4/3/2022

MACHINE LEARNING PROJECT

Sushmita Kar
GREAT LEARNING

Table of Contents

Problem Statement:

You work for an office transport company. You are in discussions with ABC Consulting company for providing transport for their employees. For this purpose, you are tasked with understanding how do the employees of ABC Consulting prefer to commute presently (between home and office). Based on the parameters like age, salary, work experience etc. given in the data set 'Transport.csv', you are required to predict the preferred mode of transport. The project requires you to build several Machine Learning models and compare them so that the model can be finalised.

Data Dictionary:

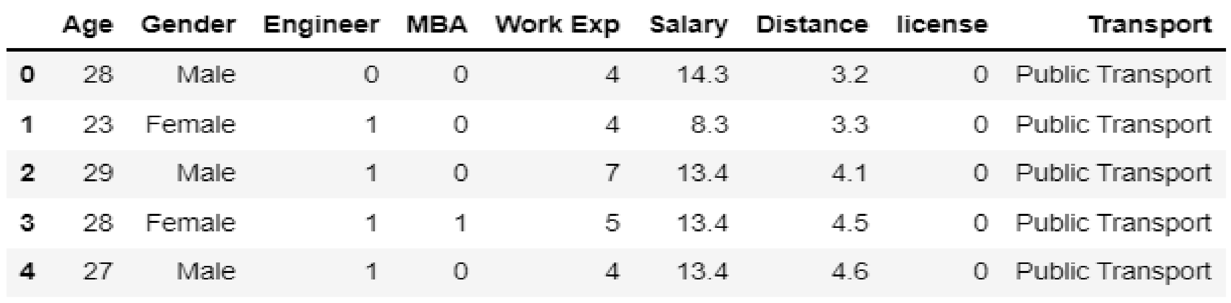
Age	Employee Age
Gender	Employee Gender Male/Female
Engineer	Employee who are Engineers Yes-1, No-0
MBA	Employee who are MBA Yes-1, No-0
Work Exp	Employee Working Experience
Salary	Employee Salary
Distance	Avg Distance travelled by Employees
license	Employees are license holder or not Yes-1, No-0
Transport	Mode of Transport-Public Transport and Private Transport

Objective: Based on the parameters like age, salary, work experience etc. given in the data set 'Transport.csv', we are required **to predict the preferred mode of transport**.

1.1) Read the dataset. Do the descriptive statistics and do null value condition check.

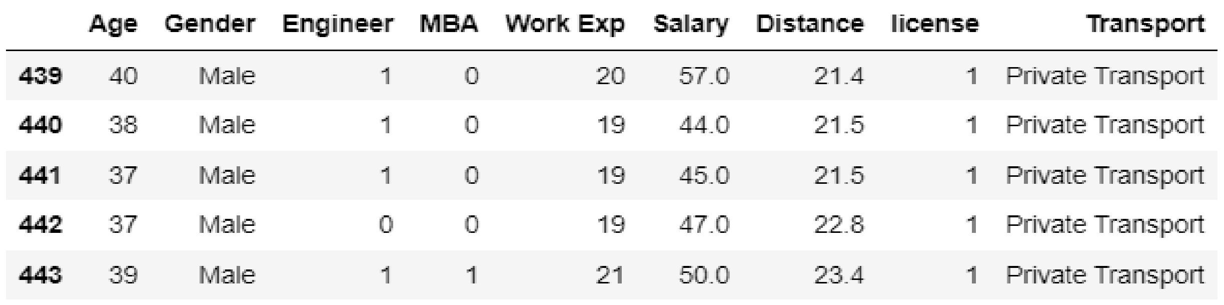
Data Description

We have out data saved in a CSV file called Transport.csv. We first read our dataset into pandas dataframe called df, and then use the head() and tail() function to show the first and last records from the dataset.



	Age	Gender	Engineer	MBA	Work Exp	Salary	Distance	license	Transport
0	28	Male	0	0	4	14.3	3.2	0	Public Transport
1	23	Female	1	0	4	8.3	3.3	0	Public Transport
2	29	Male	1	0	7	13.4	4.1	0	Public Transport
3	28	Female	1	1	5	13.4	4.5	0	Public Transport
4	27	Male	1	0	4	13.4	4.6	0	Public Transport

Table 1: First 5 records of Transport Dataset.

