FLIP ROBO

FLIGHT TICKET PRICE PREDICTION PROJECT

Submitted by:

Swati Pandey

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 Swati MahaSeth 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 co-operation 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 the cost of a 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 at the 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). It is 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 to be 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

1.4 Motivation for the Problem Undertaken

The project was the first provided to me by Flip Robo Technologies as a part of the internship programme. The exposure to real world data and the opportunity to deploy my skillset in solving a real time problem has been the primary motivation.

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. In addition, competition between airlines makes the task of determining optimal pricing is hard for everyone. *So prime motive is to build flight price predication system based on short range timeframe (7- 14 days) data available prior to actual take-off date.*

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.

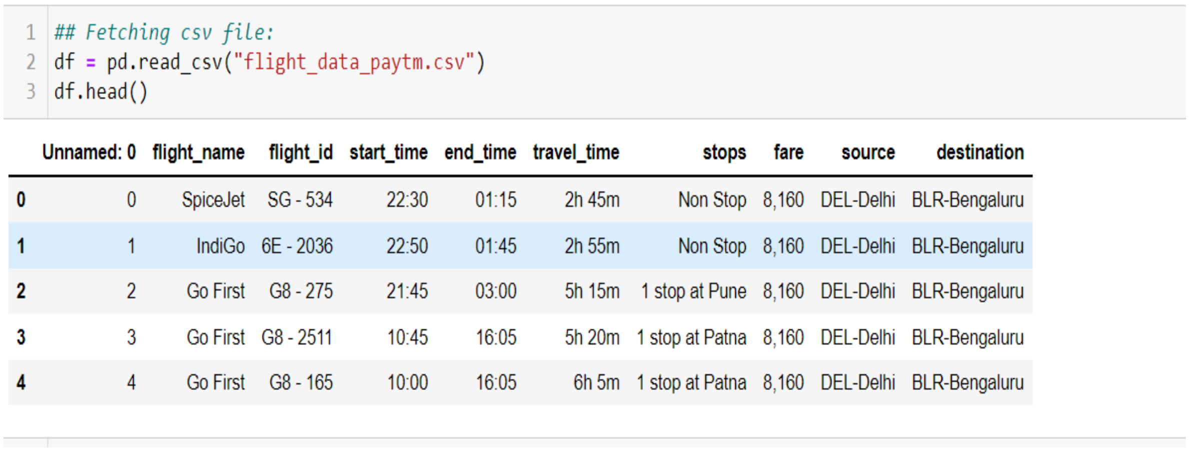
9. **Support Vector Regressor –** SVR is a supervised learning algorithm, That is used to predict discrete values. SVR uses the same principle As the SVMs. The basic idea behind SVR is to fit best line. In SVR the Best fit line is the hyperplane that has the maximum number of points.

-> 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 belowto 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

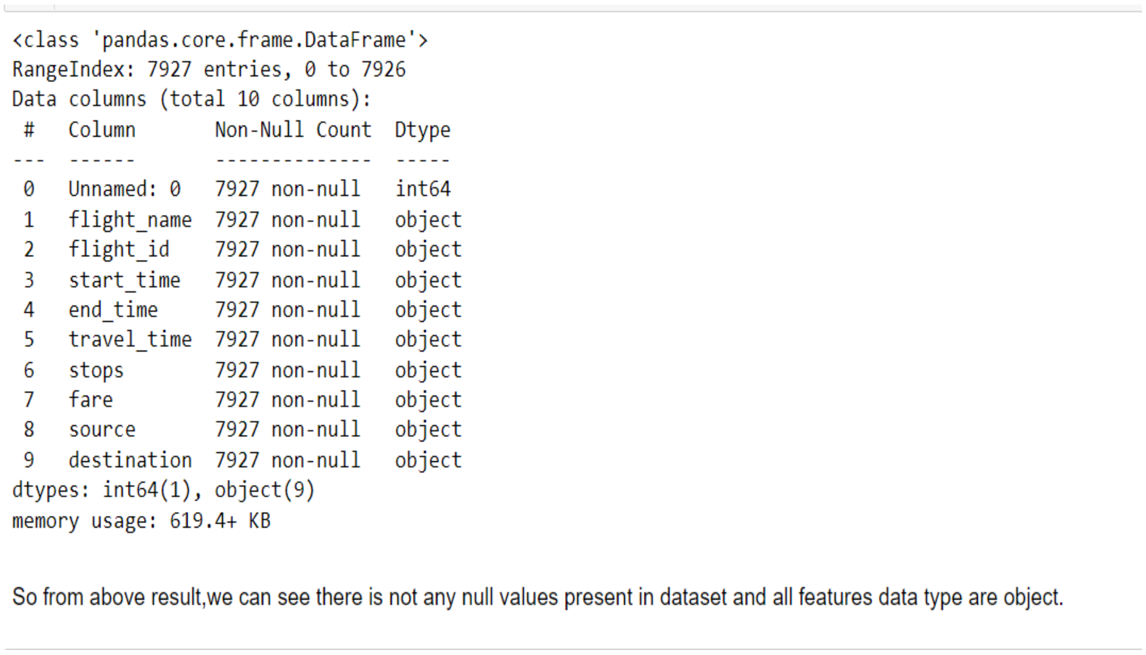
Data description

Data contains 7927 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. Start_time :- Flight's departure time from source.

4. End_time:- Flight's arrival time for destination.
5. travel_time:- Time taken by flight to reach source from destination.
6. Stops: - route via flight goes from source to destination, 0 means nonstop, greater than 0 means taking stops.
7. fare: - flight's ticket price
8. source: - From where flight takes off.
9. Destination: - Place, where flight lands.

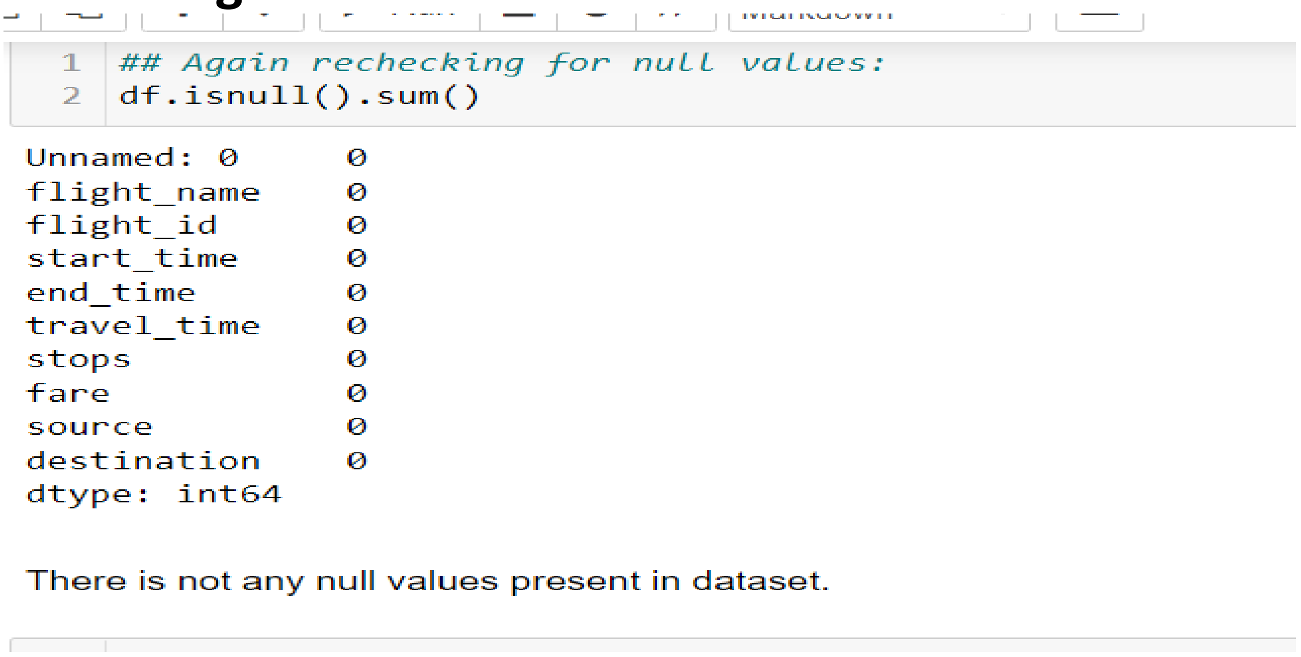
Checking the data type & info of dataset



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7927 entries, 0 to 7926
Data columns (total 10 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   Unnamed: 0   7927 non-null   int64 
 1   flight_name  7927 non-null   object
 2   flight_id    7927 non-null   object
 3   start_time   7927 non-null   object
 4   end_time     7927 non-null   object
 5   travel_time  7927 non-null   object
 6   stops        7927 non-null   object
 7   fare         7927 non-null   object
 8   source       7927 non-null   object
 9   destination  7927 non-null   object
dtypes: int64(1), object(9)
memory usage: 619.4+ KB
```

So from above result,we can see there is not any null values present in dataset and all features data type are object.

Checking the no. of null values in the dataset



```
## Again rechecking for null values:
df.isnull().sum()
```

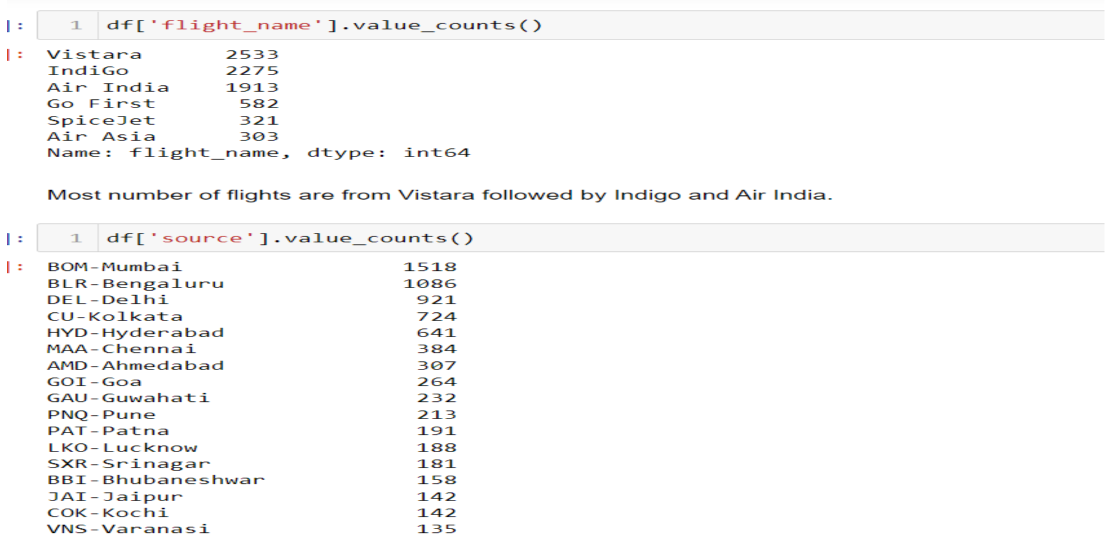
```
Unnamed: 0      0
flight_name     0
flight_id       0
start_time      0
end_time        0
travel_time     0
stops           0
fare            0
source          0
destination     0
dtype: int64
```

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       2533
IndiGo        2275
Air India     1913
Go First       582
SpiceJet       321
Air Asia       303
Name: flight_name, dtype: int64
```

