CAPSTONE PROJECT

FINAL REPORT

PROJECT SUMMARY

Batch Details	PGP DSE MAR 2021
Team Members	- Ankith Ravi - Madav Saravanan S - Raymond I - Saravana Kumar Shanmuganathan - A.P.Shalini Priya - Swathy Lakshmi Gandhi K
Domain of Project	Aviation
Proposed Project Title	Taxi-Out (Run time)Prediction for better optimizing flight cost
Group Number	3
Mentor Name	Mr. Sravan Malla

ACKNOWLEDGEMENT

We would like to thank our mentor Mr. Sravan Malla for providing his valuable guidance and suggestions over the course of our Project Work.

We also thank him for his continuous encouragement and interest towards our Project work. We are extremely grateful to all our teaching and non-teaching staff members of GREAT LEARNING, who showed keen interest in progress.

We greatly admire and acknowledge the constant support we received from our friends and team members for all the effort and hard work that they have put into completing this project

TABLE OF CONTENTS

1.1.INTRODUCTION

1.1.PROBLEM STATEMENT

The airline industry encompasses a wide range of businesses called airlines, which offer air transport services for paying customers or business partners. These air transport services are provided for both human travelers, cargo, and are most commonly offered via jets, although some airlines also use helicopters.

Flights incur a large percentage of their delays on the ground during the departure process

Between the scheduled departure from the gate and takeoff. Because of the large uncertainties associated with them, these delays are difficult to predict and account for, hindering the ability to effectively manage the Air Traffic Control (ATC) system. This project presents an effort to improve the accuracy of estimating the taxi-out time. The method is to identify the main factors that affect the taxi-out time and build a Regression model taking the TAXI-OUT as the target variable.

This file contains data about flights leaving from JKF airport between Nov 2019 and Dec-2020.

Taxi-Out prediction has been an important concept as it helps in calculating Runway time and directly impact the cost of the flight.

1.2 INDUSTRY REVIEW

Current practices:

Airline alliances allow airlines to share frequent flyer programs.An airlines basic function is to transport passengers and their luggage from one point to another. Just like any other service industry, the airline industry provides a service for a set price. The airline industry encompasses a wide range of businesses called airlines, which offer air transport services for paying customers or business partners. These air transport services are provided for both human travelers, cargo, and are most commonly offered via jets, although some airlines also use helicopters.

The Flight Operations Support Departments role is to coordinate all these technical and operational factors such as the weather, overflight permits, route planning, aircraft performance, airport facilities, the aircrafts technical condition or fuel requirements. Whether a flight can be performed

safely and in the most efficient conditions depends on a number of significant factors that must be under control from the initial planning until the aircraft is parked at its destination. The Flight Operations Support Departments role is to coordinate all these technical and operational factors such as the weather, overflight permits, route planning, aircraft performance, airport facilities, the aircrafts technical condition or fuel requirements. The department has a key role within the company.

Americans love to travel, as is witnessed by the hordes of travellers at the airport nowadays. In the United States, 665-million people travelled on at least one U.S. airline in 2000. Twenty-five-thousand (25,000) flights depart every day from American airports, and Americans are expected to travel even more in 2001.

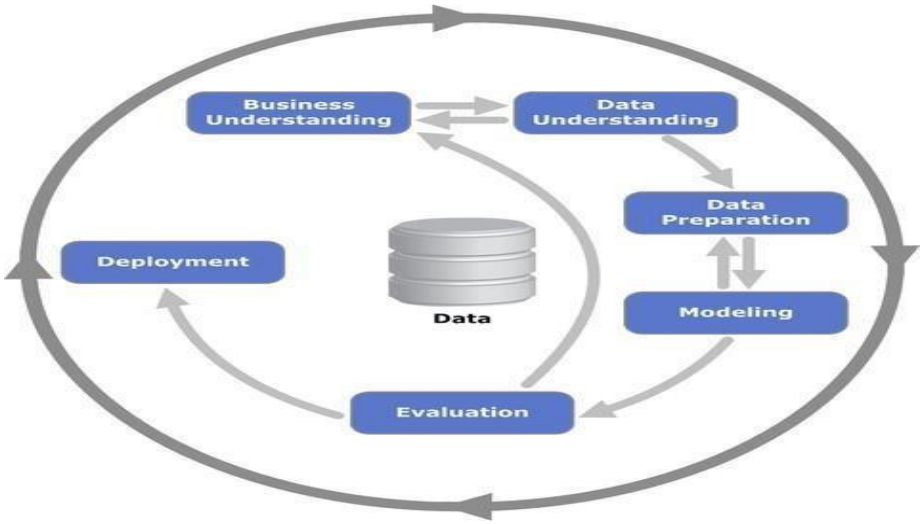
Background Research:

The origin of the aerospace industry dates to 1903 when Wilbur and Orville Wright demonstrated an airplane capable of powered, sustained flight. The Wright brothers success was due to detailed research and an excellent engineering-and-development approach. Their breakthrough innovation was a pilot-operated warping (twisting) of the wings to provide attitude control and to make turns. Perhaps the most important reason for the rapid development of airplanes in the 20th century was the popularity of **using airplanes in warfare**.

Air transport has changed significantly over the last 50 years.There are more passengers, flying further for longer distances. In the last five years alone the number of flights over 6000 nautical miles (13+ hours flight time) has increased by 70% from 24 to 41 daily flights.

In 1998 no flight over 7000 nautical miles had ever taken place. In the next 15 years passenger traffic will double and by 2032, the worlds airlines will take delivery of more than 29,220 new passenger and freighter aircraft.

1.3.METHODOLOGY



The process or methodology of is described in these six major steps

1. **Business Understanding**

Focuses on understanding the project objectives and requirements from a business perspective, and then converting this knowledge into a data mining problem definition and a preliminary plan.

2. **Data Understanding**

Starts with an initial data collection and proceeds with activities in order to get familiar with the data, to identify data quality problems, to discover first insights into the data, or to detect interesting subsets to form hypotheses for hidden information.

3. **Data Preparation**

The data preparation phase covers all activities to construct the final dataset from the initial raw data.

4. **Modeling**

Modeling techniques are selected and applied. Since some techniques like neural nets have specific requirements regarding the form of the data, there can be a loop back here to data prep.

5. **Evaluation**

Once one or more models have been built that appear to have high quality based on whichever loss functions have been selected, these need to be tested to ensure they generalize against unseen data and that all key business issues have been sufficiently considered. The end result is the selection of the champion model(s).

6. Deployment

Generally this will mean deploying a code representation of the model into an operating system to score or categorize new unseen data as it arises and to create a mechanism for the use of that new information in the solution of the original business problem. Importantly, the code representation must also include all the data prep steps leading up to modeling so that the model will treat new raw data in the same manner as during model development.

2.DATASET AND DOMAIN

2.1 DATASET SOURCE:

The data used in this project was retrieved from Kaggle which is used to Predicts Taxi_out in the Aviation industry. The Dataset contains the information related to the features of the flights leaving from JKF airport between Nov 2019-Dec-2020 details. The Dataset comprises 23 features and 27,000 data points.

2.2 DATA DICTIONARY:

S.No	Column Name	Data Description
1	MONTH	Scheduled month of the flights.
2	Day_of_Month	Date of flight.
3	DAY_OF_WEEK	Days of the week. 1(Monday)/2(Tuesday)/3(Wednesday)/4 (Thursday)/5(Friday)/ 6 (Saturday)/ 7(Sunday)
4	CARRIER_CODE	Carrier Code (Should generally be carrier company).
5	Tail_Num	Airflight Number.
6	DEST	Destination of the flight.
7	DEP_DELAY	Departure delay of the flight.
8	SCHEDULED_DURATION	Scheduled journey time of the flight in minutes
9	DISTANCE	Distance of the journey (distance between the locations).
10	SCHEDULED_DEPARTURE	Scheduled Departure Time.
11	ACTUAL_DEP_TIME	Actual Departure Time (Gate checkout of the flight not the take-off time).
12	SCHEDULED_ARRIVAL	Scheduled Arrival Time.
13	Temperature	Temperature of departure location
14	Dew Point	Dew (The air temperature at which a sample of air

		would reach 100% humidity based upon its current degree of saturation).
15	Humidity	Humidity (the amount of water vapor in the air).
16	Wind	Direction of the wind.
17	Wind Speed	The vector difference between the airspeed and the ground speed.
18	Wind Gust	Sudden increases in wind speed that typically last no longer than 20 seconds
19	Pressure	The air pressure in the altitude inside the flight.
20	Condition	Condition of the climate.
21	FLT_SCH_ARRIVAL	Number of flights scheduled for arrival.
22	FLT_SCH_DEPARTURE	Number of flights scheduled for departure.
23	TAXI_OUT	Taxi-out time (Run away Time) (Target variable).
24	Wind_Gust_cat	Wing- Gust as a category
25	Speed	Speed in Km per Hour
26	Region	DEST is reformed as Region (West,South,North east,Midwest)
27	TOTAL_SCHEDULED	Total number of flights schedules for arrival and departure
28	FLIGHT_TRAFFIC	Adding a new feature called FLIGHT_TRAFFIC which uses the TOTAL_SCHEDULED values and labels to show the status of the airport.

2.3 IRRELEVENT COLUMNS:

Column name	Reason
TAIL_NUM	The tail number do not contribute to the taxi-out time hence we can drop it
MONTH	MONTH as it holds only 11, 12, 1 and it is not going to help the model for prediction
DAY_OF_MONTH	DAY_OF_MONTH aswell holds few unique values, it is not going to help the model for prediction
DEST	Destination is reframed into Region

2.4 DATA PROCESSING:

We have renamed the following variables for better Understanding

Column Name	New Column name
OP_UNIQUE_CARRIER	SCHEDULED_DURATION
CRS_ELAPSED_TIME	SCHEDULED_DURATION
CRS_DEP_M	SCHEDULED_DEPARTURE
DEP_TIME_M	ACTUAL_DEP_TIME
CRS_ARR_M	SCHEDULED_ARRIVAL
sch_dep	FLT_SCH_ARRIVAL
sch_arr	FLT_SCH_DEPARTURE

Wind_Gust_cat

It is generally reported in METAR When the maximum speed exceeds the average speed by 10 to 15 knots, the term gusts is used while strong gusts is used for departure of 15 to 25 knots, and violent gusts when it exceeds 25 knots, ans so Wind-Gust is categorized into No Gust, Average Gust, strong Gust, violent Gust.

Dew_Points

The columns Dew_points had white spaces and it was treated.

Wind

Wind` is the direction of the wind ,Wind columns has been jotted down into 6 variables namely West, North,South,East, calm and var (`CALM` if calm, `VAR` if wind blows from various directions)

Dep_Delay

Formally DEP_DELAY was numerical, it is basically the difference between Scheduled_departure and Actual departure time. So , it has been categorized as Early,On time and Delay.

Scheduled_Departure, Actual_Dep_Time & Scheduled_Arrival

These columns contains the railway timings of departure and arrival, So it has been categorized using bins with labels Midnight,Morning, Noon,Evening,Night.

Speed

Speed in measured by Distance / Scheduled duration (time taken) , New feature speed is added.

Region

Adding new column Region via. DEST (Destination **) as** West,South,North east,Midwest.

Total Scheduled

Total number of flights schedules for arrival and departure (FLT_SCH_ARRIVAL + FLT_SCH_DEPARTURE)

TAXI_OUT

Taxi out is the Target column. Since we couldn't arrive at a Good score Using Regression we switched the analysis to Classification, which will be discussed further. This continuous column is categorized into High and Low using Bins.

3.PRE PROCESSING DATA ANALYSIS :

3.1.Null Value Imputation:

- There are totally 2 null values in our dataset.

These null values are from a single column – WIND .