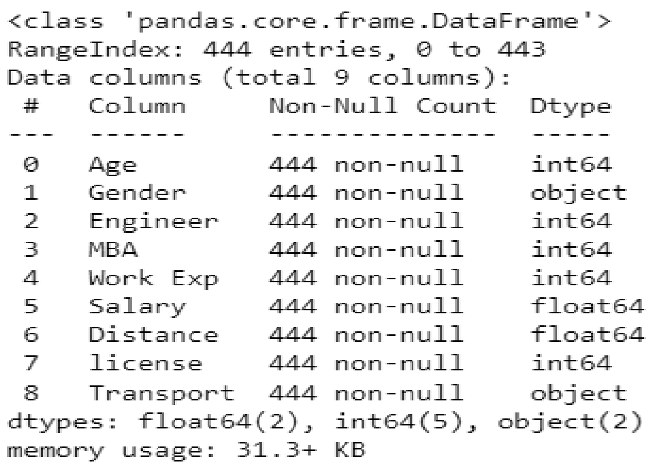


	Age	Gender	Engineer	MBA	Work Exp	Salary	Distance	license	Transport
439	40	Male	1	0	20	57.0	21.4	1	Private Transport
440	38	Male	1	0	19	44.0	21.5	1	Private Transport
441	37	Male	1	0	19	45.0	21.5	1	Private Transport
442	37	Male	0	0	19	47.0	22.8	1	Private Transport
443	39	Male	1	1	21	50.0	23.4	1	Private Transport

Table 2: Last 5 records of Transport Dataset

The following features have been provided to help us predict which mode of transport is more preferred Public or Private Transport.

Let's make sure that our data is clean (no null values, duplicates, etc)

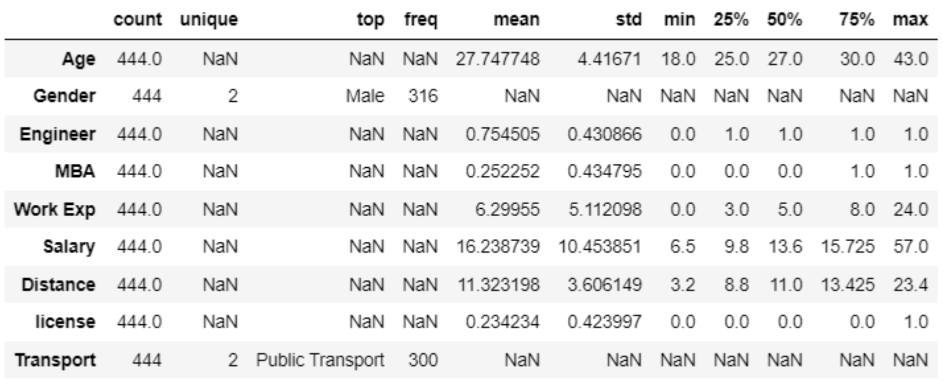


```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 444 entries, 0 to 443
Data columns (total 9 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Age        444 non-null    int64  
 1   Gender     444 non-null    object 
 2   Engineer   444 non-null    int64  
 3   MBA        444 non-null    int64  
 4   Work Exp   444 non-null    int64  
 5   Salary     444 non-null    float64
 6   Distance   444 non-null    float64
 7   license    444 non-null    int64  
 8   Transport  444 non-null    object 
dtypes: float64(2), int64(5), object(2)
memory usage: 31.3+ KB
```

- Note that there is no Null Values.
- Integer variables are Age, Engineer, MBA, Work_Exp, license.
- Object types are Gender and Transport.
- And Rest Salary and Distance are Float dtype.
- Our Target variable is 'Transport'.

As we see in our Python codebook, there are no Null Values and Duplicate records in our dataset.