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

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

```
BOM-Mumbai                1518
BLR-Bengaluru             1086
DEL-Delhi                  921
CU-Kolkata                 724
HYD-Hyderabad              641
MAA-Chennai                384
AMD-Ahmedabad              307
GOI-Goa                    264
GAU-Guwahati               232
PNQ-Pune                   213
PAT-Patna                  191
LKO-Lucknow                188
SXR-Srinagar               181
BBI-Bhubaneshwar           158
JAI-Jaipur                 142
COK-Kochi                  142
VNS-Varanasi               135
```

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']]
```

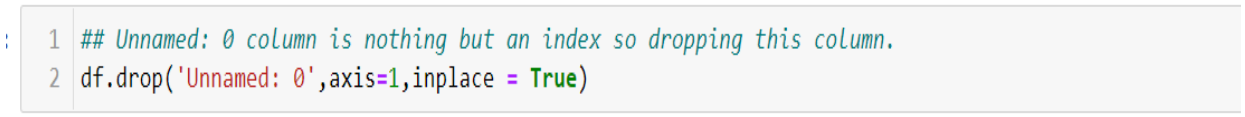
```
## let's take only city name from destination :-

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	SpiceJet	SG - 534	2h 45m	Non Stop	8,160	Delhi	Bengaluru	22	30	1	15	2	45
1	IndiGo	6E - 2036	2h 55m	Non Stop	8,160	Delhi	Bengaluru	22	50	1	45	2	55
2	Go First	G8 - 275	5h 15m	1 stop at Pune	8,160	Delhi	Bengaluru	21	45	3	0	5	15
3	Go First	G8 - 2511	5h 20m	1 stop at Patna	8,160	Delhi	Bengaluru	10	45	16	5	5	20
4	Go First	G8 - 165	6h 5m	1 stop at Patna	8,160	Delhi	Bengaluru	10	0	16	5	6	5

```
## Removing comma drom fare:
df['fare'] = [i.replace(",","") for i in df['fare']]
```

Dropping some unnecessary columns

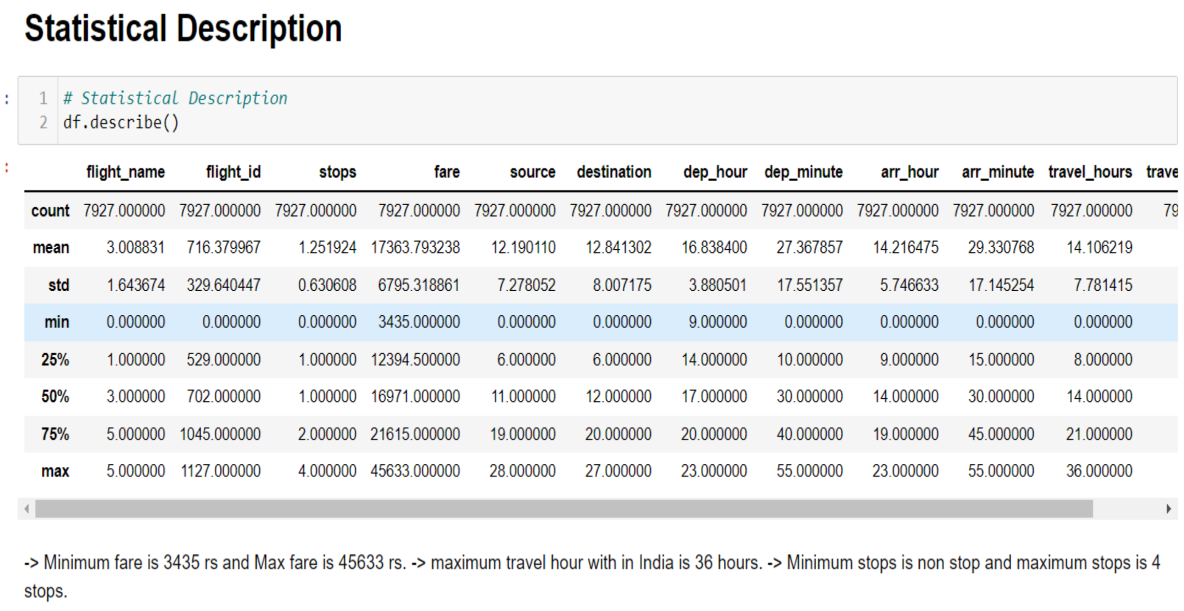


```
## Unnamed: 0 column is nothing but an index so dropping this column.
df.drop('Unnamed: 0',axis=1,inplace = True)
```

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	

-> Minimum fare is 3435 rs and Max fare is 45633 rs. -> maximum travel hour with in India is 36 hours. -> Minimum stops is non stop and maximum stops is 4 stops.

Observations

-> Minimum fare is 3435 rs and Max fare is 45633 rs.

-> maximum travel hour with in India is 36 hours.

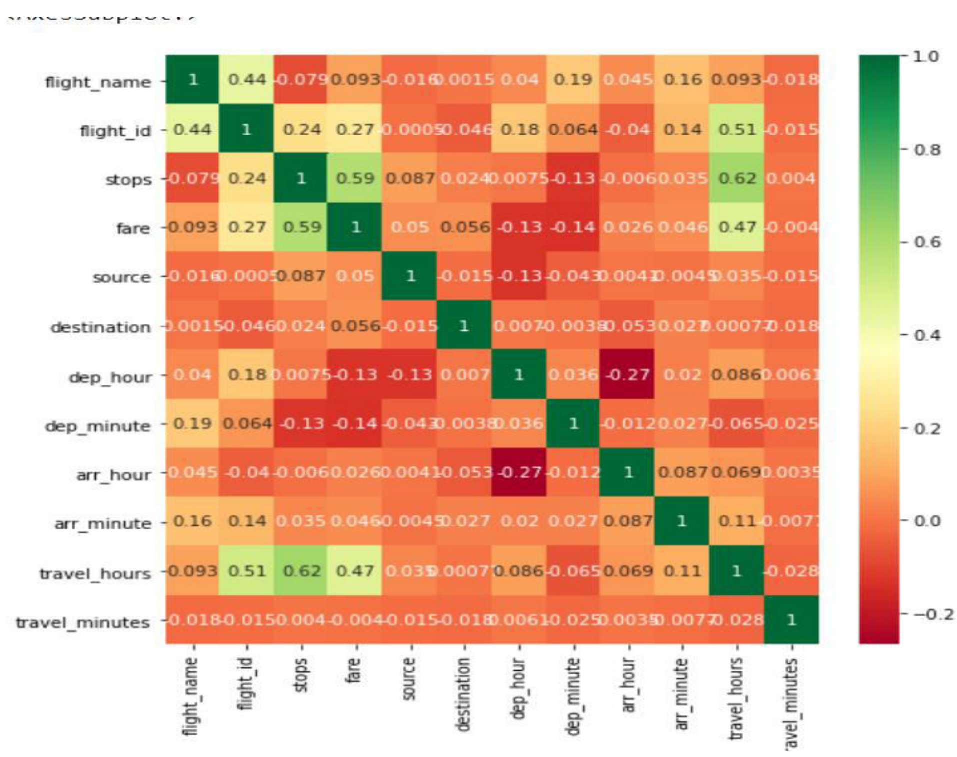
-> Minimum stops is nonstop and maximum stops is 4 stops.

Correlation Factor

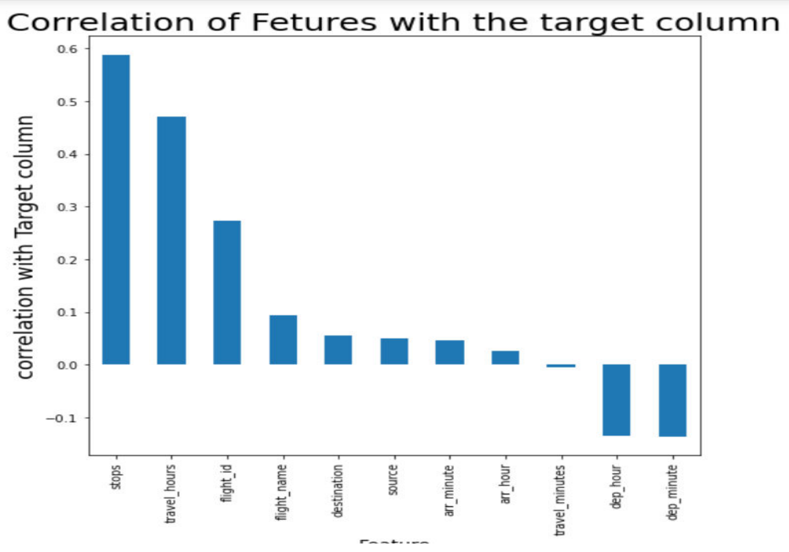
The statistical relationship between two variables is referred to as their correlation. The correlation factor represents the relation between columns in a given dataset. A correlation can be positive, meaning both variables are moving in the same direction or it can be negative, meaning that when one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



Correlation with target variable



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.