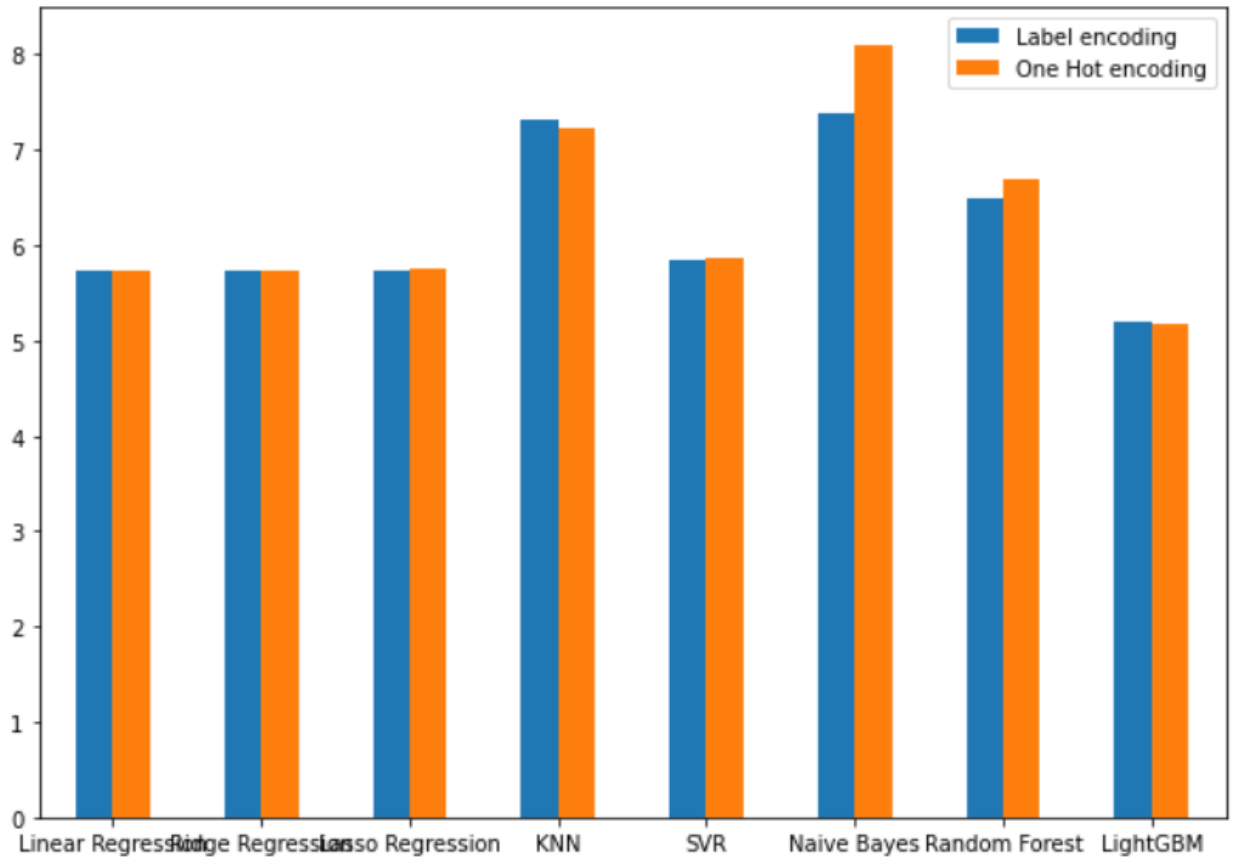
- Total % of null values in the dataset is 0.006 %.
- We have used 3 techniques to replace the null values.
    1. Replacing it with mode.
    2. Removing the null value rows.
    3. Using the group by technique and finding the relevancy compared with the CNS_Score_desc column.
- For the 2 null value, we have removed those columns

3.2 Encoding:

The categorical features such as Carrier_Code, Dest, Dep_Delay, Scheduled_Departure,

- Actual_Depature, Scheduled_Arrival, Wind, Condition, Region and Flight_Traffic have been en coded via One Hot Encoding and Label Encoding separately.
- With both encoding methods, Models have been build separately, RMSE scores of both methods have been compared.

- Label encoding is better in predicting Taxi out time in almost all algorithms.
- One hot encoding produces exceptionally big RMSE in Naive Bayes algorithm.
- On the basis of RMSE we can say that LightGBM model is best model in both Label encoded as well as One Hot Encoded data.
- So, we choose Label Encoding as our Encoding method.



4. EXPLORATORY DATA ANALYSIS

Data Analysis refers to the critical process of performing initial investigations on data so as to discover patterns, to spot anomalies, to test hypotheses and to check assumptions with the help of summary statistics and graphical representations. Since numerous features are present in the dataset, certain features need to be dropped to proceed with feature engineering and exploratory data analysis privy to the missing value treatment which helps to gain more insights.

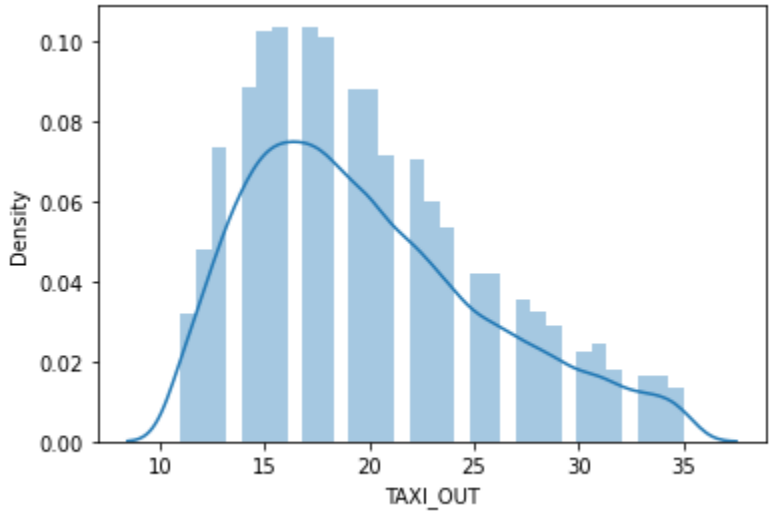
4.1 Relationship between variables:

Univariate analysis:

Univariate analysis is the simplest form of analysing data. It doesn't deal with causes or relationships. Its major purpose is to describe; It takes data, summarizes that data and finds patterns in the data.

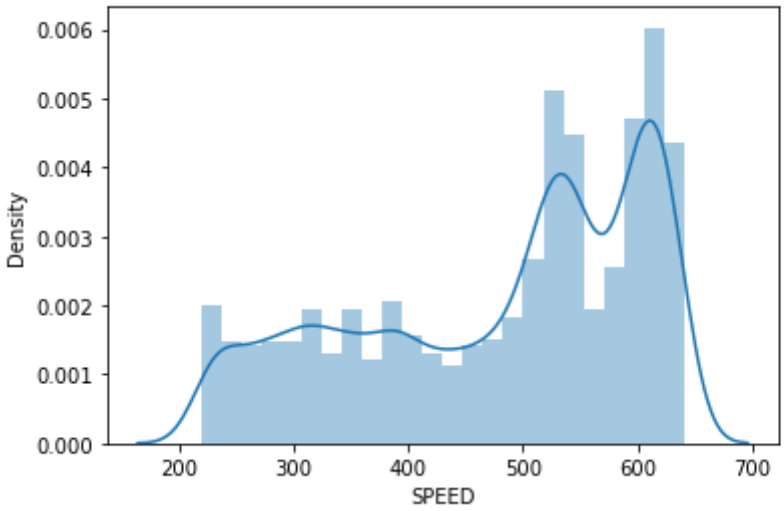
Taxi out:

Taxi-Out is the target variable. It is a continuous variable and it lies in range 5 and 41.



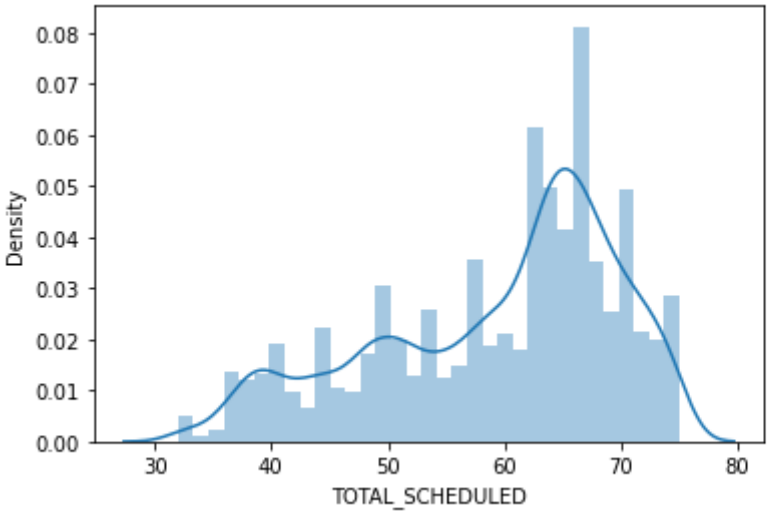
Speed:

Speed in measured by Distance / Scheduled duration (time taken) , Speed in measured in Kilometer per hour. Its skewness is -0.566



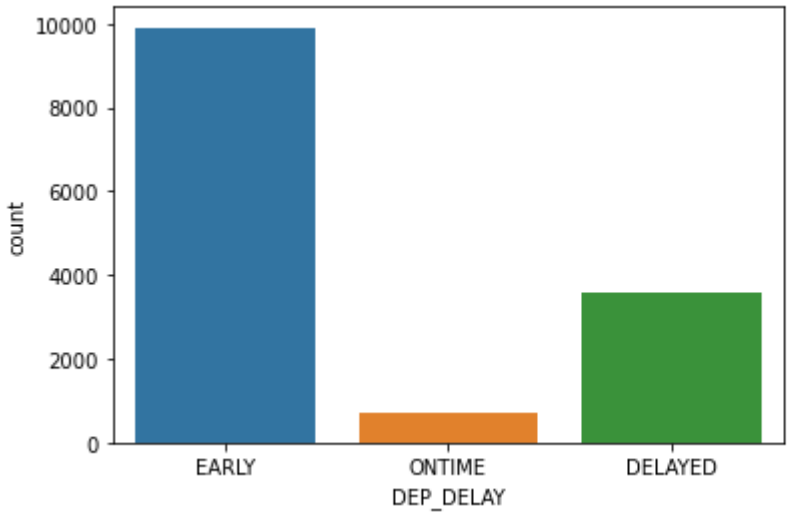
Total scheduled:

Total Number of flights scheduled for arrival and departure in a day, minimum 1 and maximum is 83.



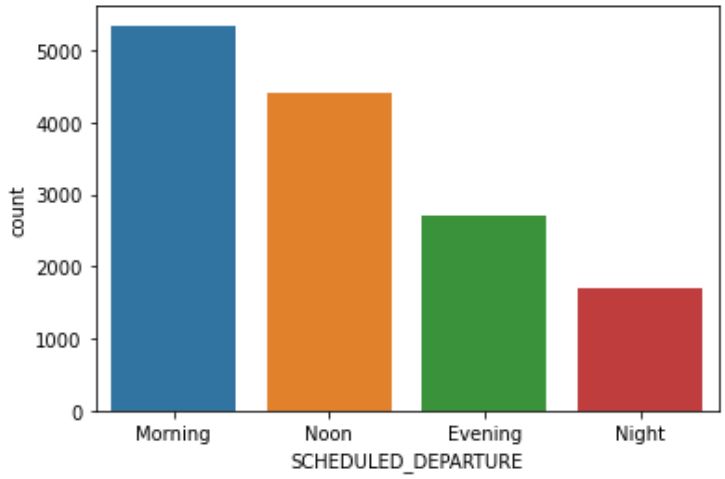
Dep_delay

Flights are 5.08 % on-time , 27.07% Delayed and 67.83% Early.



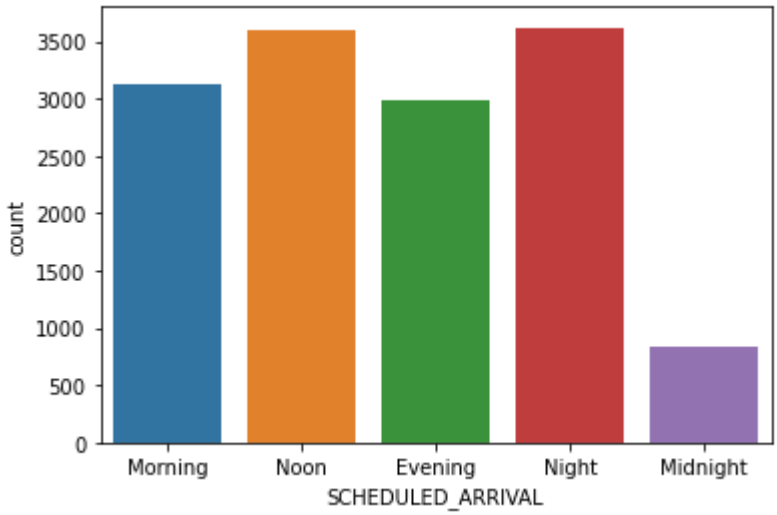
Schduled departure:

Most of the Flights scheduled for departure in the morning and very less during the night.



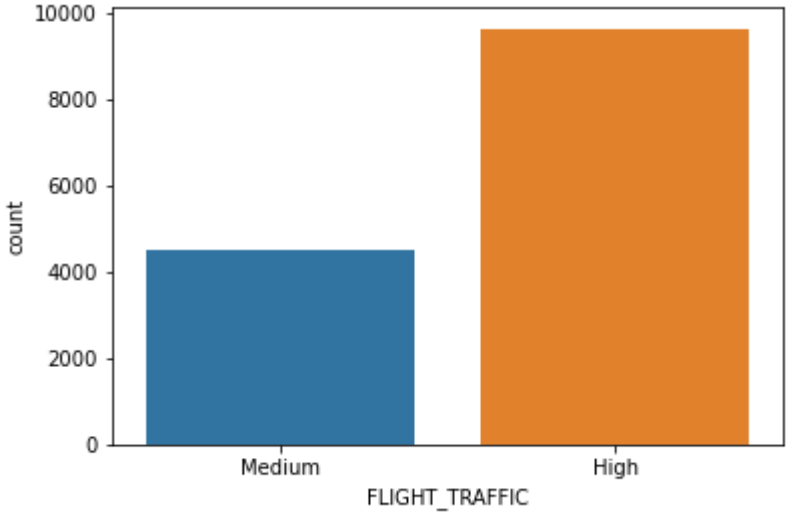
Scheduled arrival:

Most of the Flights are schedules for arrival during the night and noon and very less by Midnight.