Let, look at the Summary of the Dataset using describe() function. Which gives us the count, mean, std, min, 25%-percentile, 50%-percentile, 75%-percentile and max details of all the columns in the dataset.



	count	unique	top	freq	mean	std	min	25%	50%	75%	max
Age	444.0	NaN	NaN	NaN	27.747748	4.41671	18.0	25.0	27.0	30.0	43.0
Gender	444	2	Male	316	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Engineer	444.0	NaN	NaN	NaN	0.754505	0.430866	0.0	1.0	1.0	1.0	1.0
MBA	444.0	NaN	NaN	NaN	0.252252	0.434795	0.0	0.0	0.0	1.0	1.0
Work Exp	444.0	NaN	NaN	NaN	6.29955	5.112098	0.0	3.0	5.0	8.0	24.0
Salary	444.0	NaN	NaN	NaN	16.238739	10.453851	6.5	9.8	13.6	15.725	57.0
Distance	444.0	NaN	NaN	NaN	11.323198	3.606149	3.2	8.8	11.0	13.425	23.4
license	444.0	NaN	NaN	NaN	0.234234	0.423997	0.0	0.0	0.0	0.0	1.0
Transport	444	2	Public Transport	300	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Table 3: Summary of the Transport Dataset

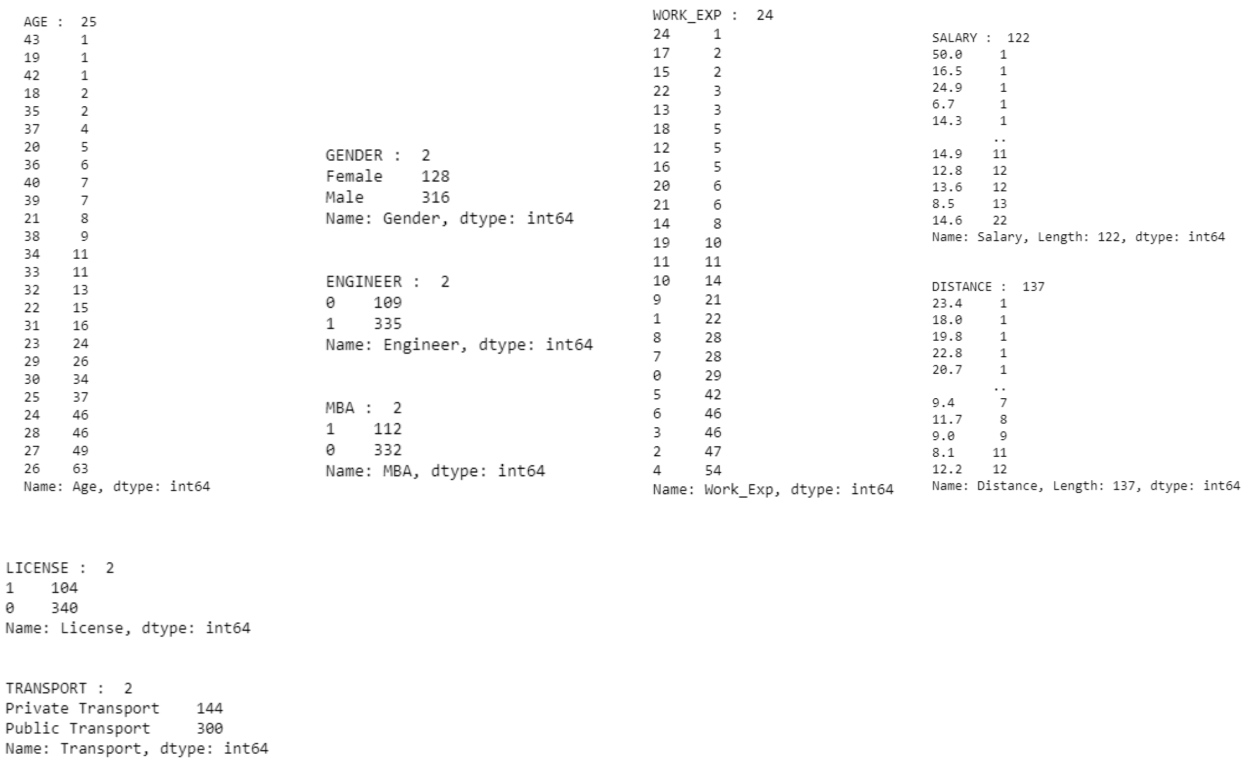
Observation:

- There are 444 rows and 9 features.
- From the Summary we can see that Age of employess are between minimum 18years to maximum 43years. The avg age of the employee is 27 years.
- There are 2 categories in Gender Male and Female. Male count is high in the dataset.
- Engineer, MBA and license column are containing inforamtion in 1 and 0.
- As per the summary employee work experience starts from 0 to 24years. Avg work exp of employee is 6.7 years.
- Salary of employee starts from 6.5 units to 57.0 units. Avg Salary of employee is 16.2 units.