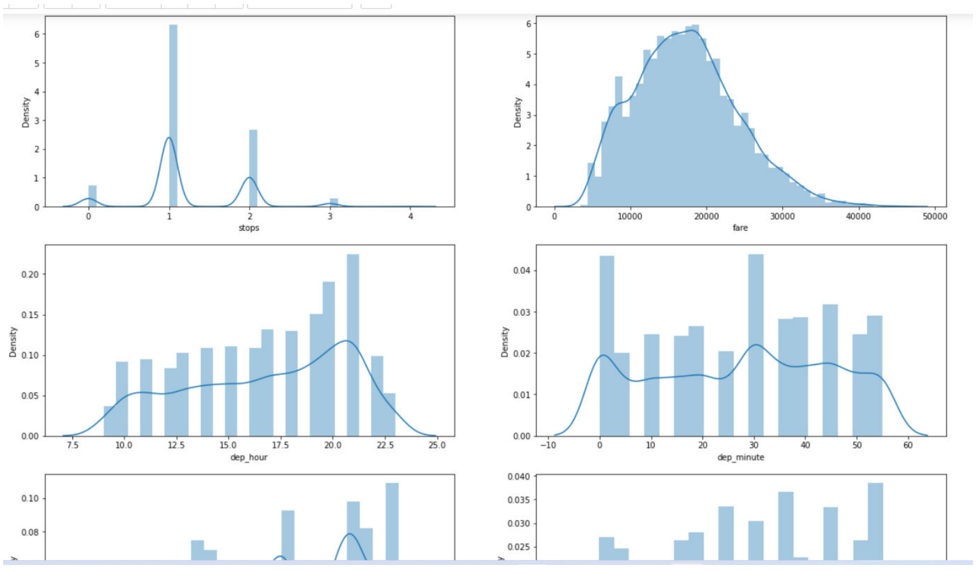
Label Encoding

```python
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
```

	flight_name	flight_id	stops	fare	source	destination	dep_hour	dep_minute	arr_hour	arr_minute	travel_hours	travel_minutes
0	4	968	0	8160	8	3	22	30	1	15	2	45
1	3	52	0	8160	8	3	22	50	1	45	2	55
2	2	782	1	8160	8	3	21	45	3	0	5	15
3	2	771	1	8160	8	3	10	45	16	5	5	20
4	2	739	1	8160	8	3	10	0	16	5	6	5
...	...	...	...	...	...	...	...	...	...	...	...	...
7922	1	641	2	19497	7	8	15	5	13	45	22	40
7923	1	659	2	24484	7	8	14	0	7	20	17	20
7924	1	659	2	24484	7	8	14	0	7	20	17	20
7925	1	641	2	24642	7	8	15	5	13	45	22	40
7926	1	659	2	28317	7	8	14	0	13	45	23	45

7927 rows × 12 columns

Checking skewness and plotting the distribution plot

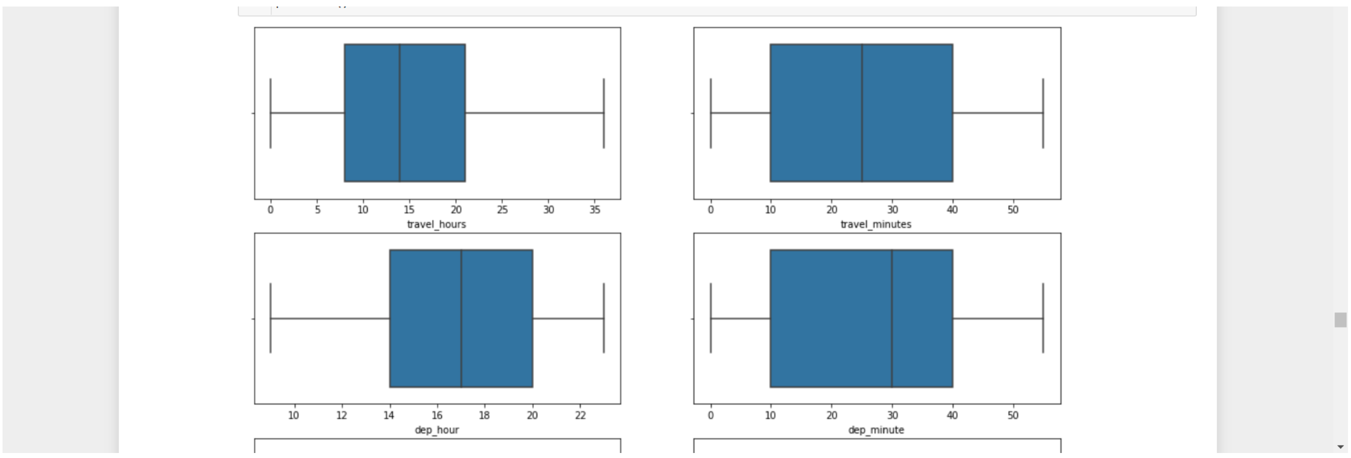


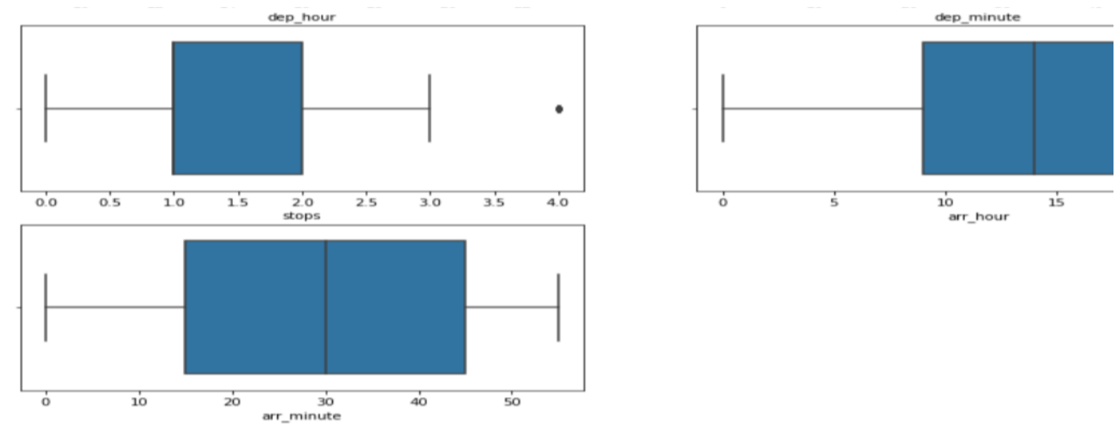
Skewness refers to distortion or asymmetry in a symmetrical bell curve, or normal distribution in a set of data. Besides positive and negative skew, distributions can also be said to have zero or undefined skew. The skewness value can be positive, zero, negative, or undefined.

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.





There is not any outliers present in our dataset.

Hardware and Software Requirements and Tools Used

For doing this project, the hardware used is a laptop with high end specification and a stable internet connection. While coming to software part, I had used anaconda navigator and in that I have used **Jupyter notebook** to do my python programming and analysis.

For using a csv file, Microsoft excel is needed. In Jupyter notebook, I had used lots of python libraries to carry out this project and I have mentioned below with proper justification:

1. Pandas- a library which is used to read the data, visualization andanalysis of data.
2. NumPy- used for working with array and various mathematicaltechniques.
3. Seaborn- visualization tool for plotting different types of plot.
4. Matplotlib- It provides an object-oriented API for embedding plotsinto applications.
5. zscore- technique to remove outliers.
6. skew ()- to treat skewed data .
7. VIF- I used this to check multicollinearity in data.
8. standard scaler- I used this to scale my data before sending it tomodel.

9. train_test_split- to split the test and train data.
10. Then I used different regression algorithms to find out the best model for predictions.
11. pickle- library used to save the model in either pickle or obj file.

MODEL/S DEVELOPMENT AND EVALUATION

Identification of possible problem-solving approaches (methods)

From the given dataset it can be concluded that it is a Regression problem as the output column “SalePrice” has continuous output. So, for further analysis of the problem, we have to import or call out the Regression related libraries in Python work frame.

The different libraries used for the problem solving are:
sklearn - Scikit-learn is a free machine learning library for Python. It features various algorithms like support vector machine, random forests, and k-neighbours, and it also supports Python numerical and scientific libraries like NumPy and SciPy.

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

In statistics, linear regression is a linear approach to modelling the relationship between a scalar response and one or more explanatory variables. The case of one explanatory variable is called simple linear regression; for more than one, the process is called multiple linear regressions.

ii. 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.

Lasso regression is a type of linear regression that uses shrinkage. Shrinkage is where data values are shrunk towards a central point, like the mean. The lasso procedure encourages simple, sparse models (i.e., models with fewer parameters). This particular type of regression is well-suited for models showing high levels of multicollinearity or when you want to automate certain parts of model selection, like variable selection/parameter elimination.

iii. Ridge - The 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 multicollinearity (correlations between predictor variables). Ridge regression is particularly useful to mitigate the problem of multicollinearity in linear regression, which commonly occurs in models with large numbers of parameters. In general, the method provides improved efficiency in parameter estimation problems in exchange for a tolerable amount of bias.

iv. SupportVectorRegressor:

To eliminate the limitations found in lasso, the elastic net includes a quadratic expression (||β||2) in the penalty, which, when used in isolation, becomes ridge regression. The quadratic expression in the penalty elevates the loss function toward being convex. The elastic net draws on best of both i.e., lasso and ridge regression. In the procedure for finding the elastic net method's estimator, there are two stages that involve both the lasso and regression techniques. It first finds the ridge regression coefficients and then conducts the second step by using a lasso sort of shrinkage of the coefficients.

2. sklearn.tree –
Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

There are several advantages of using decision trees for predictive analysis:

- Decision trees can be used to predict both continuous and discrete values i.e., they work well for both regression and classification tasks.
- They require relatively less effort for training the algorithm.
- They can be used to classify non-linearly separable data.
- They're very fast and efficient compared to KNN and other algorithms.

Decision tree learning is one of the predictive modelling approaches used in statistics, data mining and machine learning. It uses a decision tree to go from observations about an item to conclusions about the item's target value.

Decision Tree Regressor - Decision Tree is one of the most commonlyused, 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.

Decision tree builds regression or classification models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes. A decision node (e.g., Outlook) has two or more branches (e.g., Sunny, Overcast and Rainy), each representing values for the attribute tested. Leaf node (e.g., Hours Played) represents a decision on the numerical target. The topmost decision node in a tree which corresponds to the best predictor called root node. Decision trees can handle both categorical and numerical data.

3. sklearn.ensemble

The goal of ensemble methods is to combine the predictions of several base estimators built with a given learning algorithm in order to improve generalizability / robustness over a single estimator. The sklearn.ensemble module includes two averaging algorithms based on randomized decision trees: the RandomForest algorithm and the Extra-Trees method. Both algorithms are perturb-and-combine techniques specifically designed for trees. This means a diverse set of classifiers is created by introducing randomness in the classifier construction. The prediction of the ensemble is given as the averagedprediction of the individual classifiers.