3500
3000
2500
2000
1500
1000
500
0
count
Morning
Noon
Evening
Night
Midnight
SCHEDULED_ARRIVAL

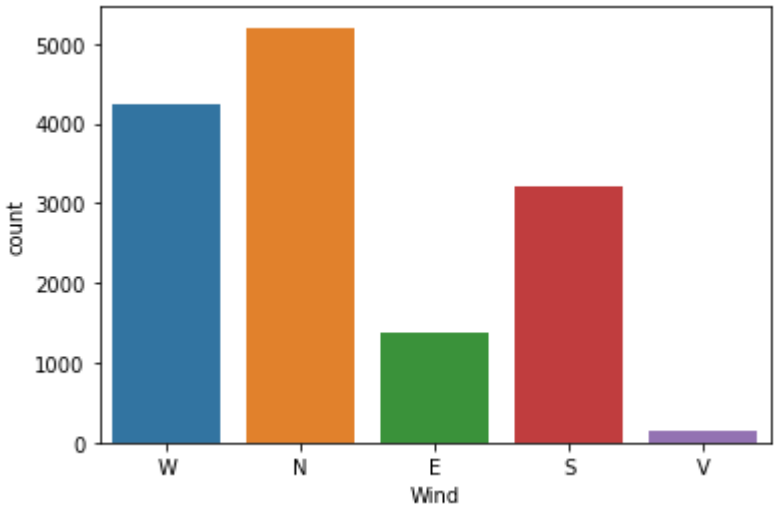
Flight_traffic:

There are rare cases where there is no Flight traffic. Flight traffic leads to more Taxi_out time.



WIND:

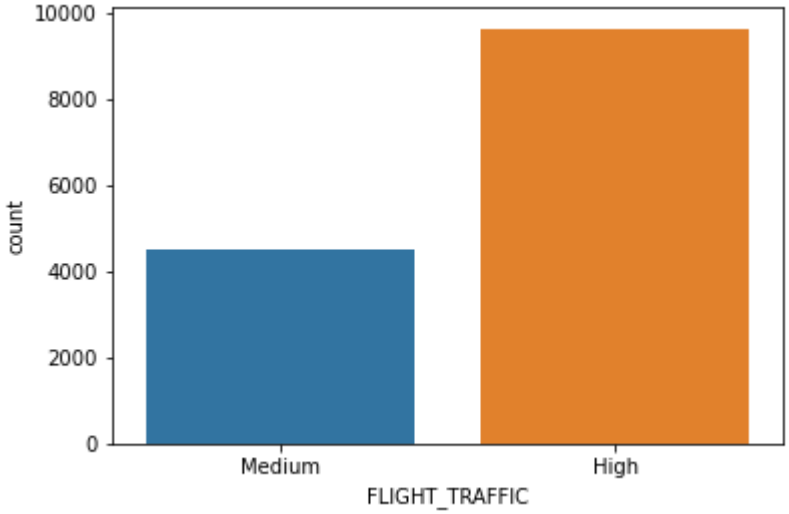
Wind direction is mostly Towards north and There is rare scenario of ‘calm’.



BIVARATE ANALYSIS

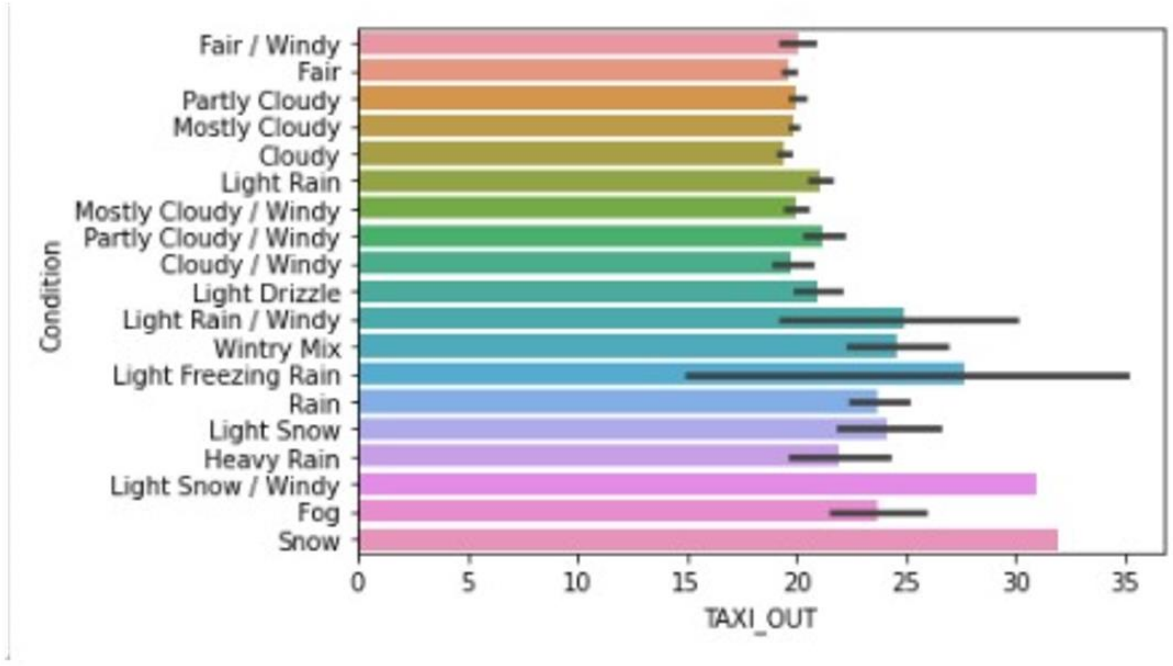
Flight traffic and Taxi_out:

Flight traffic and taxi out – taxi out seems to be high when there is high traffic and low taxi out when flight traffic is less.



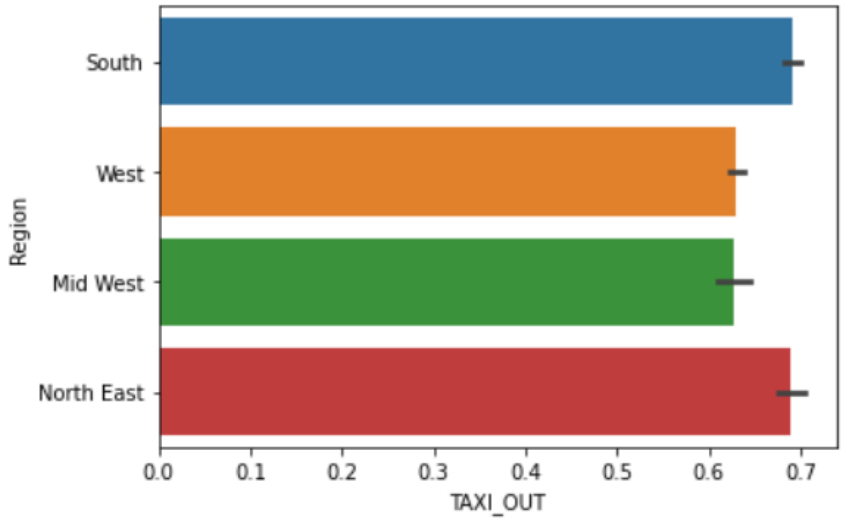
Condition and Taxi_out:

Conditions like snow,windy,freezing rain influence on increase in taxi_out.



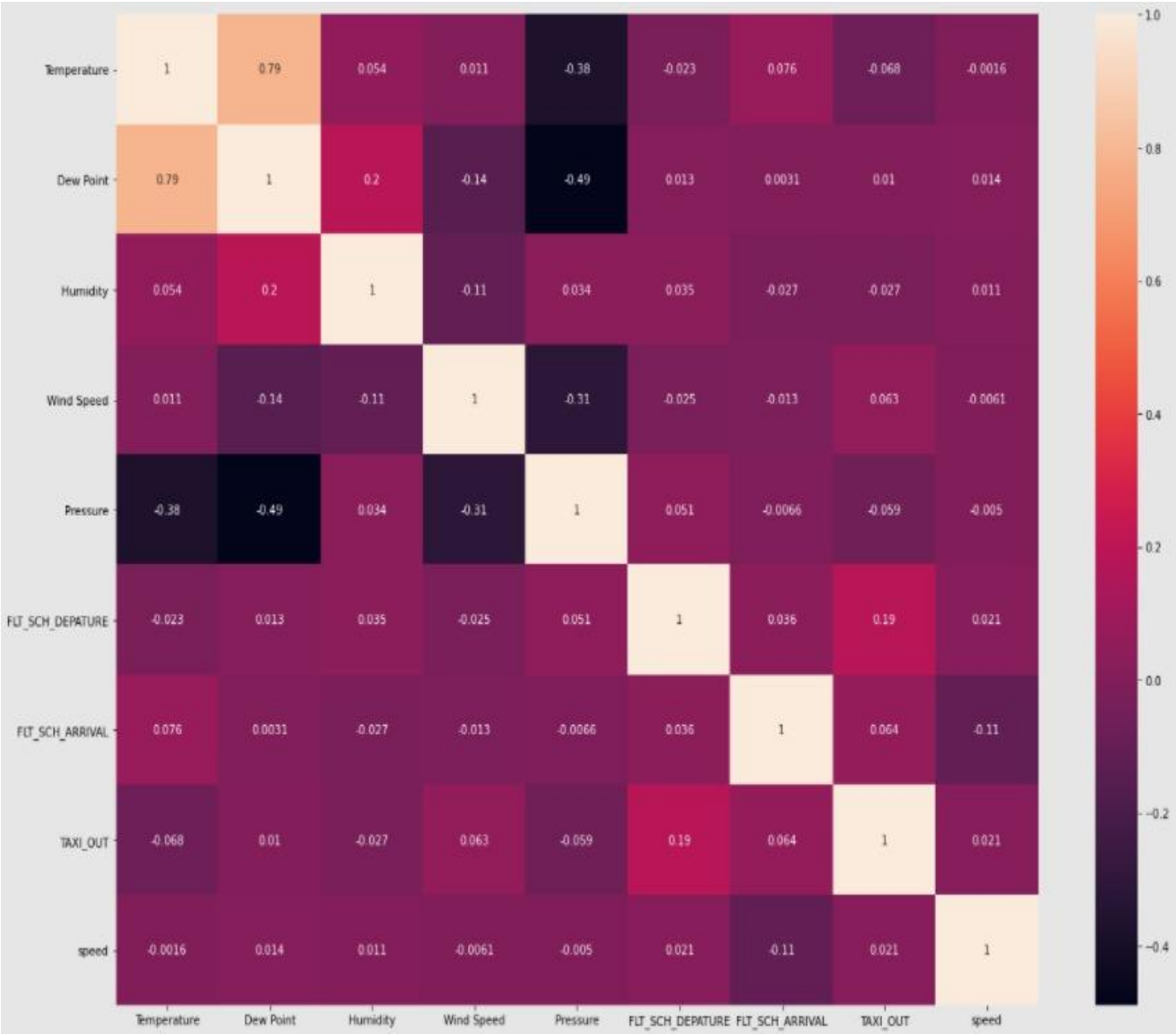
Region and Taxiout

West and Mid west region have lesser Taxi out timing comparatively. South and Northeast region has more taxi out.



Correlation matrix:

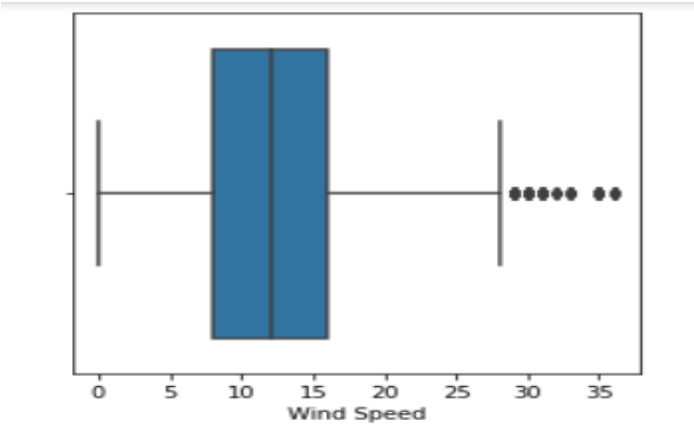
There is a correlation between Temperature and Dew points.



4.2 Treatment of outliers:

There are about 6 Numeric columns in the dataset after VIF treatment such as Dew Point, Humidity, Win d Speed, Flt_Sch_Arrival, Taxi_Out and Speed.Box-plot is used to visualize these numerical columns to check for the presence of outliers.

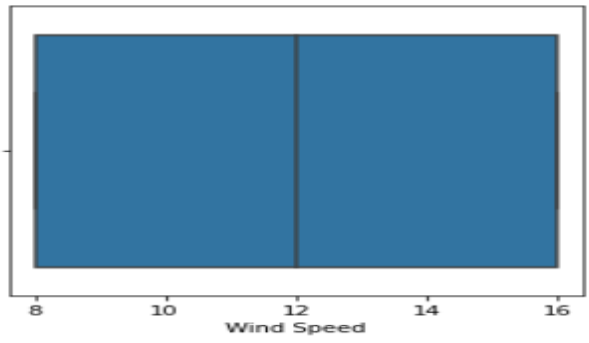
Example for presence of outliers:



Winsorize is used to Treat outliers. **Winsorization** is a way to minimize the influence of outliers in your data by either:

- Assigning the outlier a lower weight,
- Changing the value so that it is close to other values in the set.