- Distance covered by the employees for work purpose is from 3.0 km to 23.4km. Avg distance covered by the employee is 11.3km.
- As per Summary employees are using Public Transport more as compared to Private Transport.

Note: As we are aware that Engineer, MBA and license column has input data as '0' and '1'. To analyze the data in a proper manner we can convert these variables to object type by replacing the '0' and '1' with significant labels. But doing that will change out data while doing Encoding. Hence, we will continue with the same.

Unique count of all the variables in the dataset:



```
AGE :  25
43     1
19     1
42     1
18     2
35     2
37     4
20     5
36     6
40     7
39     7
21     8
38     9
34    11
33    11
32    13
22    15
31    16
23    24
29    26
30    34
25    37
24    46
28    46
27    49
26    63
Name: Age, dtype: int64
```

```
GENDER :  2
Female    128
Male      316
Name: Gender, dtype: int64

ENGINEER :  2
0    109
1    335
Name: Engineer, dtype: int64

MBA :  2
1    112
0    332
Name: MBA, dtype: int64
```

```
WORK_EXP :  24
24     1
17     2
15     2
22     3
13     3
18     5
12     5
16     5
20     6
21     6
14     8
19    10
11    11
10    14
9     21
1     22
8     28
7     28
0     29
5     42
6     46
3     46
2     47
4     54
Name: Work_Exp, dtype: int64
```

```
SALARY :  122
50.0     1
16.5     1
24.9     1
6.7      1
14.3     1
        ..
14.9    11
12.8    12
13.6    12
8.5     13
14.6    22
Name: Salary, Length: 122, dtype: int64

DISTANCE :  137
23.4     1
18.0     1
19.8     1
22.8     1
20.7     1
        ..
9.4      7
11.7     8
9.0      9
8.1     11
12.2    12
Name: Distance, Length: 137, dtype: int64
```

```
LICENSE :  2
1    104
0    340
Name: License, dtype: int64

TRANSPORT :  2
Private Transport    144
Public Transport     300
Name: Transport, dtype: int64
```

Observation: We see that in our dataset Fequency of Employees being Male, holding Engineer degree, non-MBA and employees not holding driving license is more. Which actually tells us that why the most preferred transport is Public Transport and not Private. Let's analyse this further more by building visual analysis and analysing through different models like Logistic Regression, Knn, Bagging and Boosting algorithms.

1.2) Perform Univariate and Bivariate Analysis. Do exploratory data analysis. Check for Outliers.

Let's first have look at the Outliers in the Data.

What is an Outliers? An outlier is an individual point of data that is distant from other points in the dataset. It is an anomaly in the dataset that may be caused by a range of errors in capturing, processing or manipulating data. Outliers can skew overall data trends, so outlier detection methods are an important part of statistics.

Why Outlier detection is important? Outlier detection is particularly important within machine learning. Models are trained on huge arrays of training data. The model understands the relationship between data points to help predict future events or categorise live data. Outliers in the training data may skew the model, lowering its accuracy and overall effectiveness.

Below is the boxplot to understand the same.