Boosting ensemble algorithms creates a sequence of models that attempt to correct the mistakes of the models before them in the sequence. Once created, the models make predictions which may be weighted by their demonstrated accuracy and the results are combined to create a final output prediction.

The different types of ensemble techniques used in the model are:

i. Random Forest Regressor - It 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. Random forest is a type of supervised learning algorithm that uses ensemble methods (bagging) to solve both regression and classification problems. The algorithm operates by constructing a multitude of decision trees at training time and outputting the mean/mode of prediction of the individual trees.

ii. AdaBoost Regressor - It 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.

iii. 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.

4. sklearn.metrics - The sklearn. metrics module implements several losses, score, and utility functions to measure classification performance. Some metrics might require probability estimates of the positive class, confidence values, or binary decisions values.

Important sklearn.metrics modules used in the project are:

i. mean_absolute_error - In statistics, mean absolute error is a measure of errors between paired observations expressing the same phenomenon. Examples of Y versus X include comparisons of predicted versus observed, subsequent time versus initial time, and one technique of measurement versus an alternative technique of measurement.

The MAE measures the average magnitude of the errors in a set of forecasts, without considering their direction. It measures accuracy for continuous variables. Mean Absolute Error (MAE): MAE measures the average magnitude of the errors in a set of predictions, without considering their direction. It's the average over the test sample of the absolute differences between prediction and actual observation where all individual differences have equal weight.

ii. mean_squared_error - In statistics, the mean squared error or mean squared deviation of an estimator measures the average of the squares of the errors—that is, the average squared difference between the estimated values and the actual value. MSE is a risk function, corresponding to the expected value of the squared error loss. Mean Square Error (MSE) is defined as Mean or Average of the square of the difference between actual and estimated values.

iii. r2_score - In statistics, the coefficient of determination, denoted R^2 or r^2 and pronounced "R squared", is the proportion of the variance in the dependent variable that is predictable from the independent variable. R-squared is a statistical measure of how close the data are to the fitted regression line. It is also known as the coefficient of determination, or the coefficient of multiple determinations for multiple regressions.

5. sklearn.model_selection –

i. GridSearchCV - It is a library function that is a member of sklearn’s model_selection package. It helps to loop through predefined hyper parameters and fit your estimator (model) on your training set. So, in the end, you can select the best parameters from the listed hyperparameters. GridSearchCV combines an estimator with a grid search preamble to tune hyper-parameters. The method picks the optimal parameter from the grid search and uses it with the estimator selected by the user.

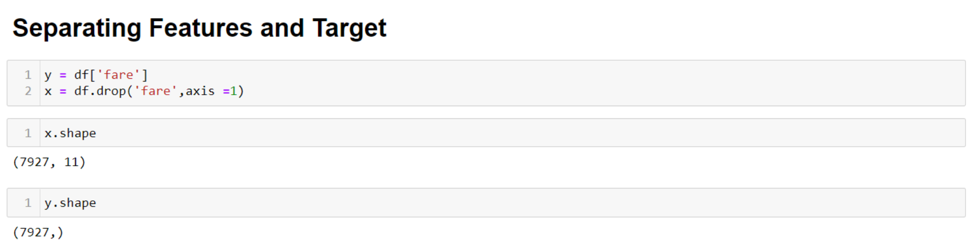
ii. cross_val_score - Cross validation helps to find out the over fitting and under fitting of the model. In the cross validation the model is made to run on different subsets of the dataset which will get multiple measures of the model. If we take 5 folds, the data will be divided into 5 pieces where each part being 20% of 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 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.

cross_val_score estimates the expected accuracy of the model on out- of-training data (pulled from the same underlying process as the training data). The benefit is that one need not set aside any data to obtain this metric, and we can still train the model on all of the available data.

Testing of Identified Approaches

After completing the required pre-processing techniques for the model building data is separated as input and output columns before passing it to the train_test_split.



Separating Features and Target

```
y = df['fare']
x = df.drop('fare',axis =1)
```

```
x.shape
```

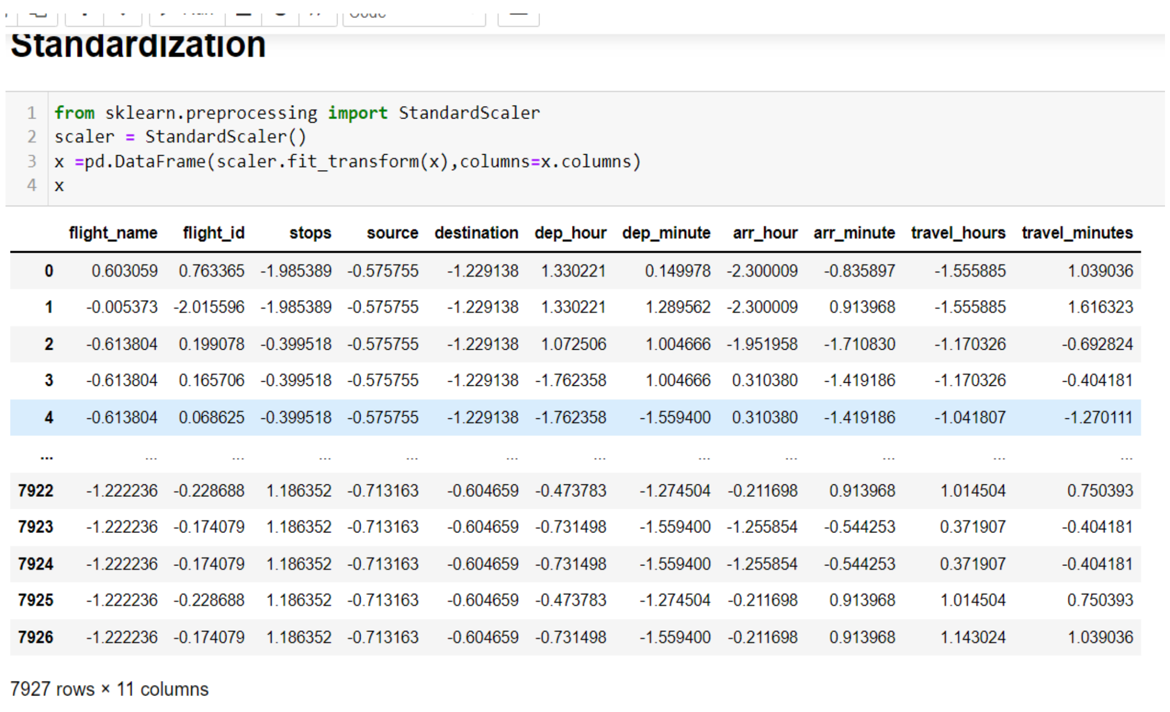
```
(7927, 11)
```

```
y.shape
```

```
(7927,)
```

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.



Standardization

```
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	0.603059	0.763365	-1.985389	-0.575755	-1.229138	1.330221	0.149978	-2.300009	-0.835897	-1.555885	1.039036
1	-0.005373	-2.015596	-1.985389	-0.575755	-1.229138	1.330221	1.289562	-2.300009	0.913968	-1.555885	1.616323
2	-0.613804	0.199078	-0.399518	-0.575755	-1.229138	1.072506	1.004666	-1.951958	-1.710830	-1.170326	-0.692824
3	-0.613804	0.165706	-0.399518	-0.575755	-1.229138	-1.762358	1.004666	0.310380	-1.419186	-1.170326	-0.404181
4	-0.613804	0.068625	-0.399518	-0.575755	-1.229138	-1.762358	-1.559400	0.310380	-1.419186	-1.041807	-1.270111
...	...	...	...	...	...	...	...	...	...	...	...
7922	-1.222236	-0.228688	1.186352	-0.713163	-0.604659	-0.473783	-1.274504	-0.211698	0.913968	1.014504	0.750393
7923	-1.222236	-0.174079	1.186352	-0.713163	-0.604659	-0.731498	-1.559400	-1.255854	-0.544253	0.371907	-0.404181
7924	-1.222236	-0.174079	1.186352	-0.713163	-0.604659	-0.731498	-1.559400	-1.255854	-0.544253	0.371907	-0.404181
7925	-1.222236	-0.228688	1.186352	-0.713163	-0.604659	-0.473783	-1.274504	-0.211698	0.913968	1.014504	0.750393
7926	-1.222236	-0.174079	1.186352	-0.713163	-0.604659	-0.731498	-1.559400	-0.211698	0.913968	1.143024	1.039036

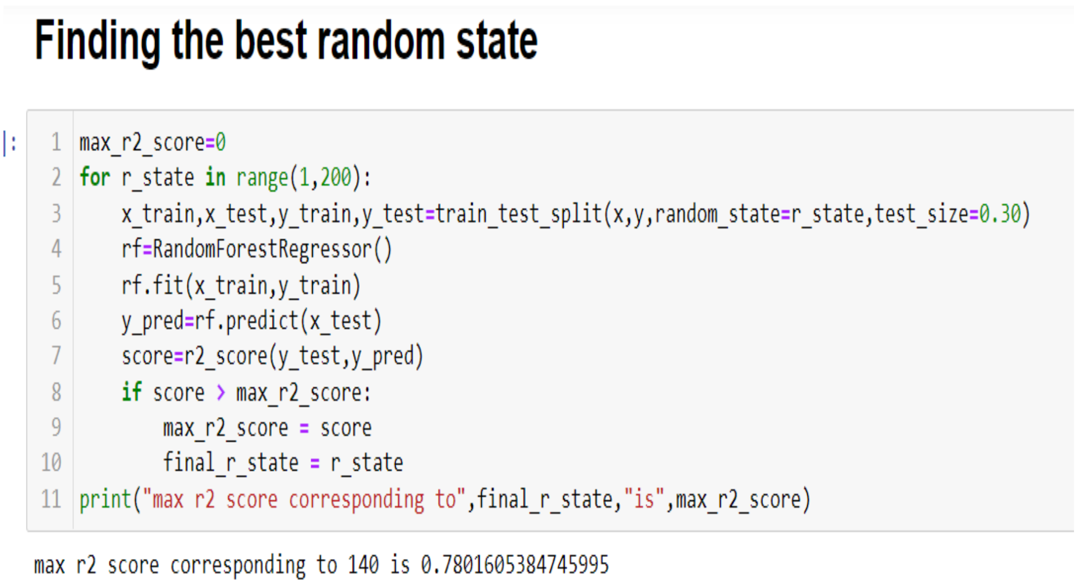
7927 rows × 11 columns

Using VIF

An important machine learning method for checking multi collinearity in data. VIF score of an independent variable represents how well the variable is explained by other independent variables.

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 140 is 0.7801605384745995
```