Example After Treating Outliers:



5.Statistical Modelling and Assumptions:

5.1.LINEAR REGRESSION MODEL:

Linear regression model means estimating the values of the coefficients used in the representation with the data that we have available. With simple linear regression when we have a single input, we can use statistics to estimate the coefficients. This requires to calculate statistical properties from the data such as means, standard deviations, correlations and covariance. All of the data must be available to traverse and calculate statistics.

To evaluate the model, we use the metrics R2 score, Root Mean Squared Error and Mean Absolute Percentage Error. Applying linear regression on the Train and Test we get the following results.

	R2 Train	R2 Test	RMSE Train	RMSE Test
Base Model	0.019105	0.016587	5.705372	5.672939

5.2.FEATURE ELIMINATION AND TUNING:

Feature Selection is one of the main steps of the pre-processing phase as the features which we choose directly affects the model performance. While some of the features seem to be less useful in terms of the context, others seem to equally useful. The better features we use the better our model will perform. Irrelevant or partially relevant features can negatively impact model performance. Feature selection and Data cleaning should be the first and most important step of your model designing. Feature Selection is the process where you automatically or manually select those features which contribute most to your prediction variable or output in which you are interested in. Having irrelevant features in your data can decrease the accuracy of the models and make your model learn based on irrelevant features.

BACKWARD ELIMINATION:

Backward elimination is a feature selection technique while building a machine learning model. It is used to remove those features that do not have a significant effect on the dependent variable or prediction of output.

FORWARD ELIMINATION:

Forward elimination is a type of stepwise regression which begins with an empty model and adds in variables one by one. In that, you start with a model that includes every possible variable and eliminate the extraneous variables one by one.

RECURSIVE FEATURE ELIMINATION:

Recursive Feature Elimination technique is to choose the desired number of most important features. The Recursive Feature Elimination (or RFE) works by recursively removing attributes and building a model on those attributes that remain. It uses the model accuracy to identify which attributes (and combination of attributes) contribute the most to predicting the target attribute. We can use it directly from the scikit library by importing the RFE module or function provided by the scikit.

We have done Backward elimination, Forward Elimination and Recursive feature Elimination and found the below results.

	Full Model	Backward Elimination	Forward Selection	RFE
R2 Train	0.019105	0.016743	0.016743	0.017573
R2 Test	0.016587	0.016294	0.016294	0.017215
RMSE Train	5.705372	5.712238	5.712238	5.709825
RMSE Test	5.672939	5.673786	5.673786	5.671129

5.3.FEATURE ENGINEERING:

Scaling:

In scaling the numerical columns here we use the Standard Scaler. StandardScaler is that it will transform your data such that its distribution will have a mean value 0 and standard deviation of 1.Then we are checking the p-values of the features.If the pvalue is greater then 0.05 then we remove the insignificant variable .

Feature selection:

Feature Selection is one of the main steps of the pre-processing phase as the features which we choose directly affects the model performance. While some of the features seem to be less useful in

terms of the context, others seem to equally useful. The better features we use the better our model will perform. Irrelevant or partially relevant features can negatively impact model performance.

Feature selection and Data cleaning should be the first and most important step of your model designing. Feature Selection is the process where you automatically or manually select those features which contribute most to your prediction variable or output in which you are interested in. Having irrelevant features in your data can decrease the accuracy of the models and make your model learn based on irrelevant features.

Benefits of Feature selection:

- Reduces Overfitting:
- Less redundant data means less opportunity to make decisions based on noise.
- Improves Accuracy:
- Less misleading data means modelling accuracy improves.
- Reduces Training Time:
- Fewer data points reduce algorithm complexity and algorithms train faster.

Variance threshold:

If the variance is low or close to zero, then a feature is approximately constant and will not improve the performance of the model. Hence, we remove such features from our model using the scaled data for threshold calculations

	variables	VIF
0	Temperature	81.535591
1	Dew Point	22.166141
2	Humidity	7.629838
3	Wind Speed	5.241164
4	Pressure	84.730990
5	FLT_SCH_DEPATURE	12.207621
6	FLT_SCH_ARRIVAL	13.282434
7	TAXI_OUT	10.980042
8	speed	13.466932

From above scaled features there is a high variation inflation factor on the temperature , pressure and Flt_Sch_Depature, so we removed the insignificant variable.

After removing , high Multicollinearity feature from the dataset. The Multicollinearity in the dataset has been lowered.

	variables	VIF
0	Dew Point	6.804386
1	Humidity	6.594255
2	Wind Speed	4.482867
3	FLT_SCH_ARRIVAL	9.525968
4	TAXI_OUT	9.021286
5	speed	9.729450

Removing Unnecessary Columns:

- Features like Humidity, Wind Speed ,Dep_Delay_Ontime are found to be a not significant
- Strong Multicollinearity is found

Dew Point	-0.0432	0.017	-2.583	0.010	-0.076	-0.010
Humidity	0.0038	0.017	0.222	0.824	-0.030	0.037
Wind Speed	-0.0286	0.019	-1.508	0.132	-0.066	0.009
FLT_SCH_ARRIVAL	-0.2122	0.023	-9.082	0.000	-0.258	-0.166
speed	0.0613	0.023	2.723	0.006	0.017	0.105
DAY_OF_WEEK	-0.0239	0.008	-3.094	0.002	-0.039	-0.009
CARRIER_CODE	-0.0257	0.008	-3.127	0.002	-0.042	-0.010
DEST	-0.0060	0.001	-6.634	0.000	-0.008	-0.004
Wind	0.0695	0.011	6.434	0.000	0.048	0.091
Condition	-0.0126	0.002	-5.966	0.000	-0.017	-0.008
DEP_DELAY_EARLY	0.3031	0.036	8.501	0.000	0.233	0.373
DEP_DELAY_ONTIME	-0.0672	0.072	-0.934	0.350	-0.208	0.074

- The pval of columns are greater then 0.05 so we are removing the insignificant variables.

5.4.MODEL BUILDING:

5.4.1.LR MODEL:

Linear regression attempts to model the relationship between two variables by fitting a linear equation to observed data.

5.4.2.LASSO:

Lasso is to obtain the subset of predictors that minimizes prediction error for a quantitative response variable. The lasso does this by imposing a constraint on the model parameters that causes regression coefficients for some variables to shrink toward zero.

5.4.3.RIDGE:

Ridge regression is a model tuning method that is used to analyse any data that suffers from multicollinearity. This method performs L2 regularization. When the issue of multicollinearity occurs, least-squares are unbiased, and variances are large, this results in predicted values to be far away from the actual values

5.4.4.ELASTIC:

the fitting of linear or logistic regression models, the elastic net is a regularized regression method that linearly combines the L_1 and L_2 penalties of the lasso and ridge methods.

5.4.5.LIGHT GBM:

Light GBM is a fast, distributed, high-performance gradient boosting framework based on decision tree algorithm, used for ranking, classification and many other machine learning tasks

5.4.5.SVR:

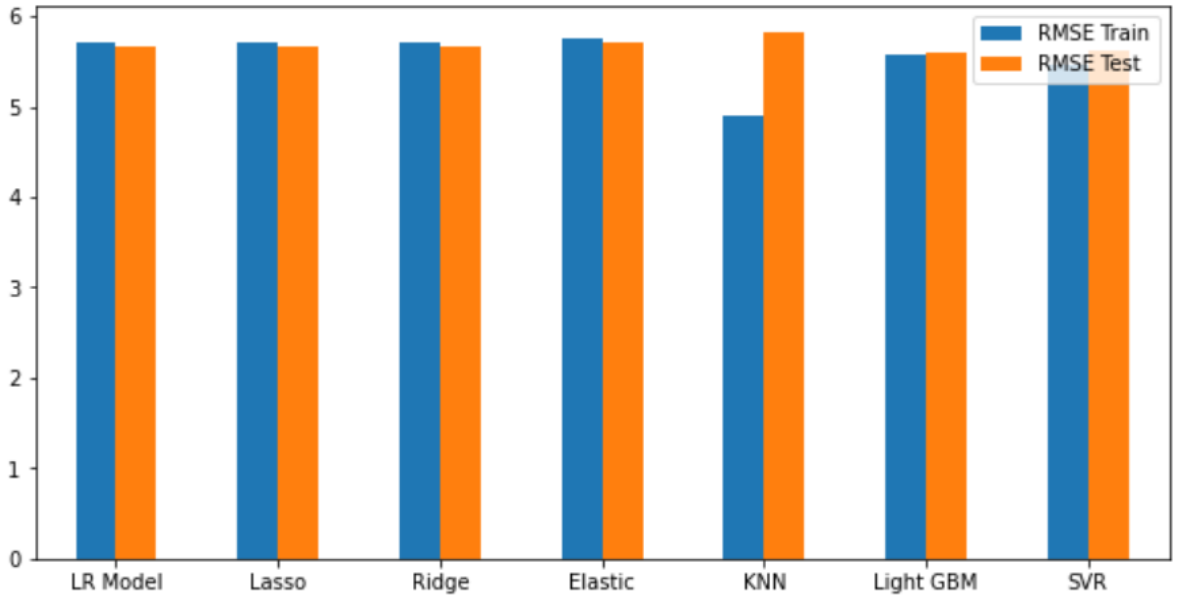
SVR is an regression algorithm, so we can use SVR for working with continuous values instead of Classification which is SVM. Kernel: The function used to map a lower dimensional data into a higher dimensional data. ... The support vectors can be on the Boundary lines or outside it.

To evaluate the model, we use the metrics R2 score, Root Mean Squared Error and Mean Absolute Percentage Error. Applying regression on the Train and Test we get the following results.

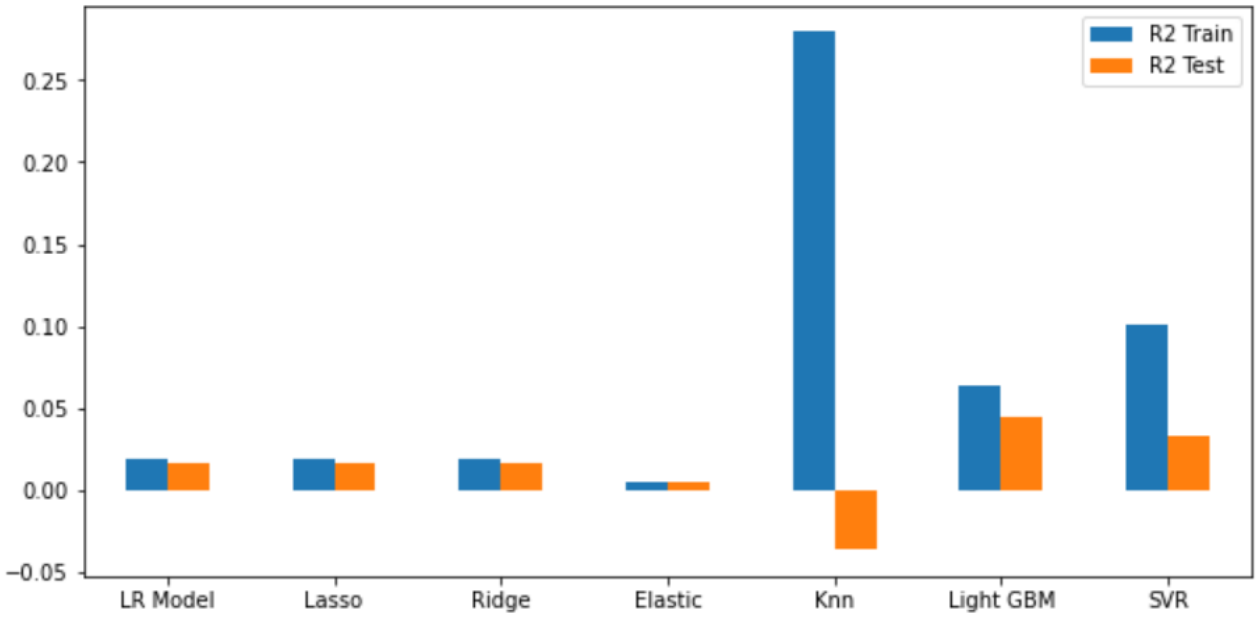
:

	LR Model	Lasso	Ridge	Elastic	Light GBM	SVR
R2 Train	0.019105	0.019104	0.018888	0.005519	0.063575	0.100461
R2 Test	0.016587	0.016619	0.016804	0.005535	0.044610	0.033664
RMSE Train	5.705372	5.705374	5.706002	5.744746	5.574541	5.463648
RMSE Test	5.672939	5.672848	5.672313	5.704729	5.591528	5.623470

RMSE TRAIN AND RMSE TEST ERROR VALUES:



R2 TRAIN AND R2 TEST ERROR VALUES:



5.5.INFERENCE:

From the above models performed we found the model didn't perform well and minimal poor values are found. Usually the R2 train and R2 test should have more values comparing the obtain values which are given below and RMSE train ,RMSE test error values must be less but our datset model had poor values in R2 train and test which less than RMSE train and split. Compared to other models Light gbm and SVR seemed to have high scores after hyper paramet tuning but still it didn't satisfy the minimum scores required in R2 train and R2test.