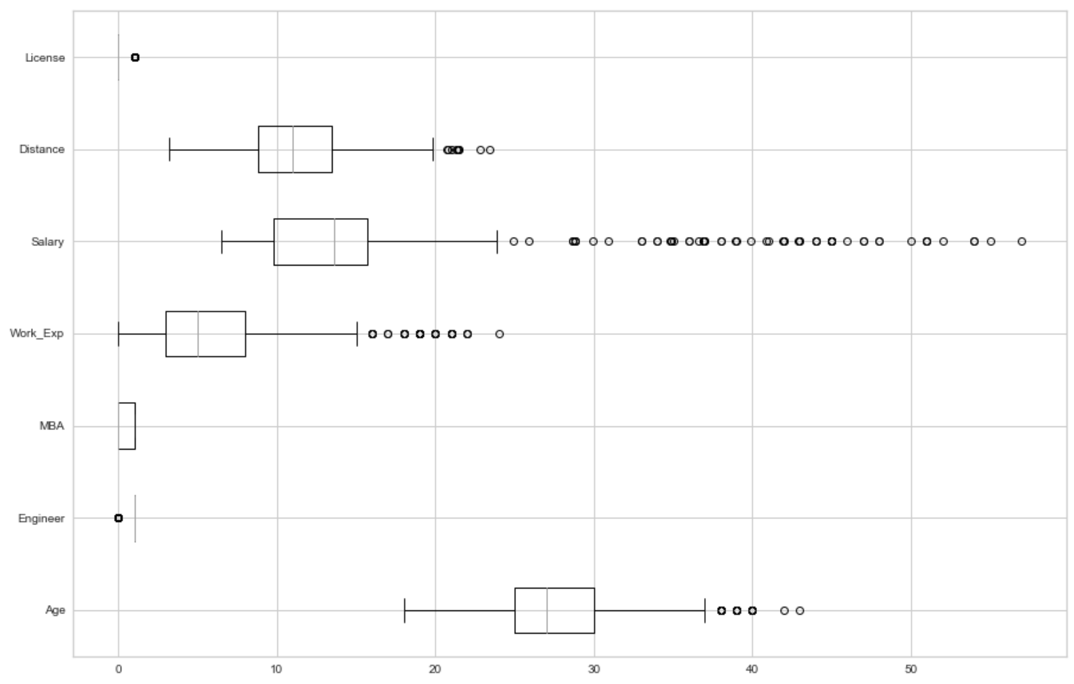
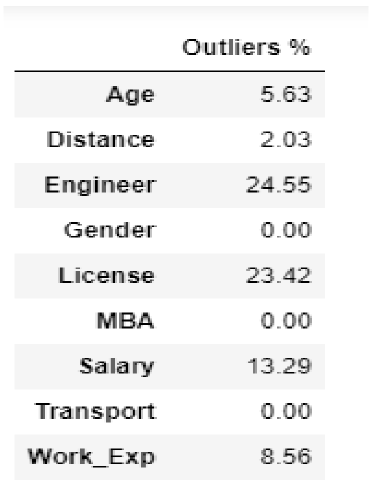
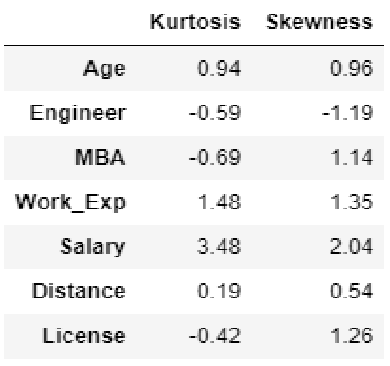


Figure 1: Boxplot to check Outliers in the dataset

Percentage of Outliers present in the dataset. And Also, Kurtosis and Skewness to understand the distribution of data in the dataset.



	Outliers %
Age	5.63
Distance	2.03
Engineer	24.55
Gender	0.00
License	23.42
MBA	0.00
Salary	13.29
Transport	0.00
Work_Exp	8.56



	Kurtosis	Skewness
Age	0.94	0.96
Engineer	-0.59	-1.19
MBA	-0.69	1.14
Work_Exp	1.48	1.35
Salary	3.48	2.04
Distance	0.19	0.54
License	-0.42	1.26

Table 4: Outlier Percentage

Table 5: Kurtosis and Skewness

We can see that there are outliers present in the variables such as ‘Age’, ‘Distance’, ‘Salary’ and ‘Work_Exp’. There are many common causes of Outliers in the dataset, data entry errors, measurement errors, experimental errors, Intentional, Natural outliers, etc.

- In our case study, these outliers are natural outliers. Company has older employees whose age is higher than 40 which is very much in age bracket. Can’t treat them as the age is showing relevant age of employee.
- In Salary column also employees with more years of work experience are going to have higher salary bracket. Hence, again these are relevant data and removing them will mean that we will be compromising with the date of employees.
- Work_exp do show outliers but they are relevant no of years of working experience recorded and it is linked to salary of employee too. Hence, treating them again means compromising in relevant data hence, wont treat any outliers in this dataset.

Hence, we are not going to treat the outliers as these are relevant data. We also, know that we can take care of the outliers by transforming variables, i.e. Scaling. Further down the line we will scale the variable, hence these data will not affect out Model building.