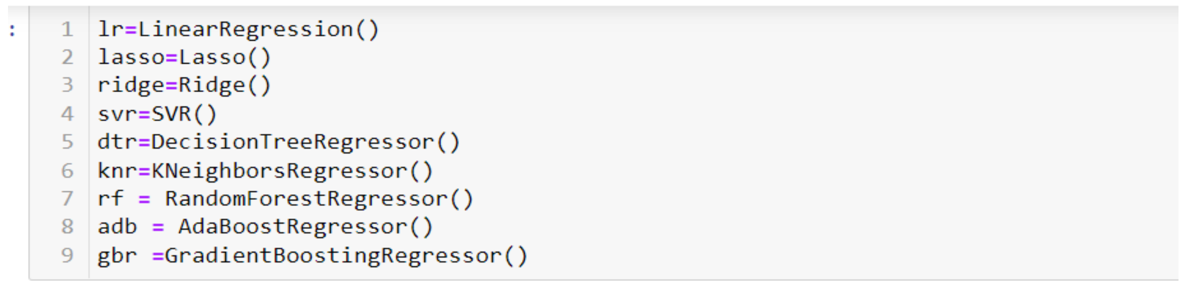
Train Test Split

Scikit-learn is a Python library that offers various features for data processing that can be used for classification, clustering, and model selection. Model_selection is a method for setting a blueprint to analyze data and then using it to measure new data. Selecting a proper model allows you to generate accurate results when making a prediction. If we have one dataset, then it needs to be split by using the Sklearn train_test_split function first. By default, Sklearn train_test_split will make random partitions for the two subsets.

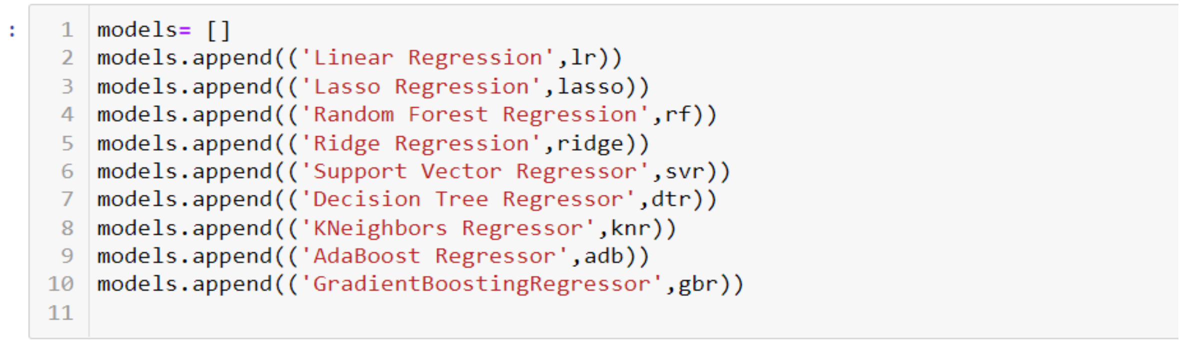
The train_test_split is a function in Sklearn model selection for splitting data arrays into two subsets: for training data and for testing data. With this function, we don't need to divide the dataset manually. The train_test_split function is for splitting a single dataset for two different purposes: training and testing. The testing subset is for building your model. The testing subset is for using the model on unknown data to evaluate the performance of the model.

```
#Creating train_test_split using best random_state

x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=r_state,te
st_size=.30)
```

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()
```

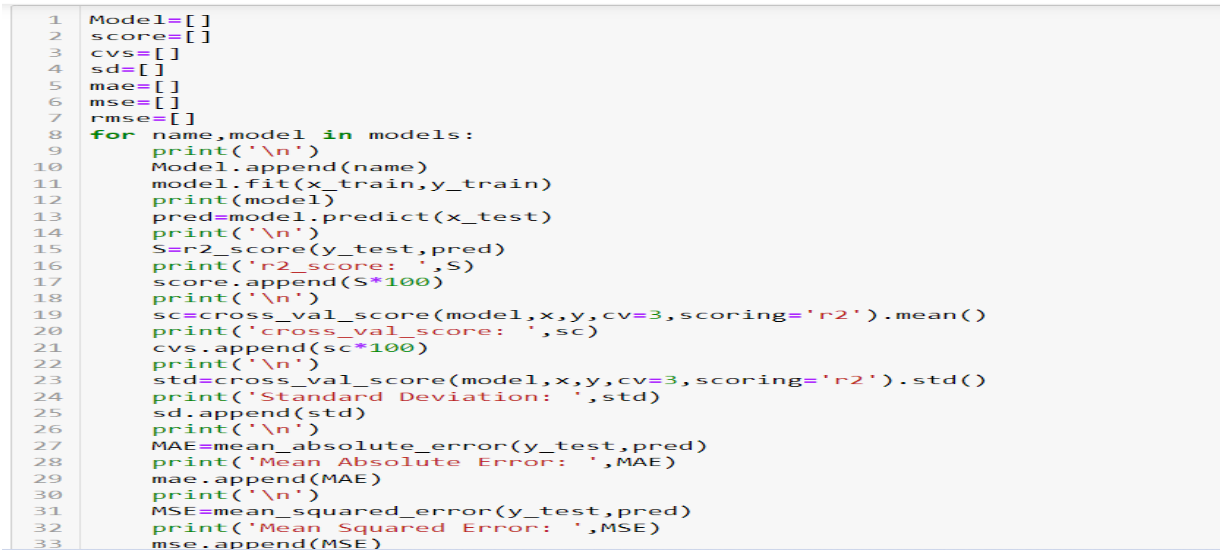


```
models= []
models.append(('Linear Regression',lr))
models.append(('Lasso Regression',lasso))
models.append(('Random Forest Regression',rf))
models.append(('Ridge Regression',ridge))
models.append(('Support Vector Regressor',svr))
models.append(('Decision Tree Regressor',dtr))
models.append(('KNeighbors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

```

As you can see above, I had called the algorithms, then I called the empty list with the name models [ ], and calling all the model one by one and storing the result in that.

We can observe that I imported the metrics in order to interpret the model's output. Then I also selected the model to find the cross_validation_score value.

Let's check the code below:

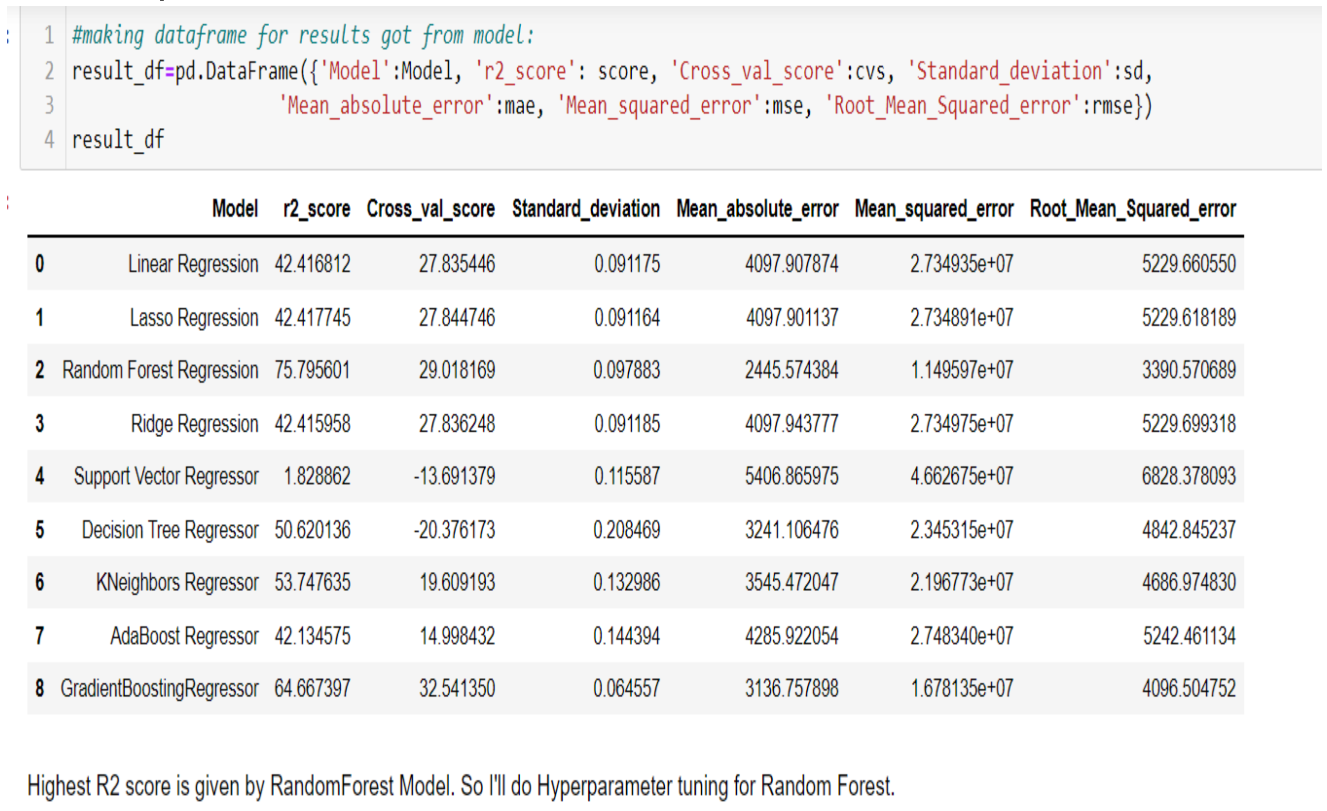