	LR Model	Lasso	Ridge	Elastic	Light GBM	SVR
R2 Train	0.019105	0.019104	0.018888	0.005519	0.063575	0.100461
R2 Test	0.016587	0.016619	0.016804	0.005535	0.044610	0.033664
RMSE Train	5.705372	5.705374	5.706002	5.744746	5.574541	5.463648
RMSE Test	5.672939	5.672848	5.672313	5.704729	5.591528	5.623470

6.Challenges & limitation:

Choosing the right Encoding method among label encoding and one hot encoding was challenging. Certain variable were encoded using label encoding and else with one-hot encoding.

Earlier the Target variable was continuous, Trying out different regression model , It was understandable that Regression models were not suitable. The scores of R2 and RMSE were very poor. So, we switched to Classification by labeling the Target variable as 0 and 1 using bins. Classification models were arrived with good Accuracy.

7.SUPERVISED LEARNING CLASSIFICATION APPROACH:

7.1.LOGISTIC REGRESSION RESULTS:

Dep. Variable:	TAXI_OUT	No. Observations:	20172
Model:	Logit	Df Residuals:	20144
Method:	MLE	Df Model:	27
Date:	Thu, 09 Sep 2021	Pseudo R-squ.:	0.02623
Time:	19:57:08	Log-Likelihood:	-12624.
converged:	True	LL-Null:	-12964.
Covariance Type:	nonrobust	LLR p-value:	5.971e-126

	coef	std err	z	P>|z|	[0.025	0.975]
const	0.4653	0.108	4.316	0.000	0.254	0.677
Dew Point	-0.0542	0.017	-3.250	0.001	-0.087	-0.022
Humidity	-0.0051	0.017	-0.299	0.765	-0.039	0.028
Wind Speed	-0.0254	0.019	-1.347	0.178	-0.062	0.012
FLT_SCH_ARRIVAL	-0.2254	0.023	-9.722	0.000	-0.271	-0.180
speed	0.0309	0.022	1.406	0.160	-0.012	0.074
DAY_OF_WEEK	-0.0249	0.008	-3.236	0.001	-0.040	-0.010
CARRIER_CODE	-0.0245	0.008	-2.974	0.003	-0.041	-0.008
Wind	0.0709	0.011	6.566	0.000	0.050	0.092
Condition	-0.0126	0.002	-5.947	0.000	-0.017	-0.008
DEP_DELAY_EARLY	0.3168	0.036	8.886	0.000	0.247	0.387
DEP_DELAY_ONTIME	0.0102	0.072	0.142	0.887	-0.131	0.151
SCHEDULED_DEPARTURE_Morning	0.2600	0.210	1.240	0.215	-0.151	0.671
SCHEDULED_DEPARTURE_Night	0.0092	0.127	0.072	0.942	-0.239	0.258
SCHEDULED_DEPARTURE_Noon	0.2869	0.133	2.162	0.031	0.027	0.547
ACTUAL_DEPATURE_Midnight	1.2990	0.325	3.997	0.000	0.662	1.936
ACTUAL_DEPATURE_Morning	-0.4325	0.204	-2.120	0.034	-0.832	-0.033
ACTUAL_DEPATURE_Night	0.2336	0.121	1.932	0.053	-0.003	0.471
ACTUAL_DEPATURE_Noon	0.1425	0.128	1.117	0.264	-0.108	0.393
SCHEDULED_ARRIVAL_Midnight	-0.0954	0.116	-0.820	0.412	-0.323	0.133
SCHEDULED_ARRIVAL_Morning	0.0278	0.079	0.354	0.723	-0.126	0.182
SCHEDULED_ARRIVAL_Night	-0.2699	0.076	-3.564	0.000	-0.418	-0.121
SCHEDULED_ARRIVAL_Noon	-0.0819	0.063	-1.305	0.192	-0.205	0.041
SCHEDULED_ARRIVAL_Noon	-0.0819	0.063	-1.305	0.192	-0.205	0.041
Wind Gust_strong Gust	-0.1669	0.078	-2.145	0.032	-0.319	-0.014
Wind Gust_violent Gust	-0.4992	0.051	-9.820	0.000	-0.599	-0.400
Region_North East	0.2548	0.065	3.893	0.000	0.127	0.383
Region_South	0.2702	0.057	4.741	0.000	0.159	0.382
Region_West	0.0126	0.061	0.207	0.836	-0.106	0.132

7.2.MULTICOLLINEARITY TREATMENT:

Multicollinearity is a situation where two or more predictors are highly linearly related. In general, an absolute correlation coefficient of >0.7 among two or more predictors indicates the presence of multicollinearity.

VIF method is used to treat the MULTICOLLINEARITY for numerical features

Initially,

	variables	VIF
0	Temperature	81.536018
1	Dew Point	22.166601
2	Humidity	7.629428
3	Wind Speed	5.243004
4	Pressure	84.731560
5	FLT_SCH_DEPATURE	12.209776
6	FLT_SCH_ARRIVAL	13.281467
7	TAXI_OUT	10.985133
8	speed	13.466947

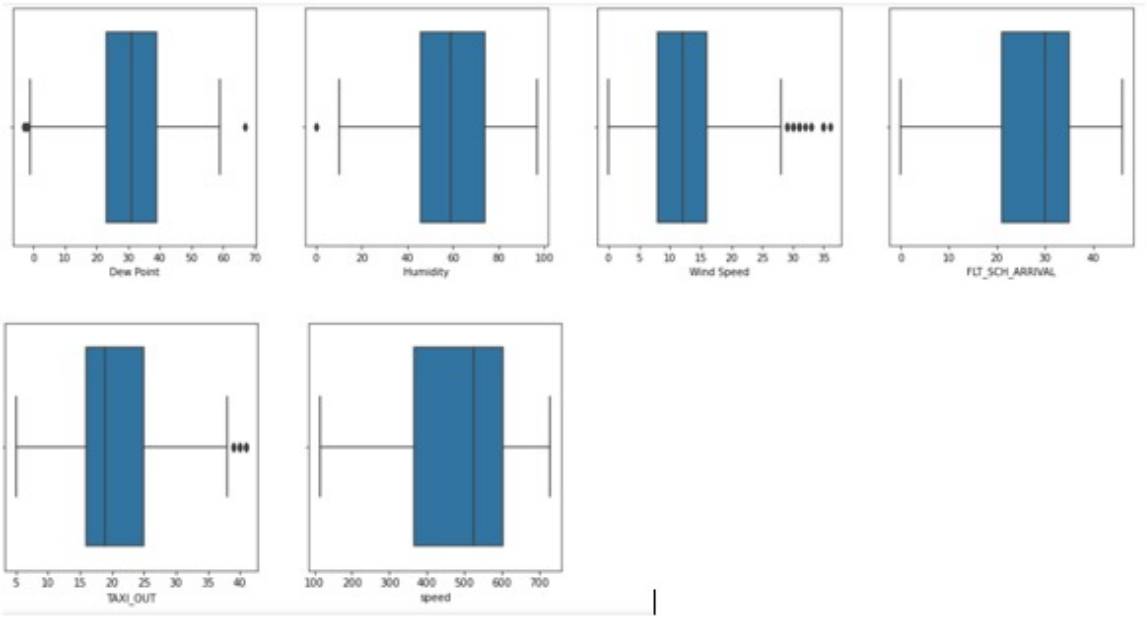
Features like Pressure , Temperature and FLT_SCH_DEPARTURE are removed as their VIF values are seen to be greater than 10.

	variables	VIF
0	Dew Point	6.806284
1	Humidity	6.593929
2	Wind Speed	4.484775
3	FLT_SCH_ARRIVAL	9.525327
4	TAXI_OUT	9.026111
5	speed	9.729297

Multicollinearity is removed

7.3.OUTLIER TREATMENT:

Before Outlier Treatment:



Winsorrize method is used for treating Outliers

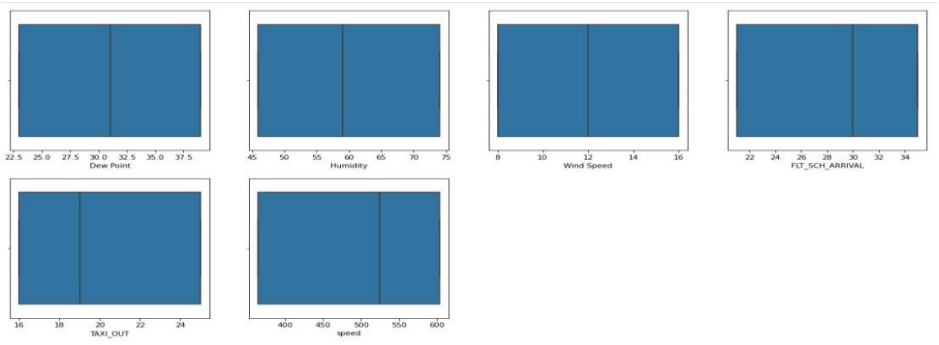
```
from scipy import stats
```

```
def capping(df,columns,lower_end,upper_end):
    for i in columns:
        stats.mstats.winsorize(a=df[i],limits=(lower_end,upper_end),inplace=True)
```

```
capping(df,df_num.columns,0.25,0.25)
```

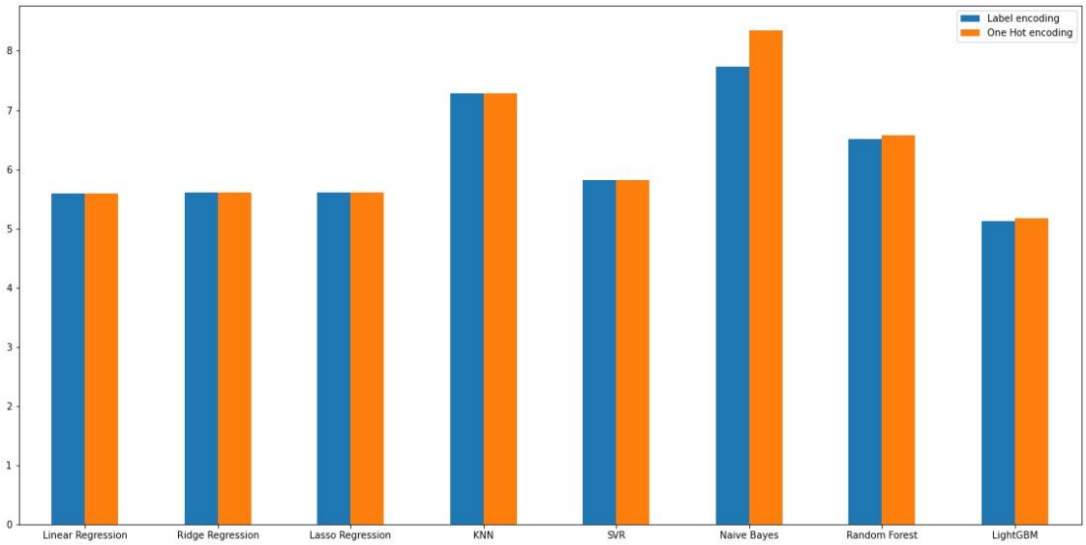
```
k = 1
plt.figure(figsize =(20,20))
for i in df_num.columns:
    if(df[i].dtypes != 'object'):
        plt.subplot(4,4,k)
        sns.boxplot(x=df[i])
        k+=1
```

After Outlier Treatment:



7.4.ENCODING PROCESS:

Label Encoding and One hot encoding are performed for the categorical variables



- Label encoding is better in predicting taxi_out time in almost all algorithms.
- One hot encoding produces exceptionally high RMSE in Naive Bayes algorithm.
- On the basis of RMSE we can say that LightGBM model is best model in both Label encoded as well as One Hot Encoded data.

