relationship with fare.

-> dep_hour and dep_minute is negatively correlated with fare.

-> travel_minutes is very much weakly negative correlation with fare.

Label Encoding

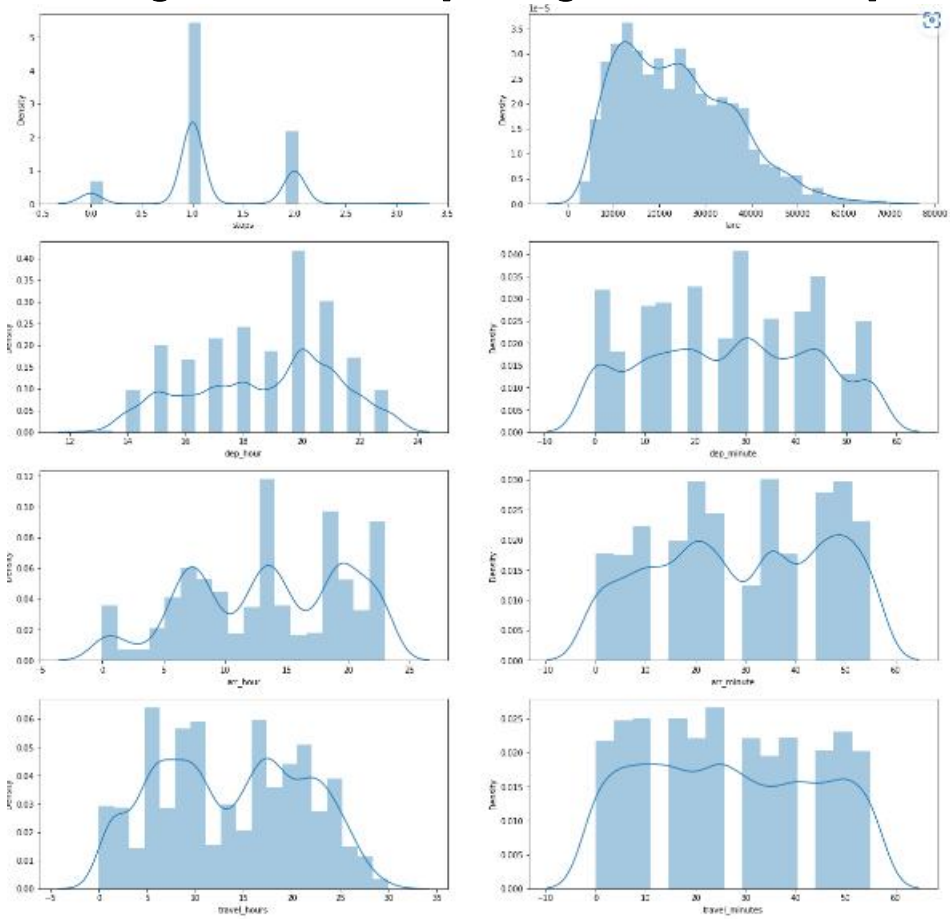
```
from sklearn.preprocessing import LabelEncoder
categorical_cols = [ i for i in df.columns if df[i].dtype==object]
lec = LabelEncoder()
for column in categorical_cols:
    df[column] = lec.fit_transform(df[column])
df
```

3]:

	flight_name	flight_id	stops	fare	source	destination	dep_hour	dep_minute	arr_hour	arr_minute	travel_hours	travel_minutes
0	4	458	1	6592	8	19	19	15	0	20	5	5
1	4	458	1	6906	8	19	19	15	9	5	13	50
2	4	448	0	7319	8	19	15	15	17	20	2	5
3	4	445	0	7319	8	19	18	20	20	35	2	15
4	4	447	0	7319	8	19	20	50	23	5	2	15
...	...	...	...	...	...	...	...	...	...	...	...	...
4317	5	337	1	13713	14	8	20	5	7	35	11	30
4318	5	337	1	13713	14	8	20	5	7	35	11	30
4319	5	64	1	6877	7	8	21	40	6	45	9	5
4320	5	337	1	13713	7	8	20	5	7	35	11	30
4321	5	337	1	13713	7	8	20	5	7	35	11	30