```
Model=[]
score=[]
cvs=[]
sd=[]
mae=[]
mse=[]
rmse=[]
for name,model in models:
    print('\n')
    Model.append(name)
    model.fit(x_train,y_train)
    print(model)
    pred=model.predict(x_test)
    print('\n')
    S=r2_score(y_test,pred)
    print('r2_score: ',S)
    score.append(S*100)
    print('\n')
    sc=cross_val_score(model,x,y,cv=3,scoring='r2').mean()
    print('cross_val_score: ',sc)
    cvs.append(sc*100)
    print('\n')
    std=cross_val_score(model,x,y,cv=3,scoring='r2').std()
    print('Standard Deviation: ',std)
    sd.append(std)
    print('\n')
    MAE=mean_absolute_error(y_test,pred)
    print('Mean Absolute Error: ',MAE)
    mae.append(MAE)
    print('\n')
    MSE=mean_squared_error(y_test,pred)
    print('Mean Squared Error: ',MSE)
    mse.append(MSE)
```

rmse.append(RMSE) print('\n\n') #Last 2 lines

As you can observe above, I made a for loop and called all the algorithms oneby one and appending their result to models. The same I had done to store MSE, RMSE, MAE, SD and cross validation score. Let me show the output so that we can glance the result in more appropriate way.

The following are the outputs of the different algorithms I had used, along with the metrics score obtained and after finalizing the outputs in a data frame, it willbe as follows:



```
#making dataframe for results got from model:
result_df=pd.DataFrame({'Model':Model, 'r2_score': score, 'Cross_val_score':cvs, 'Standard_deviation':sd,
                        'Mean_absolute_error':mae, 'Mean_squared_error':mse, 'Root_Mean_Squared_error':rmse})
result_df
```

	Model	r2_score	Cross_val_score	Standard_deviation	Mean_absolute_error	Mean_squared_error	Root_Mean_Squared_error
0	Linear Regression	42.416812	27.835446	0.091175	4097.907874	2.734935e+07	5229.660550
1	Lasso Regression	42.417745	27.844746	0.091164	4097.901137	2.734891e+07	5229.618189
2	Random Forest Regression	75.795601	29.018169	0.097883	2445.574384	1.149597e+07	3390.570689
3	Ridge Regression	42.415958	27.836248	0.091185	4097.943777	2.734975e+07	5229.699318
4	Support Vector Regressor	1.828862	-13.691379	0.115587	5406.865975	4.662675e+07	6828.378093
5	Decision Tree Regressor	50.620136	-20.376173	0.208469	3241.106476	2.345315e+07	4842.845237
6	KNeighbors Regressor	53.747635	19.609193	0.132986	3545.472047	2.196773e+07	4686.974830
7	AdaBoost Regressor	42.134575	14.998432	0.144394	4285.922054	2.748340e+07	5242.461134
8	GradientBoostingRegressor	64.667397	32.541350	0.064557	3136.757898	1.678135e+07	4096.504752

Highest R2 score is given by RandomForest Model. So I'll do Hyperparameter tuning for Random Forest.

We can see that Random Forest Regression algorithms are performing well, as compared to other algorithms. Now we will try Hyperparameter Tuning to find out the best parameters and try to increase their scores.

Key Metrics for success in solving problem under consideration

The key metrics used here were r2_score, cross_val_score, sd, MAE, MSE and RMSE. We tried to find out the best parameters and also to increase our scores by using Hyperparameter Tuning and we will be using GridSearchCV method.

1. Cross Validation:

Cross-validation helps to find out the over fitting 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 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 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. R2 Score:

It is a statistical measure that represents the goodness of fit of a regression model. The ideal value for r-square is 1. The closer the value of r-square to 1, the better is the model fitted.

3. Mean Squared Error (MSE):

MSE of an estimator (of a procedure for estimating an unobserved quantity) measures the average of the squares of the errors — that is, the average squared difference between the estimated values and what is estimated. MSE is a risk function, corresponding to the expected value of the squared error loss. RMSE is the Root Mean Squared Error.

4. Mean Absolute Error (MAE):

MAE measures the average magnitude of the errors in a set of predictions, without considering their direction. It’s the average over the test sample of the absolute differences between prediction and actual observation where all individual differences have equal weight.

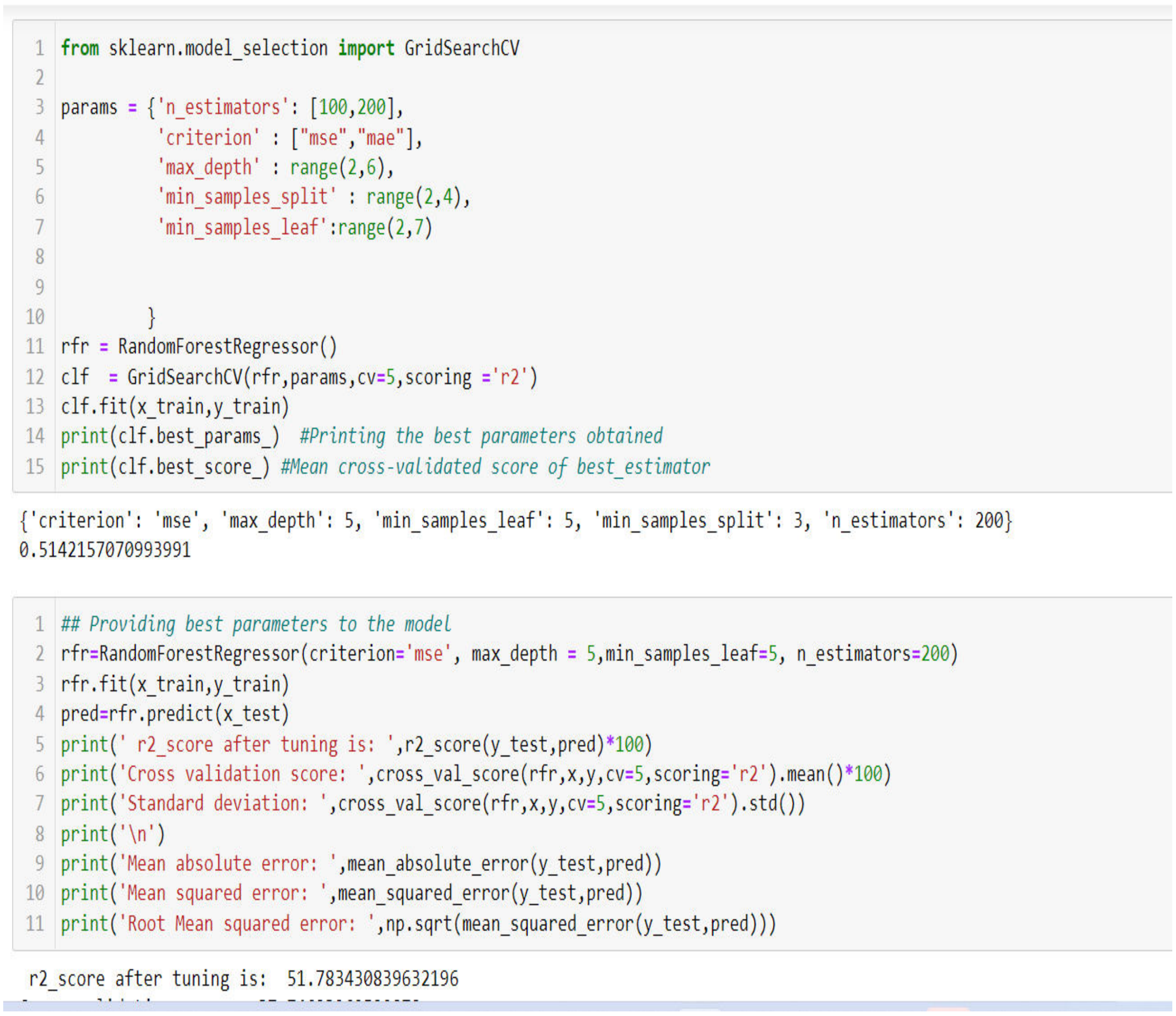
5. Hyperparameter Tuning:

There is a list of different machine learning models. They all are different in some way or the other, but what makes them different is nothing but input parameters for the model. These input parameters are named as **Hyperparameters.** These hyperparameters will define the architecture of the model, and the best part about these is that you get a choice to select these for your model. You must select from a specific list of hyperparameters for a given model as it varies from model to model.

We are not aware of optimal values for hyperparameters which would generate the best model output. So, what we tell the model is to explore and select the optimal model architecture automatically. This selection procedure for hyperparameter is known as **Hyperparameter Tuning.** We can do tuning by using **GridSearchCV.**

GridSearchCV is a function that comes in Scikit-learn (or SK-learn) model selection package. An important point here to note is that we need to have Scikit-learn library installed on the computer. This function helps to loop through predefined hyperparameters and fit your estimator (model) on your training set. So, in the end, we can select the best parameters from the listed hyperparameters.

Random Forest



```python
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
```

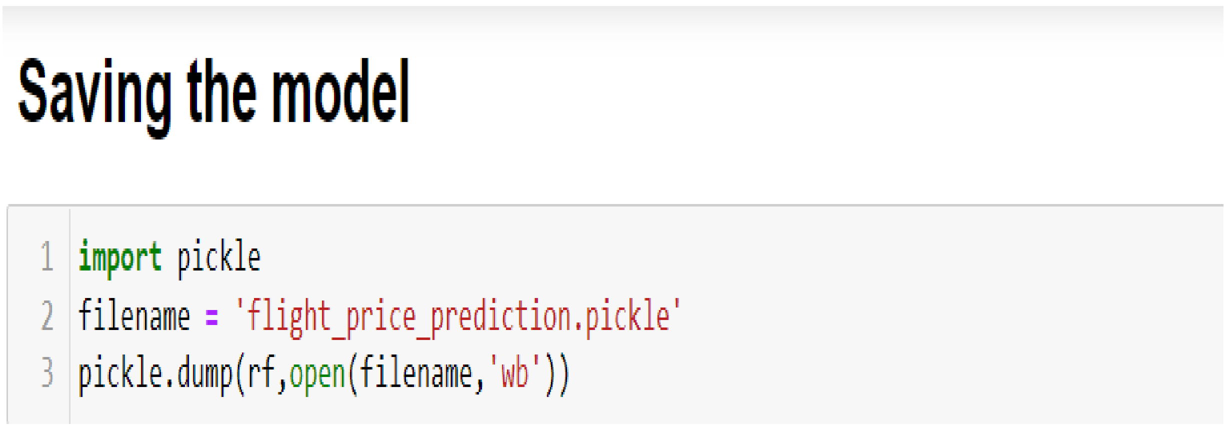
```
{'criterion': 'mse', 'max_depth': 5, 'min_samples_leaf': 5, 'min_samples_split': 3, 'n_estimators': 200}
0.5142157070993991
```