7.5.SCALING:

In scaling the numerical columns here we use the Standard Scaler. StandardScaler is that it will transform your data such that its distribution will have a mean value 0 and standard deviation of 1.Then we are checking the p-values of the features.If the pvalue is greater then 0.05 then we remove the insignificant variable .

```
1 from sklearn.preprocessing import StandardScaler
```

```
1 ss = StandardScaler()
2 num[['Dew Point','Humidity','Wind Speed','FLT_SCH_ARRIVAL','speed']]=ss.fit_transform(num[['Dew Point','Humidity','Wind Spee
3 num_scaled = num
4 num_scaled.head(3)
```

	Dew Point	Humidity	Wind Speed	FLT_SCH_ARRIVAL	speed
0	0.471137	-0.146852	1.238438	-1.265467	-0.014755
1	0.471137	-0.146852	1.238438	-1.265467	1.094672
2	0.471137	-0.146852	1.238438	-1.265467	0.756784

7.6.NORMALITY CHECK:

Shapiro is performed to check the normality of the data

```
1 for i in x.columns:
2     print(i,shapiro(x[i]))
```

```
Dew Point ShapiroResult(statistic=0.8410568833351135, pvalue=0.0)
Humidity ShapiroResult(statistic=0.8403312563896179, pvalue=0.0)
Wind Speed ShapiroResult(statistic=0.8114496469497681, pvalue=0.0)
FLT_SCH_ARRIVAL ShapiroResult(statistic=0.8012262582778931, pvalue=0.0)
speed ShapiroResult(statistic=0.8185033202171326, pvalue=0.0)
DAY_OF_WEEK ShapiroResult(statistic=0.9202751517295837, pvalue=0.0)
CARRIER_CODE ShapiroResult(statistic=0.8928393721580505, pvalue=0.0)
Wind ShapiroResult(statistic=0.8543143272399902, pvalue=0.0)
Condition ShapiroResult(statistic=0.801169216632843, pvalue=0.0)
DEP_DELAY_EARLY ShapiroResult(statistic=0.5885580778121948, pvalue=0.0)
DEP_DELAY_ONTIME ShapiroResult(statistic=0.22644412517547607, pvalue=0.0)
SCHEDULED_DEPARTURE_Morning ShapiroResult(statistic=0.6258369088172913, pvalue=0.0)
SCHEDULED_DEPARTURE_Night ShapiroResult(statistic=0.3886132836341858, pvalue=0.0)
SCHEDULED_DEPARTURE_Noon ShapiroResult(statistic=0.5675216913223267, pvalue=0.0)
ACTUAL_DEPATURE_Midnight ShapiroResult(statistic=0.041541874408721924, pvalue=0.0)
ACTUAL_DEPATURE_Morning ShapiroResult(statistic=0.6247833967208862, pvalue=0.0)
ACTUAL_DEPATURE_Night ShapiroResult(statistic=0.3997676968574524, pvalue=0.0)
ACTUAL_DEPATURE_Noon ShapiroResult(statistic=0.5617985725402832, pvalue=0.0)
SCHEDULED_ARRIVAL_Midnight ShapiroResult(statistic=0.23276418447494507, pvalue=0.0)
SCHEDULED_ARRIVAL_Morning ShapiroResult(statistic=0.5528894662857056, pvalue=0.0)
SCHEDULED_ARRIVAL_Night ShapiroResult(statistic=0.5351464748382568, pvalue=0.0)
SCHEDULED_ARRIVAL_Noon ShapiroResult(statistic=0.5418931841850281, pvalue=0.0)
Wind Gust_strong Gust ShapiroResult(statistic=0.19938212633132935, pvalue=0.0)
Wind Gust_violent Gust ShapiroResult(statistic=0.41524630784988403, pvalue=0.0)
Region_North East ShapiroResult(statistic=0.408480703830719, pvalue=0.0)
Region_South ShapiroResult(statistic=0.5999220609664917, pvalue=0.0)
Region_West ShapiroResult(statistic=0.6274762153625488, pvalue=0.0)
```

7.7.NULL VALUE TREATMENT:

No null values detected

```
]:  1 x.isnull().sum()
```

```
]: Dew Point                      0
   Humidity                       0
   Wind Speed                     0
   FLT_SCH_ARRIVAL                0
   speed                          0
   DAY_OF_WEEK                    0
   CARRIER_CODE                   0
   Wind                           0
   Condition                      0
   DEP_DELAY_EARLY                0
   DEP_DELAY_ONTIME               0
   SCHEDULED_DEPARTURE_Morning    0
   SCHEDULED_DEPARTURE_Night      0
   SCHEDULED_DEPARTURE_Noon       0
   ACTUAL_DEPATURE_Midnight       0
   ACTUAL_DEPATURE_Morning        0
   ACTUAL_DEPATURE_Night          0
   ACTUAL_DEPATURE_Noon           0
   SCHEDULED_ARRIVAL_Midnight     0
   SCHEDULED_ARRIVAL_Morning      0
```

8.BASE MODEL BUILDING AND MODEL EVALUATION

Train Test Split:

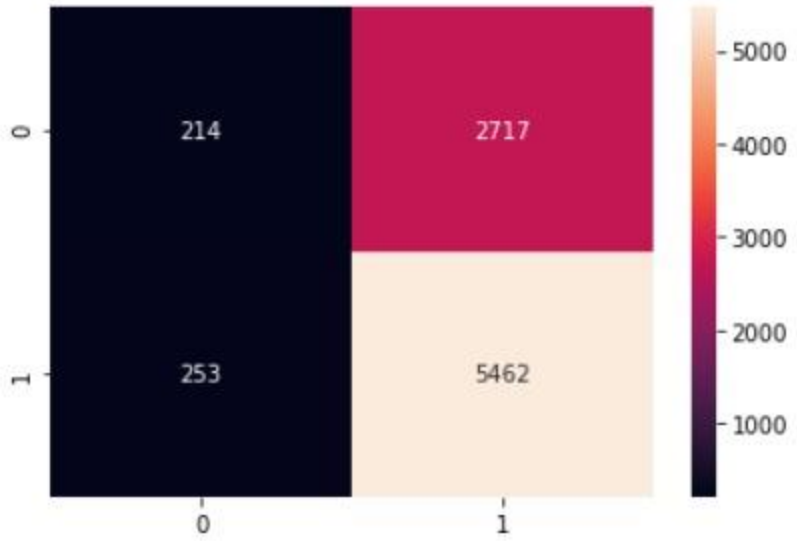
```
1 x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=45)
2 x_train.shape,x_test.shape,y_train.shape,y_test.shape
```

```
((20172, 23), (8646, 23), (20172,), (8646,))
```

The data is split into dependent feature Y (target) and independent features X. The data is then split into training and testing sets in order to avoid data leakage. The default 70:30 split is done.

8.1.LOGISTIC REGRESSION:

The basic Logistic Regression Model has been built as the Target Variable now falls under the Binary Classification. Let us identify the False Positive, False Negative, True Positive and True Negative:



TRAIN REPORT:

```
from sklearn.metrics import classification_report
print(classification_report(y_train,y_train_pred))
```

	precision	recall	f1-score	support
0	0.54	0.08	0.14	6908
1	0.67	0.96	0.79	13264
accuracy			0.66	20172
macro avg	0.60	0.52	0.46	20172
weighted avg	0.62	0.66	0.57	20172

TEST REPORT:

```
from sklearn.metrics import classification_report
print(classification_report(y_test,y_test_pred))
```

	precision	recall	f1-score	support
0	0.46	0.07	0.13	2931
1	0.67	0.96	0.79	5715
accuracy			0.66	8646
macro avg	0.56	0.51	0.46	8646
weighted avg	0.60	0.66	0.56	8646

INFERENCE:

Imbalance in data is found,therefore SMOTE test need to be performed

8.2.SMOTE:

SMOTE (synthetic minority oversampling technique) is one of the most commonly used oversampling methods to solve the imbalance problem. It aims to balance class distribution by randomly increasing minority class examples by replicating them.

```
1 from imblearn.over_sampling import SMOTE
```

```
1 smote = SMOTE(sampling_strategy=1,random_state=10)
```

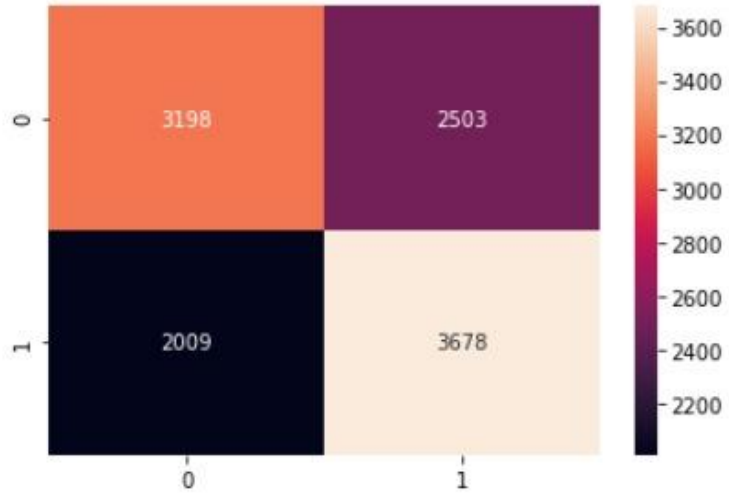
```
1 x_smote,y_smote = smote.fit_resample(x,y)
2 x_sm = pd.DataFrame(x_smote, columns=x.columns)
3 y_sm = pd.DataFrame(y_smote, columns=['TAXI_OUT'])
```

```
1 y_sm.value_counts()
```

```
TAXI_OUT
1           18979
0           18979
dtype: int64
```

8.3.LOGISTIC REGRESSION AFTER SMOTE:

CONFUSION MATRIX:



TRAIN REPORT:

```
: 1 from sklearn.metrics import classification_report
  2 print(classification_report(y_train,y_train_pred))
```

	precision	recall	f1-score	support
0	0.62	0.57	0.59	13278
1	0.60	0.65	0.62	13292
accuracy			0.61	26570
macro avg	0.61	0.61	0.61	26570
weighted avg	0.61	0.61	0.61	26570

TEST REPORT:

```
1 from sklearn.metrics import classification_report
2 print(classification_report(y_test,y_test_pred))
```

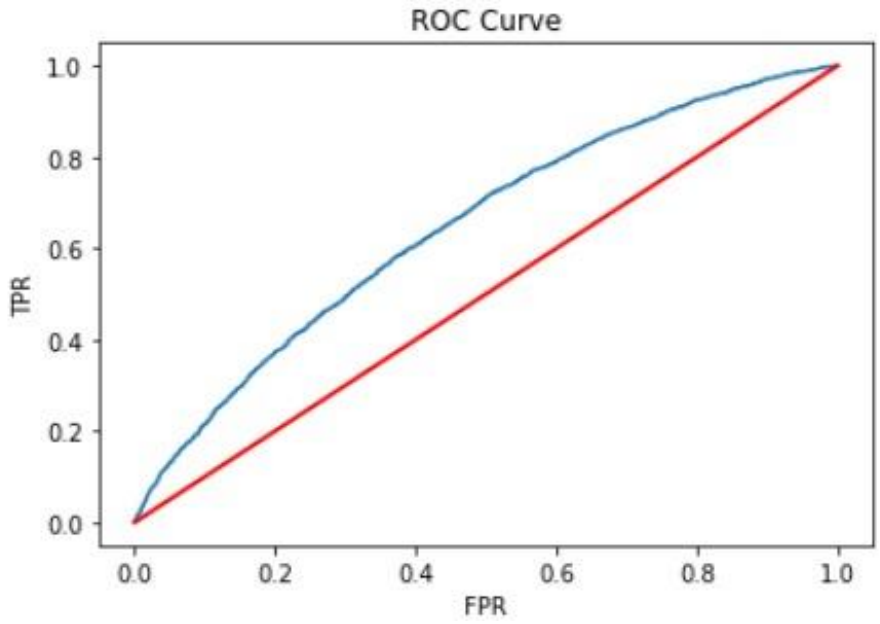
	precision	recall	f1-score	support
0	0.61	0.56	0.59	5701
1	0.60	0.65	0.62	5687
accuracy			0.60	11388
macro avg	0.60	0.60	0.60	11388
weighted avg	0.60	0.60	0.60	11388

BIASED ERROR AND VARIANCE ERROR:

```
: 1  from sklearn.model_selection import KFold
  2  from sklearn.model_selection import cross_val_score
  3  k = KFold(n_splits = 5, shuffle = True, random_state = 48)
  4  scores_smote = cross_val_score(log_reg,x.astype('float'),y.astype('float'),cv=k,scoring = 'f1_weighted')
  5  print(scores_smote)
  6  log_bias_smote = 1-np.mean(scores_smote)
  7  log_var_smote = np.std(scores)/np.mean(scores_smote)
  8  print('Scores:',scores)
  9  print('Bias error:',1-np.mean(scores_smote))
 10  print('Variance error:', np.std(scores_smote)/np.mean(scores_smote))
```

```
[0.61022381 0.60380451 0.61081594 0.60280804 0.60310972]
Scores: [0.59721812 0.58318314 0.58871726 0.59655967 0.59524004]
Bias error: 0.3938475955967362
Variance error: 0.005915229349366793
```

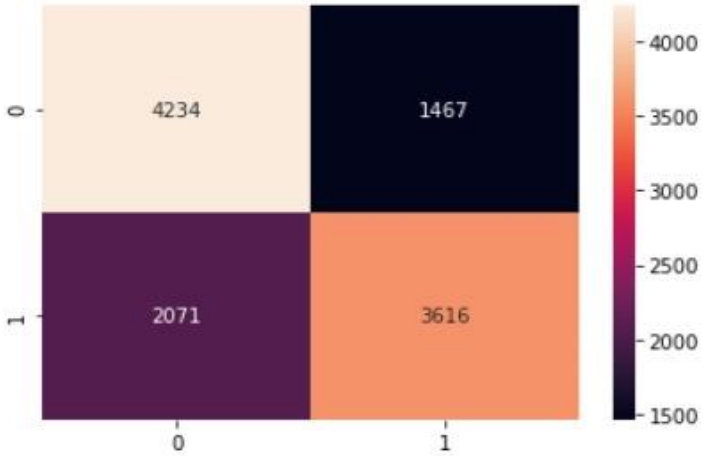
ROC CURVE:



8.4.KNN:

- KNN algorithms use data and classify new data points based on similarity measures (e.g. distance function).
- Classification is done by a majority vote to its neighbors. The data is assigned to the class which has the nearest neighbors.
- As we increase the number of nearest neighbors, the value of k, accuracy might increase.