4322 rows × 12 columns

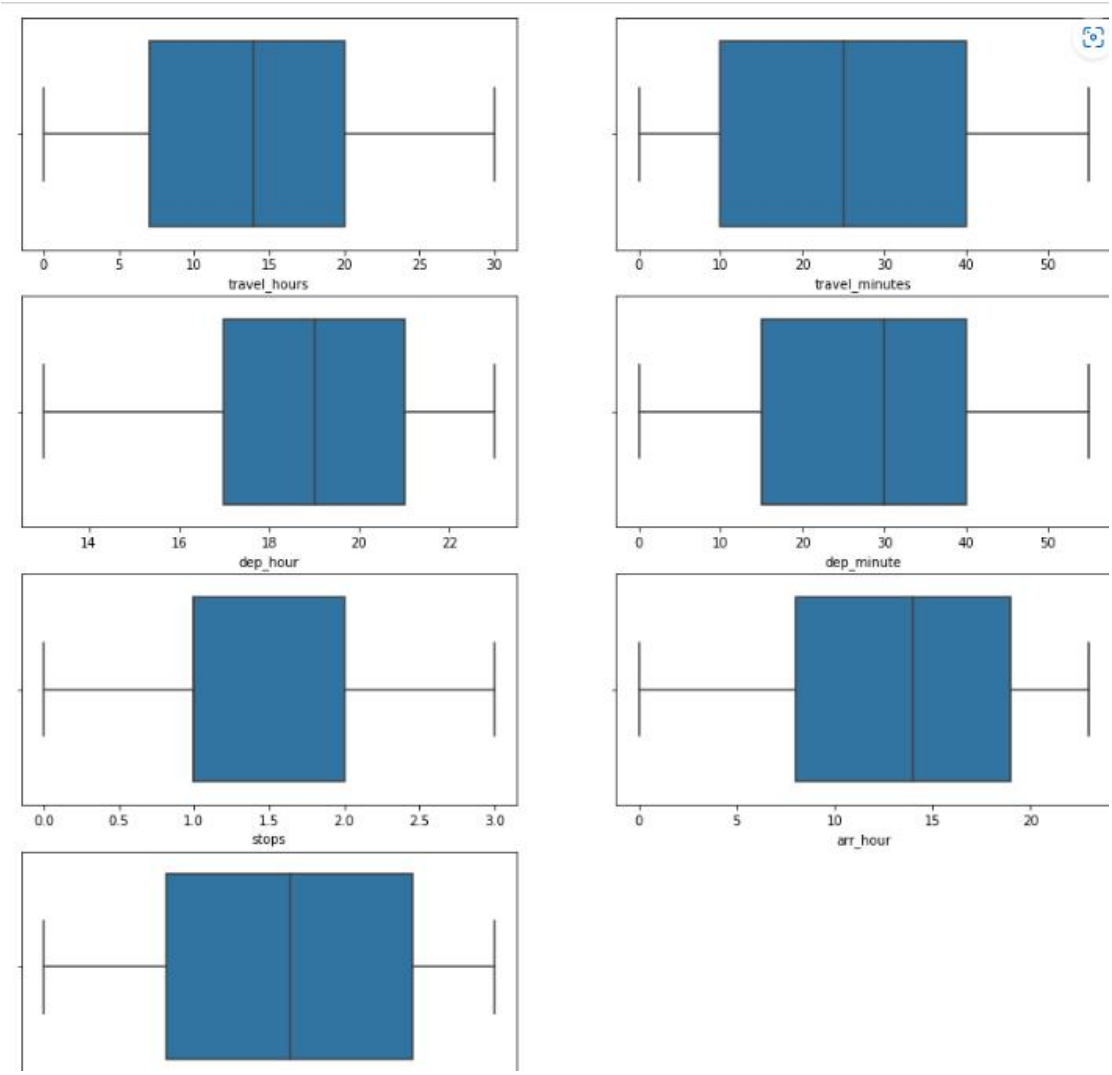
Checking skewness and plotting the distribution plot



Skewness refers to distortion or asymmetry in a symmetrical bell curve, or normal distribution in a set of data. Besides positive and negative skew, distributions can also be said to have zero or undefined skew. The skewness value can be positive, zero, negative, or undefined.