```python
## Providing best parameters to the model
rfr=RandomForestRegressor(criterion='mse', max_depth = 5,min_samples_leaf=5, n_estimators=200)
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:  51.783430839632196
```

After applying Hyperparameter, we can see that Random ForestRegressor is the best performing algorithm among all other algorithms as it is giving a r2_score of 51.78 and cross validation score of 37.74.

Final the model



Saving the model

```python
import pickle
filename = 'flight_price_prediction.pickle'
pickle.dump(rf,open(filename,'wb'))
```

Visualizations

Now, we will see the different plots done with this dataset in order to know the insight of the data present. Below are the codes given for the plots and the output obtained:

Importing required libraries and plotting graphs for categorical data

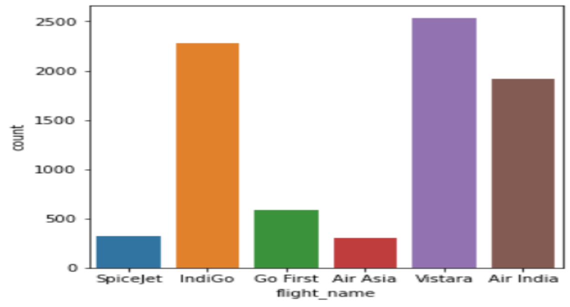
Below are some of the graphs we obtain after running this code:

Univariate Analysis

```
categorical_cols = df.select_dtypes(include=['object']).columns  # checking categorcial columns

```

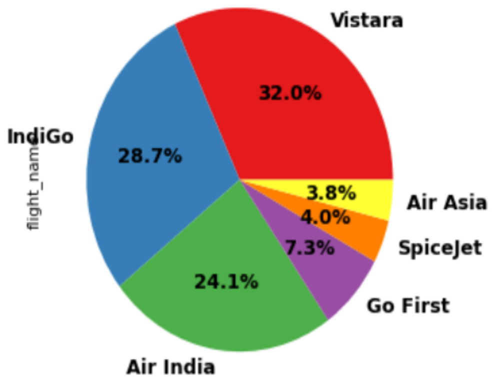
```
Index(['flight_name', 'flight_id', 'source', 'destination'], dtype='object')
```

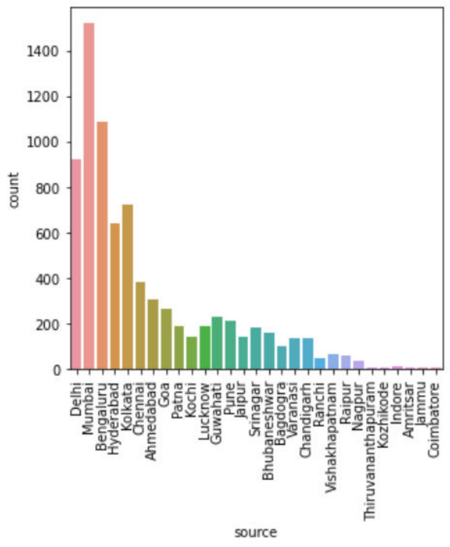
```
plt.figure(figsize=(5,5))
sns.countplot(df['flight_name'])
plt.show()
```



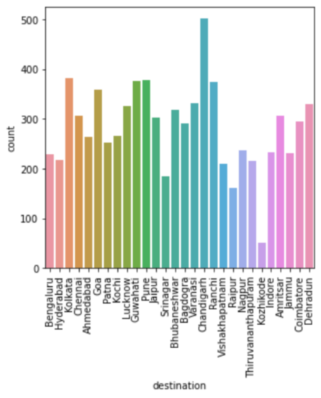
```
# Pie & count plot of Airline types
df['flight_name'].value_counts().plot.pie(autopct='%2.1f%%',
                                          textprops ={ 'fontsize':13,'fontweight' :'bold'})

```