CONFUSION MATRIX:



CLASSIFICATION REPORT:

```
1 print(classification_report(y_train,y_train_pred))
```

```
              precision    recall  f1-score   support

           0       0.78      0.85      0.81     13278
           1       0.83      0.76      0.79     13292

    accuracy                           0.80     26570
   macro avg       0.80      0.80      0.80     26570
weighted avg       0.80      0.80      0.80     26570
```

```
1 print(classification_report(y_test,y_test_pred))
```

```
              precision    recall  f1-score   support

           0       0.67      0.74      0.71      5701
           1       0.71      0.64      0.67      5687

    accuracy                           0.69     11388
   macro avg       0.69      0.69      0.69     11388
weighted avg       0.69      0.69      0.69     11388
```

Train accuracy = 0.8013549115543847

TEST accuracy = 0.6893220934316825

BIASED AND VARIANCE ERROR:

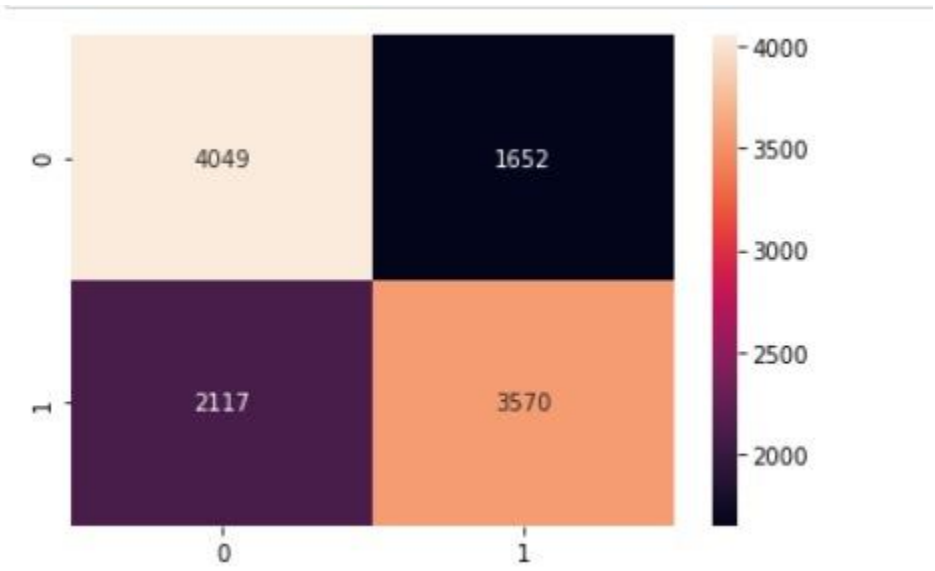
```
1 k = KFold(n_splits = 5, shuffle = True, random_state = 48)
2 scores = cross_val_score(knn,x,y,cv=k,scoring = 'f1_weighted')
3 knn_bias = 1-np.mean(scores)
4 knn_var = np.std(scores)/np.mean(scores)
5 print('Scores:',scores)
6 print('Bias error:',1-np.mean(scores))
7 print('Variance error:', np.std(scores)/np.mean(scores))
```

```
Scores: [0.69572631 0.69456884 0.69928444 0.70213834 0.70374934]
Bias error: 0.30090654535759787
Variance error: 0.005069231202062406
```

8.5.Decision Tree Classifier:

Decision tree builds 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.

CONFUSION MATRIX:



CLASSIFICATION REPORT:

```
: 1 print(classification_report(y_train,y_train_pred))
```

	precision	recall	f1-score	support
0	0.91	0.98	0.94	13278
1	0.97	0.91	0.94	13292
accuracy			0.94	26570
macro avg	0.94	0.94	0.94	26570
weighted avg	0.94	0.94	0.94	26570

```
: 1 print(classification_report(y_test,y_test_pred))
```

	precision	recall	f1-score	support
0	0.66	0.71	0.68	5701
1	0.68	0.63	0.65	5687
accuracy			0.67	11388
macro avg	0.67	0.67	0.67	11388
weighted avg	0.67	0.67	0.67	11388

Train accuracy = 0.9407978923598043

Test accuracy = 0.669037583421145

BIAS AND VARIANCE ERROR:

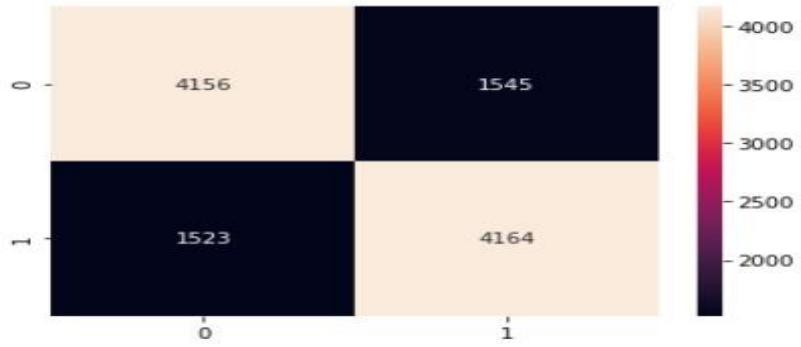
```
k = KFold(n_splits = 5, shuffle = True, random_state = 48)
scores = cross_val_score(dt,x,y,cv=k,scoring = 'f1_weighted')
dt_bias = 1-np.mean(scores)
dt_var = np.std(scores)/np.mean(scores)
print('Scores:',scores)
print('Bias error:',1-np.mean(scores))
print('Variance error:', np.std(scores)/np.mean(scores))
```

```
Scores: [0.68736945 0.68700316 0.69022061 0.69114404 0.6879555 ]
Bias error: 0.31126144730543603
Variance error: 0.002384179624058304
```

8.6.RANDOM FOREST CLASSIFIER:

Random forest belongs to supervised learning method algorithm used for classification that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes or means prediction of the individual trees. The decision tree is a tree structure (which can be a binary tree or a non-binary tree). Each of its non-leaf nodes corresponds to a test of a feature, each branch representing the output of the feature attribute over a range of values, and each leaf node storing a category. The decision tree starts with a root node, tests the corresponding feature attributes in the category to be classified, and output branches are selected according to their values until the leaf node is reached, finally the category stored by the leaf node is regards as the decision result. A random forest is a collection of decision trees in which each decision tree is unrelated.

CONFUSION MATRIX:



CLASSIFICATION REPORT:

```
1 print(classification_report(y_train,y_train_pred))
```

	precision	recall	f1-score	support
0	0.94	0.95	0.94	13278
1	0.94	0.94	0.94	13292
accuracy			0.94	26570
macro avg	0.94	0.94	0.94	26570
weighted avg	0.94	0.94	0.94	26570

```
1 print(classification_report(y_test,y_test_pred))
```

	precision	recall	f1-score	support
0	0.73	0.73	0.73	5701
1	0.73	0.73	0.73	5687
accuracy			0.73	11388
macro avg	0.73	0.73	0.73	11388
weighted avg	0.73	0.73	0.73	11388

Train accuracy = 0.94

Test accuracy = 0.73

8.6.1.HYPERPARAMETER TUNING IS PERFORMED AS THE MODEL IS OVERFITTING

GRID SEARCHCV:

GridSearchCV is a library function that is a member of sklearn's model_selection package. It helps to loop through predefined hyperparameters and fit your estimator (model) on your training set. So, in the end, you can select the best parameters from the listed hyperparameter

BEST PARAMETERS :

{'criterion': 'gini',

'max_depth': 9,

'min_samples_split': 2,

'n_estimators': 200}

8.7.RANDOM FOREST CLASSIFIER USING BEST PARAMETERS

CONFUSION MATRIX:

CLASSIFICATION REPORT:

```
print(classification_report(y_train,y_train_pred))
```

	precision	recall	f1-score	support
0	0.71	0.70	0.70	13278
1	0.70	0.72	0.71	13292
accuracy			0.71	26570
macro avg	0.71	0.71	0.71	26570
weighted avg	0.71	0.71	0.71	26570

```
print(classification_report(y_test,y_test_pred))
```

	precision	recall	f1-score	support
0	0.67	0.64	0.65	5701
1	0.65	0.68	0.66	5687
accuracy			0.66	11388
macro avg	0.66	0.66	0.66	11388
weighted avg	0.66	0.66	0.66	11388

Train accuracy = 0.7088445615355664

Test accuracy = 0.6577098700386371

BIAS AND VARIANCE ERROR:

```
k = KFold(n_splits = 5, shuffle = True, random_state = 48)
scores = cross_val_score(rf,x,y,cv=k,scoring = 'f1_weighted')
rf_bias = 1-np.mean(scores)
rf_var = np.std(scores)/np.mean(scores)
print('Scores:',scores)
print('Bias error:',1-np.mean(scores))
print('Variance error:', np.std(scores)/np.mean(scores))
```

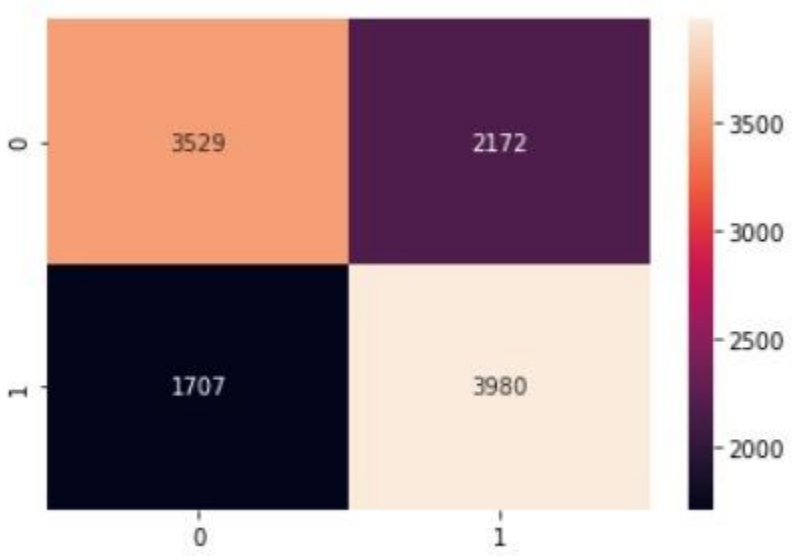
```
Scores: [0.66179148 0.66787847 0.66779985 0.66181603 0.66355703]
Bias error: 0.3354314289706004
Variance error: 0.0041323302836778816
```

8.8.GRADIENT BOOSTING:

Gradient boosting is a type of machine learning boosting. It relies on the intuition that the best possible next model, when combined with previous models, minimizes the overall prediction error. The key idea is to set the target outcomes for this next model in order to minimize the error.

This model is for regression, classification and other tasks, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. When a decision tree is a weak learner, the resulting algorithm is called gradient boosted trees, which usually outperforms random forest. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function.

CONFUSION MATRIX:



CLASSIFICATION REPORT:

```
1 print(classification_report(y_train,y_train_pred))
```

	precision	recall	f1-score	support
0	0.69	0.65	0.67	13278
1	0.67	0.71	0.69	13292
accuracy			0.68	26570
macro avg	0.68	0.68	0.68	26570
weighted avg	0.68	0.68	0.68	26570

```
1 print(classification_report(y_test,y_test_pred))
```

	precision	recall	f1-score	support
0	0.67	0.62	0.65	5701
1	0.65	0.70	0.67	5687
accuracy			0.66	11388
macro avg	0.66	0.66	0.66	11388
weighted avg	0.66	0.66	0.66	11388

Train accuracy = 0.6790365073391043

Test accuracy = 0.6593782929399368

BIAS AND VARIANCE ERROR:

```
1 k = KFold(n_splits = 5, shuffle = True, random_state = 48)
2 scores = cross_val_score(gd,x,y,cv=k,scoring = 'f1_weighted')
3 gd_bias = 1-np.mean(scores)
4 gd_var = np.std(scores)/np.mean(scores)
5 print('Scores:',scores)
6 print('Bias error:',1-np.mean(scores))
7 print('Variance error:', np.std(scores)/np.mean(scores))
```

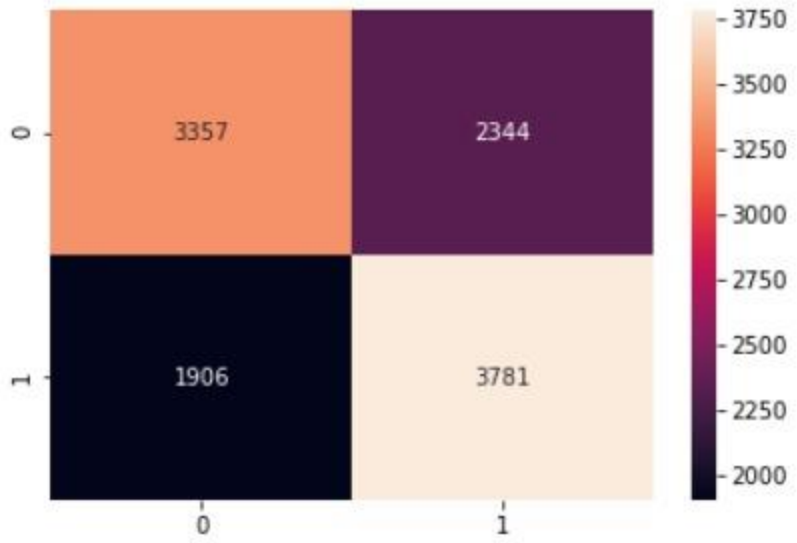
```
Scores: [0.66396658 0.66677209 0.6734615  0.6678294  0.67030942]
Bias error: 0.331532202360421
Variance error: 0.004818408696685169
```

8.9.ADA BOOST:

AdaBoost algorithm, short for Adaptive Boosting, is a Boosting technique used as an Ensemble Method in Machine Learning. It is called Adaptive Boosting as the weights are re-assigned to each instance, with higher weights assigned to incorrectly classified instances.