Checking outliers and plotting it

An outlier is a data point in a data set which is distant or far from all other observations available. It is a data point which lies outside the overall distribution which is available in the dataset. In statistics, an outlier is an observation point that is distant from other observations. A box plot is a method or a process for graphically representing groups of numerical data through their quartiles. Outliers may also be plotted as an individual point. If there is an outlier it will plotted as point in box plot but other numerical data will be grouped together and displayed as boxes in the diagram. In most cases a threshold of 3 or - 3 is used i.e., if the Z-score value is higher than or less than 3 or -3 respectively, that particular data point will be identified as outlier.



Scaling the data using Standard Scaler

For each value in a feature, StandardScaler subtracts the minimum value in the feature and then divides by the range. The range is the difference between the original maximum and original minimum. StandardScaler preserves the shape of the original distribution.

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
x =pd.DataFrame(scaler.fit_transform(x),columns=x.columns)
x
```

	flight_name	flight_id	stops	source	destination	dep_hour	dep_minute	arr_hour	arr_minute	travel_hours	travel_minutes
0	-0.628263	0.209663	-0.325499	-0.543377	0.671102	0.107193	-0.710872	-2.130998	-0.541002	-1.158758	-1.268304
1	-0.628263	0.209663	-0.325499	-0.543377	0.671102	0.107193	-0.710872	-0.707185	-1.422015	-0.091974	1.364976
2	-0.628263	0.157823	-2.088417	-0.543377	0.671102	-1.501452	-0.710872	0.558427	-0.541002	-1.558802	-1.268304
3	-0.628263	0.142271	-2.088417	-0.543377	0.671102	-0.294968	-0.410100	1.033031	0.340012	-1.558802	-0.683131
4	-0.628263	0.152639	-2.088417	-0.543377	0.671102	0.509355	1.394534	1.507636	-1.422015	-1.558802	-0.683131
...	...	...	...	...	...	...	...	...	...	...	...
4317	-0.152803	-0.417602	-0.325499	0.400797	-0.633508	0.509355	-1.312416	-1.023588	0.340012	-0.358670	0.194629
4318	-0.152803	-0.417602	-0.325499	0.400797	-0.633508	0.509355	-1.312416	-1.023588	0.340012	-0.358670	0.194629
4319	-0.152803	-1.832835	-0.325499	-0.700740	-0.633508	0.911516	0.792989	-1.181789	0.927354	-0.625366	-1.268304
4320	-0.152803	-0.417602	-0.325499	-0.700740	-0.633508	0.509355	-1.312416	-1.023588	0.340012	-0.358670	0.194629
4321	-0.152803	-0.417602	-0.325499	-0.700740	-0.633508	0.509355	-1.312416	-1.023588	0.340012	-0.358670	0.194629

4322 rows × 11 columns

Run and evaluate selected models

We will find the best random state value so that we can create our train_test_split.

Finding the best random state

```
max_r2_score=0
for r_state in range(1,200):
    x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=r_state,test_size=0.30)
    rf=RandomForestRegressor()
    rf.fit(x_train,y_train)
    y_pred=rf.predict(x_test)
    score=r2_score(y_test,y_pred)
    if score > max_r2_score:
        max_r2_score = score
        final_r_state = r_state
print("max r2 score corresponding to",final_r_state,"is",max_r2_score)
```

```
max r2 score corresponding to 56 is 0.8474228131709626
```

Now, we will run a for loop for all regression algorithms and find the best model

```
lr=LinearRegression()
lasso=Lasso()
ridge=Ridge()
svr=SVR()
dtr=DecisionTreeRegressor()
knr=KNeighborsRegressor()
rf = RandomForestRegressor()
adb = AdaBoostRegressor()
gbr =GradientBoostingRegressor()
```

```
models= []
models.append(('Linear Regression',lr))
models.append(('Lasso Regression',lasso))
models.append(('Random Forest Regression',rf))
models.append(('Ridge Regression',ridge))
models.append(('Support Vector Regressor',svr))
models.append(('Decision Tree Regressor',dtr))
models.append(('KNeighbors Regressor',knr))
models.append(('AdaBoost Regressor',adb))
models.append(('GradientBoostingRegressor',gbr))
```

Random Forest