Univariate Analysis

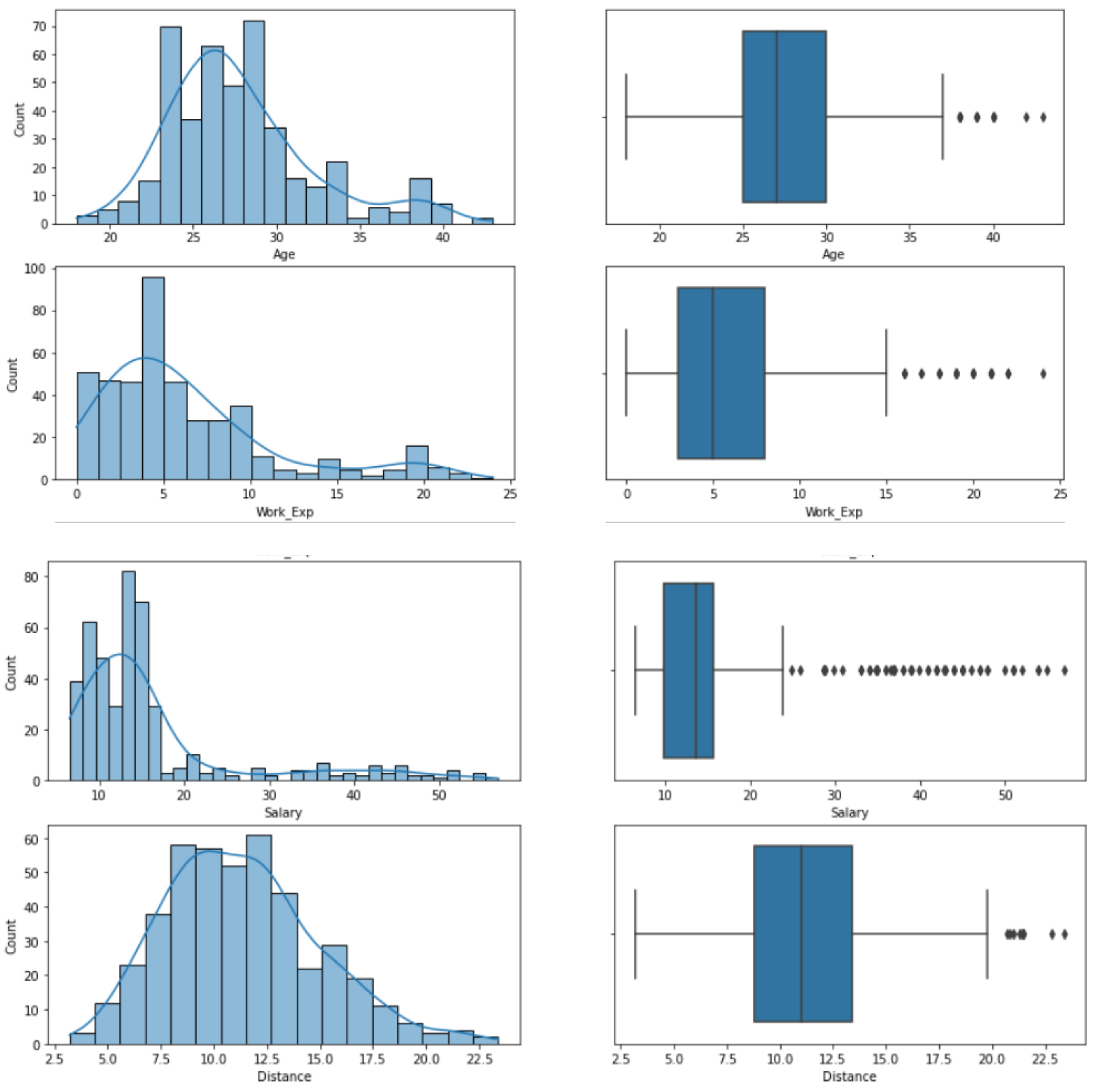


Figure 2: Histplot and Boxplot of Continuous Variable.

Inference:

- All the variables are positively skewed i.e right skewed.
- Age and Distance variable distribution is approximately symmetric.
- In the company maximum employees are of age between 25-30.
- Maximum employees are having working experience from 3 to 8 yrs.
- Maximum employees are in salary bracket of 10 to 18 units. And Distance covered by the employees are between 8 to 13km.