Boosting is used to reduce bias as well as variance for supervised learning. It works on the principle of learners growing sequentially. Except for the first, each subsequent learner is grown from previously grown learners. In simple words, weak learners are converted into strong ones. The AdaBoost algorithm works on the same principle as boosting with a slight difference.

CONFUSION MATRIX:



CLASSIFICATION REPORT:

```
1 print(classification_report(y_train,y_train_pred))
```

	precision	recall	f1-score	support
0	0.65	0.60	0.62	13278
1	0.63	0.67	0.65	13292
accuracy			0.64	26570
macro avg	0.64	0.64	0.64	26570
weighted avg	0.64	0.64	0.64	26570

```
1 print(classification_report(y_test,y_test_pred))
```

	precision	recall	f1-score	support
0	0.64	0.59	0.61	5701
1	0.62	0.66	0.64	5687
accuracy			0.63	11388
macro avg	0.63	0.63	0.63	11388
weighted avg	0.63	0.63	0.63	11388

Train accuracy = 0.6368460669928491

Test accuracy = 0.6268001404987706

BIAS AND VARIANCE ERROR:

```
1 k = KFold(n_splits = 5, shuffle = True, random_state = 48)
2 scores = cross_val_score(ada,x,y,cv=k,scoring = 'f1_weighted')
3 ada_bias = 1-np.mean(scores)
4 ada_var = np.std(scores)/np.mean(scores)
5 print('Scores:',scores)
6 print('Bias error:',1-np.mean(scores))
7 print('Variance error:', np.std(scores)/np.mean(scores))
```

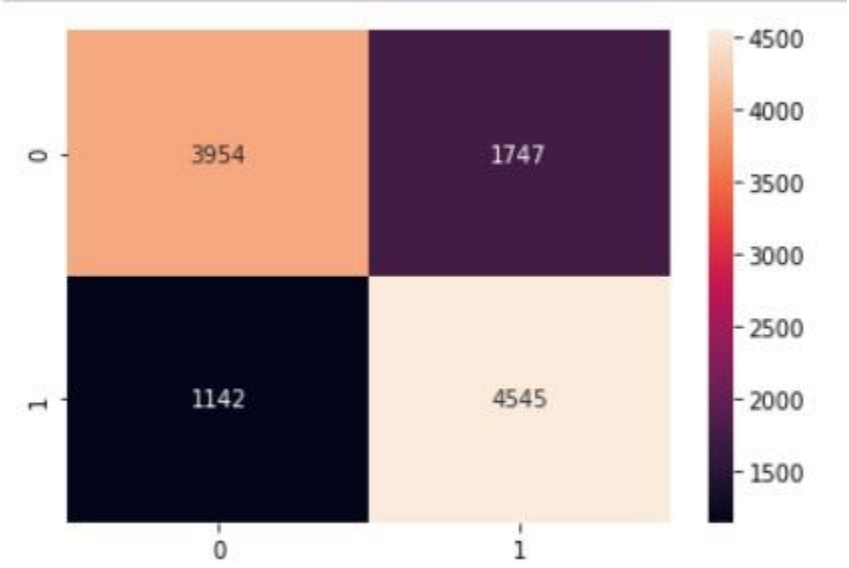
```
Scores: [0.63079182 0.62837831 0.64270299 0.63232221 0.63209299]
Bias error: 0.36674233619324215
Variance error: 0.007779041335699271
```

8.10.XGBOOST:

XGBoost is a popular and efficient open-source implementation of the gradient boosted trees algorithm. Gradient boosting is a supervised learning algorithm, which attempts to accurately predict a target variable by combining the estimates of a set of simpler, weaker models.

When using gradient boosting for regression, the weak learners are regression trees, and each regression tree maps an input data point to one of its leafs that contains a continuous score. XGBoost minimizes a regularized objective function that combines a convex loss function based on the difference between the predicted and target outputs and a penalty term for model complexity .

CONFUSION MATRIX:



CLASSIFICATION REPORT:

```
1 print(classification_report(y_train,y_train_pred))
```

	precision	recall	f1-score	support
0	0.84	0.77	0.80	13278
1	0.79	0.86	0.82	13292
accuracy			0.81	26570
macro avg	0.81	0.81	0.81	26570
weighted avg	0.81	0.81	0.81	26570

```
1 print(classification_report(y_test,y_test_pred))
```

	precision	recall	f1-score	support
0	0.78	0.69	0.73	5701
1	0.72	0.80	0.76	5687
accuracy			0.75	11388
macro avg	0.75	0.75	0.75	11388
weighted avg	0.75	0.75	0.75	11388

Train accuracy = 0.8116672939405344

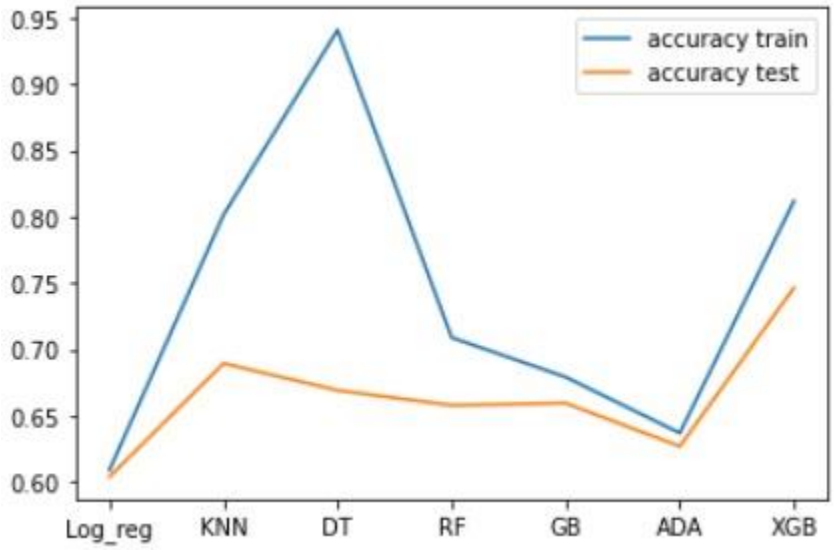
Test accuracy = 0.7463119072708114

BIAS AND VARIANCE ERROR:

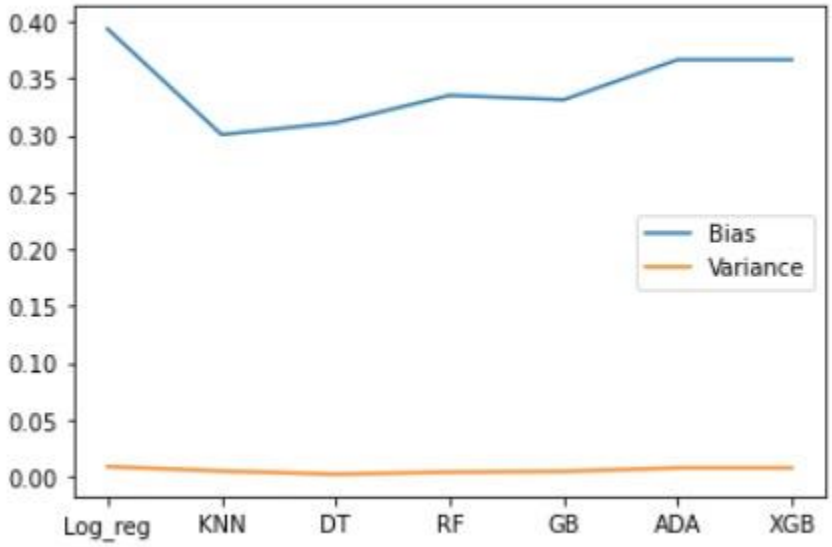
```
1 k = KFold(n_splits = 5, shuffle = True, random_state = 48)
2 scores = cross_val_score(ada,x,y,cv=k,scoring = 'f1_weighted')
3 xg_bias = 1-np.mean(scores)
4 xg_var = np.std(scores)/np.mean(scores)
5 print('Scores:',scores)
6 print('Bias error:',1-np.mean(scores))
7 print('Variance error:', np.std(scores)/np.mean(scores))
```

```
Scores: [0.63079182 0.62837831 0.64270299 0.63232221 0.63209299]
Bias error: 0.36674233619324215
Variance error: 0.007779041335699271
```

8.11.OVERALL TEST AND TRAIN ACCURACY:



8.12.OVERALL BIAS AND VARIANCE ERROR



8.13.XG Booster Hyper Parameter Tuning:

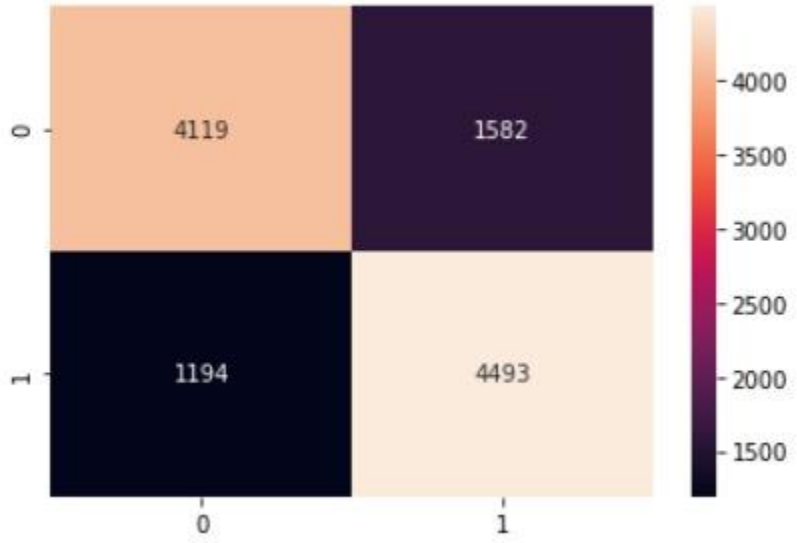
We see that XGBOOST has got better values than the other models but imbalance is found

So Hyperparameter testing is performed on XGBOOST (GRIDSEARCH CV)

BEST PARAMS USED:

{'max_depth': 7, 'min_child_weight': 1, 'n_estimators': 150}

CONFUSION MATRIX:



CLASSIFICATOIN REPORT:

TRAIN RESULT:

```
1 print(classification_report(y_train,xg_train_new))
```

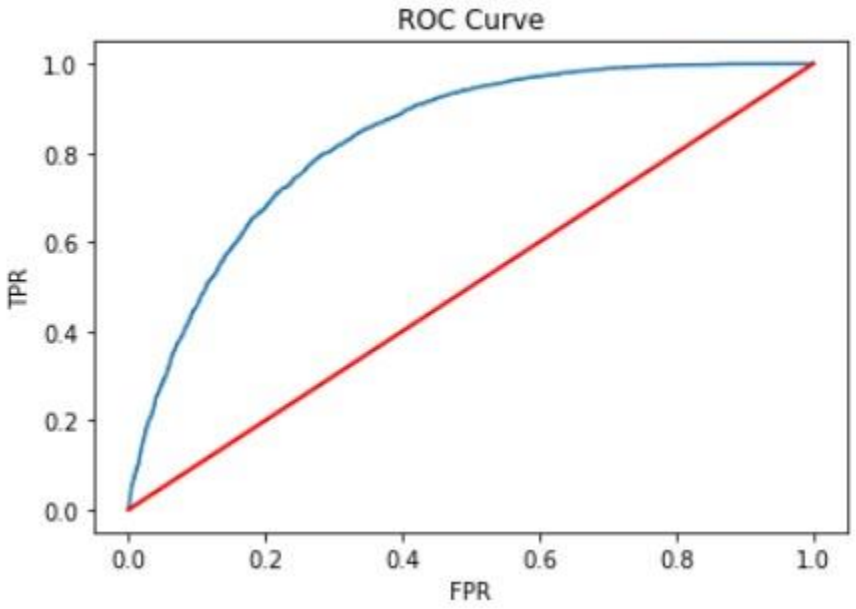
	precision	recall	f1-score	support
0	0.87	0.82	0.84	13278
1	0.83	0.88	0.85	13292
accuracy			0.85	26570
macro avg	0.85	0.85	0.85	26570
weighted avg	0.85	0.85	0.85	26570

TEST RESULT:

```
1 print(classification_report(y_test,xg_test_pred))
```

	precision	recall	f1-score	support
0	0.78	0.72	0.75	5701
1	0.74	0.79	0.76	5687
accuracy			0.76	11388
macro avg	0.76	0.76	0.76	11388
weighted avg	0.76	0.76	0.76	11388

ROC CURVE:



ROC curve score: 0.8309742826592665

9.Conclusion:

By using taxi-out runtime dataset, our prediction was to predict the taxi out was high or low for optimizing flight cost.We found the main objectives of the research with the application of data science skills such as data visualization and machine learning, It was found that different features differ in importance depending on the taxi-out runtime, and some features are not required for predicting taxi out . So we built a model to classify taxi - out is high or low.

This demonstrates that machine learning algorithms, in this case the XGBoost Classifier algorithm with tuned hyperparameters, is a good technique to build taxi - out is high or low. Model development revealed that features had different weights and different importance accordingly to the taxi - out , So we tried building a model which can satisfy all the criteria.