```
from sklearn.model_selection import GridSearchCV

params = {'n_estimators': [100,200],
          'criterion' : ["mse","mae"],
          'max_depth' : range(2,6),
          'min_samples_split' : range(2,4),
          'min_samples_leaf':range(2,7)



        }
rfr = RandomForestRegressor()
clf  = GridSearchCV(rfr,params,cv=5,scoring ='r2')
clf.fit(x_train,y_train)
print(clf.best_params_)  #Printing the best parameters obtained
print(clf.best_score_) #Mean cross-validated score of best_estimator
```

```
{'criterion': 'mse', 'max_depth': 5, 'min_samples_leaf': 3, 'min_samples_split': 2, 'n_estimators': 100}
0.6406801715974881
```

```
## Providing best parameters to the model
rfr=RandomForestRegressor(criterion='mse', max_depth = 5,min_samples_leaf=3, n_estimators=100)
rfr.fit(x_train,y_train)
pred=rfr.predict(x_test)
print(' r2_score after tuning is: ',r2_score(y_test,pred)*100)
print('Cross validation score: ',cross_val_score(rfr,x,y,cv=5,scoring='r2').mean()*100)
print('Standard deviation: ',cross_val_score(rfr,x,y,cv=5,scoring='r2').std())
print('\n')
print('Mean absolute error: ',mean_absolute_error(y_test,pred))
print('Mean squared error: ',mean_squared_error(y_test,pred))
print('Root Mean squared error: ',np.sqrt(mean_squared_error(y_test,pred)))
```

```
 r2_score after tuning is:  66.24201292462574
Cross validation score:  29.562623874572168
Standard deviation:  0.30158492060793207


Mean absolute error:  5653.461335881796
Mean squared error:  53220552.96550928
Root Mean squared error:  7295.241803087083
```

CONCLUSION Key Findings and Conclusions of the Study

-> After getting an insight of this dataset, we were able to understand that the Flight ticket prices are done on basis of different features.

-> First, I loaded the dataset and did the EDA process and other preprocessing techniques like skewness check and removal, checking the outliers present, format the data, visualizing the distribution of data, etc.

-> Then I did the model training, building the model and finding out the best model on the basis of different metrices scores I got like Mean Absolute Error, Mean squared Error, Root Mean Squared Error, etc.

-> I Random Forest Regressor as the best algorithm among all as it gave more r2_score and cross_val_score. Then for finding out the best parameter and improving the scores, we performed Hyperparameter Tuning.

-> I saved the model in a pickle with a filename in order to use whenever we require.

-> I predicted the values obtained and saved it.

-> From this project, I learnt scrapping data from travel website, This will be useful while we are working in a real-time case study as we can get any new data from the client we work on and we can proceed our analysis by loading the best model we obtained and start working on the analysis of the new data we have.

-> Overall, we can say that this dataset is good for predicting the flight ticket prices using regression analysis and RandomForestRegressor is the best working algorithm model we obtained.

-> I can improve the data by adding more features that are positively correlated with the target variable, having less outliers, normally distributed values, etc.

1. Most No of Flights are available from Mumbai.

2. Among Vistara, Air India, Indigo, Go First, Air Asia, SpiceJet flights, Air India flights are most expensive followed by Vistara flights

.

3.Non -Stop Flights are cheaper than flights including stops.

4. As Travel Hour increases, Fare Increases.

5.Vistara flights availability is more than any other airlines for different places.

6.Evening flights availability is more than morning flights. Early morning and late nights flights are cheaper.

Learning Outcomes of the Study in respect of Data Science.

1. **Price Prediction modeling** – This allows predicting the prices of flight tickets & how they are varying in nature considering the different factors affecting the prices in the real time scenarios.

2. **Deployment of ML models** – The Machine learning models can also predict the houses depending upon the needs of the buyers and recommend them, so customers can make final decisions as per the need

3. Ways to select features and to do hyperparameter tuning efficiently.