```
<AxesSubplot:ylabel='flight_name'>
```





Count of Mumbai as source staion is more than any other cities.

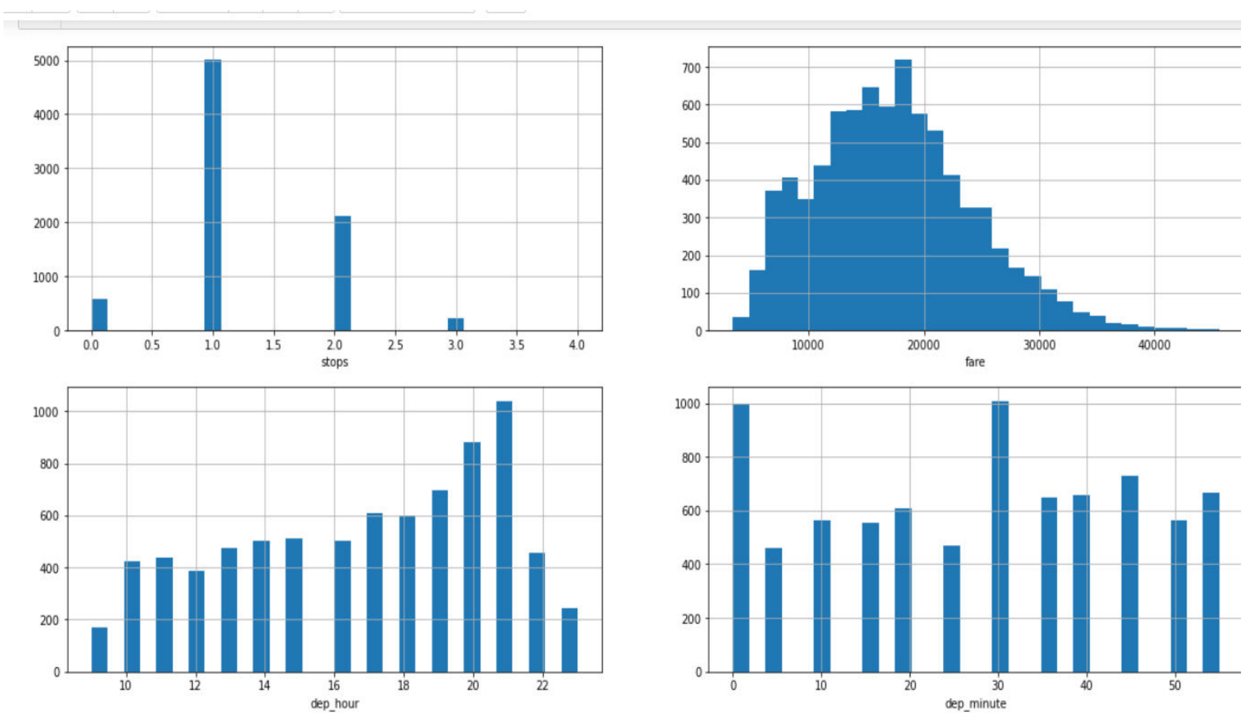


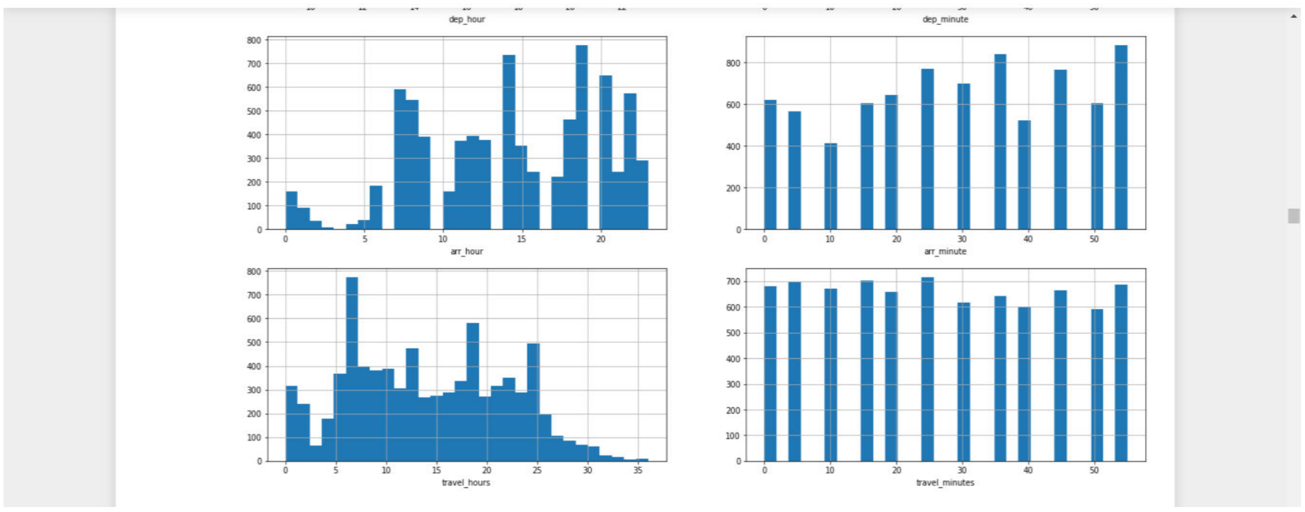
count of Chandigarh as destination is more than any other cities.

Observations:

- ➔Vistara flights are more in counts followed by Indigo and Air India.
- ➔Count of Mumbai as source station is more than any other cities.
- ➔count of Chandigarh as destination is more than any other cities.

Taking all continuous data and plotting histogram

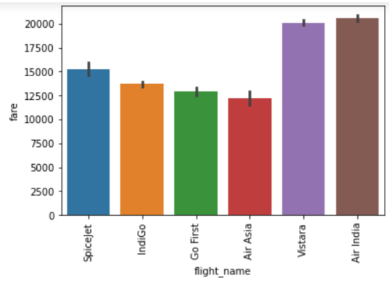




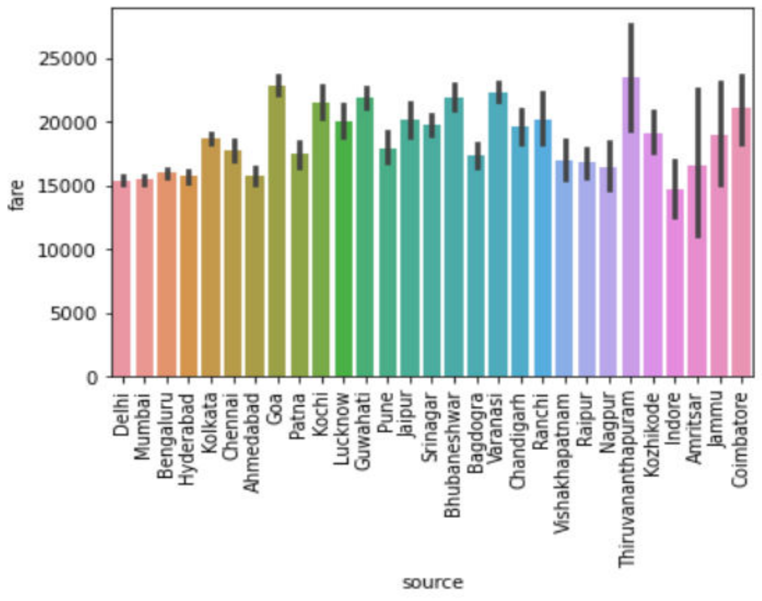
Observations:

- ➔ **Most of stops are 1 stop followed by 2nd stop.**
- ➔ **Most of fare ranges between 10,000 to 25,000 rs.**

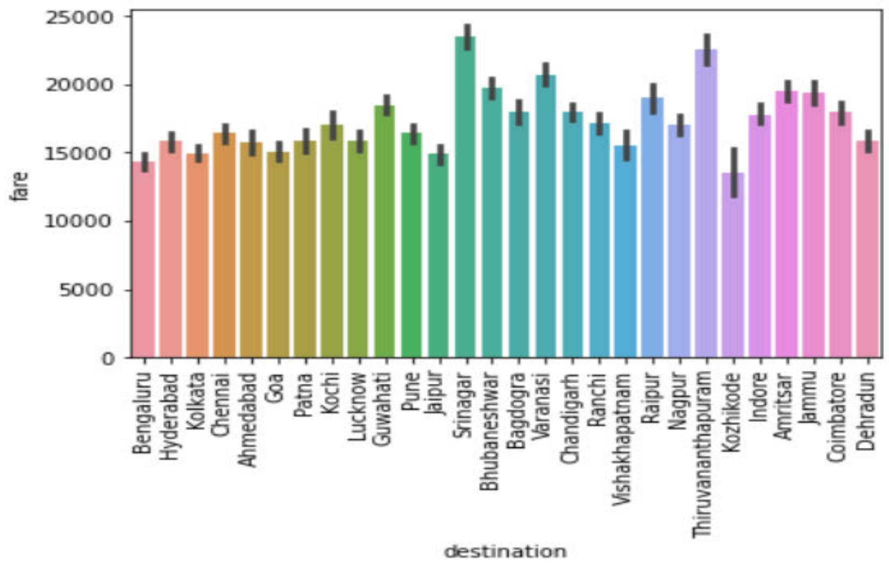
Bivariate Analysis with ‘fare’ for categoricaldata



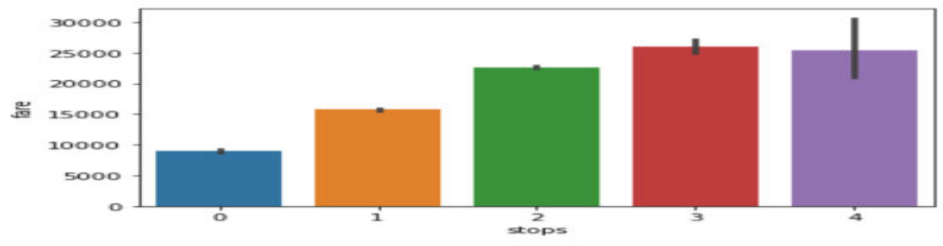
Air India flight's fares are costly followed by Vistara flights.



From Goa,Thiruvananthpuram as source ,fares are high.



Bhubaneshwar and Thiruvananthpuram as destination,fares are high.



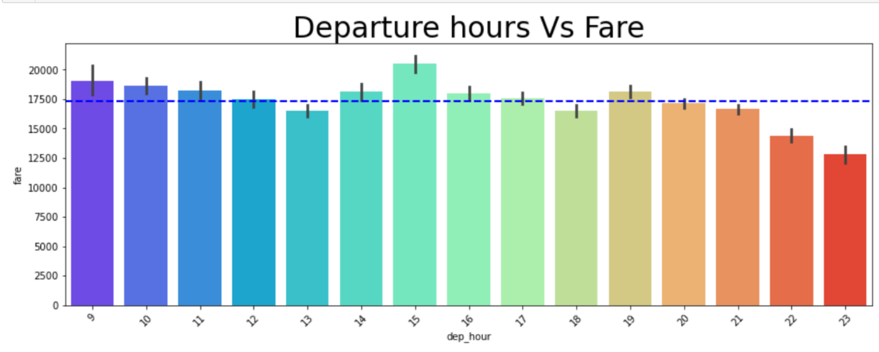
-> More stops ,higher fare. -> fare is lowest for non stop flights.

Above are some of the plots obtained and its value counts for bivariate analysis of all columns with the target variable ‘fare’.

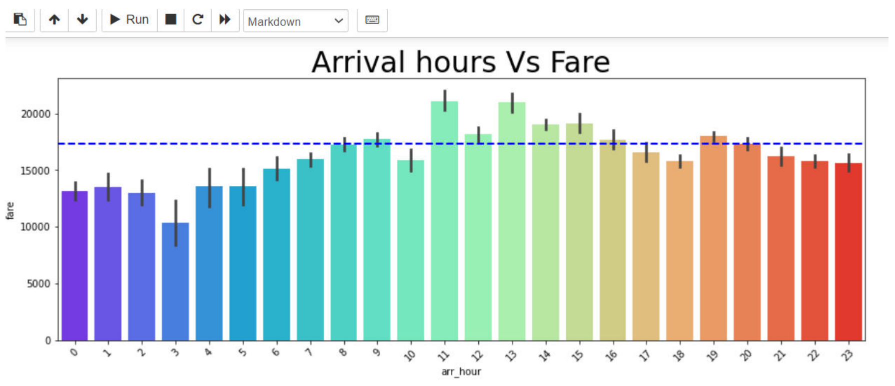
Observations:

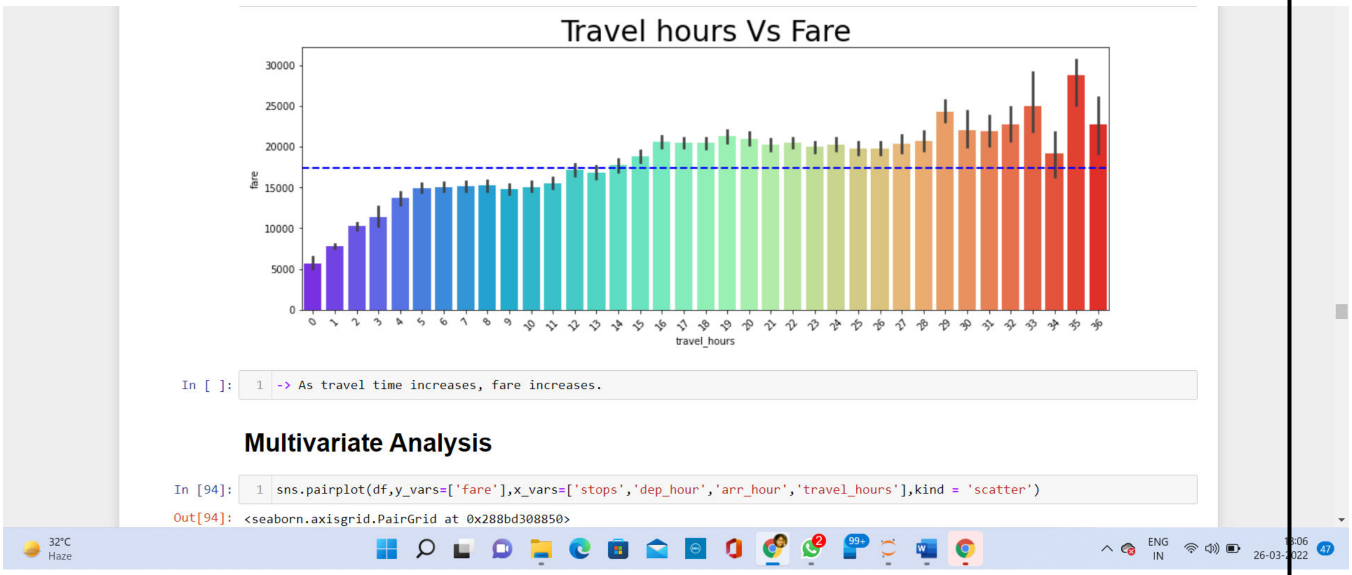
- ➔ Air India flight's fares are costly followed by Vistara flights.
- ➔ From Goa, Thiruvananthapuram as source, fares are high.
- ➔ Bhubaneshwar and Thiruvananthapuram as destination, fares are high.
- ➔ More stops, higher fare.
- ➔ fare is lowest for non -stop flights.

Plotting for continuous data



-> Late night, after 9pm flight's fare are cheaper. -> Afternoon flights 2pm to 5pm ,fare is higher.





```
In [ ]:  -> As travel time increases, fare increases.
```

Multivariate Analysis

```
In [94]:  sns.pairplot(df,y_vars=['fare'],x_vars=['stops','dep_hour','arr_hour','travel_hours'],kind = 'scatter')
Out[94]: <seaborn.axisgrid.PairGrid at 0x288bd308850>
```

Observations:

- ➔ **Late night, after 9pm flight's fare are cheaper.**
- ➔ **Afternoon flights 2pm to 5pm, fare is higher.**
- ➔ **Arriving flights at 11 am and 13 pm, those flight's fare is higher.**
- ➔ **As travel time increases, fare increases.**

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 pre-processing 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 HyperparameterTuning.
-> 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 newdata from the client we work on and we can proceed our analysis by loading the best model we obtained and start working on the analysisof 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.

- Most No of Flights are available from Mumbai.
- Among Vistara, Air India, Indigo, Go First, Air Asia, SpiceJet flights, Air India flights are most expensive followed by Vistara flights.
- Non -Stop Flights are cheaper than flights including stops.
- As Travel Hour increases, Fare Increases.
- Vistara flights availability is more than any other airlines for different places.
- 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.

Limitations of this work and Scope for Future Work

1. The biggest limitation is less features, that’s why my model is overfitted.
2.we can increase the efficiency of this model by scrapping more features like different class, additional info and week days and weekends and taking large dataset to make a better model.