Let us now look at the Countplot of all the categorical variables

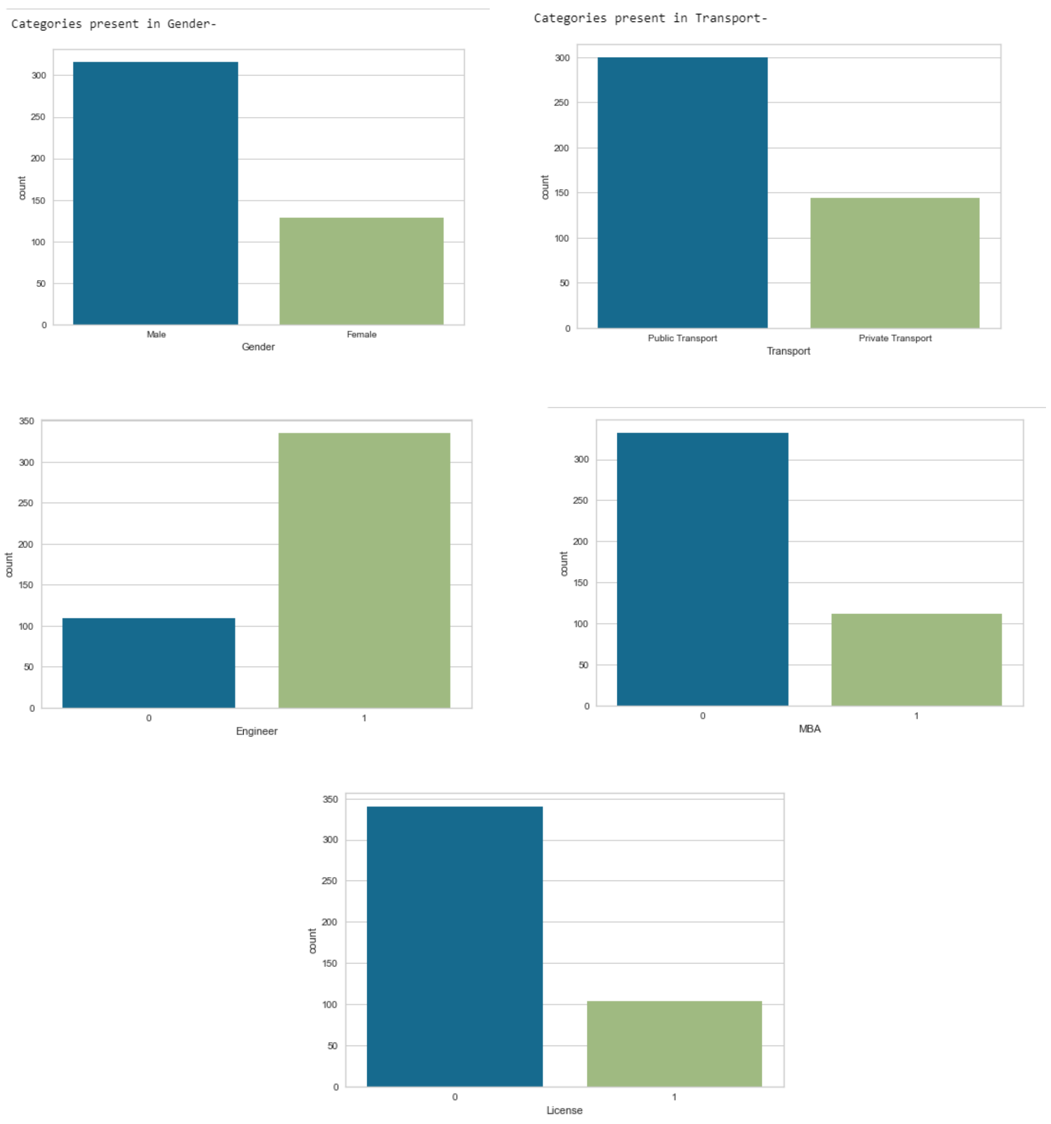


Figure 3: Countplot of Categorical Variable.

Inference:

- As per the count plot of these categorical variable it seems that Male count is more in the data.
- Employees are mostly commuting in Public Transport as we see that employee holding driving license is also less.
- There are more engineer in the company.
- Number of employees holding MBA degree are less.

Bivariate Analysis

Plotting scatter plot to show the relationship between 2 numeric variables.

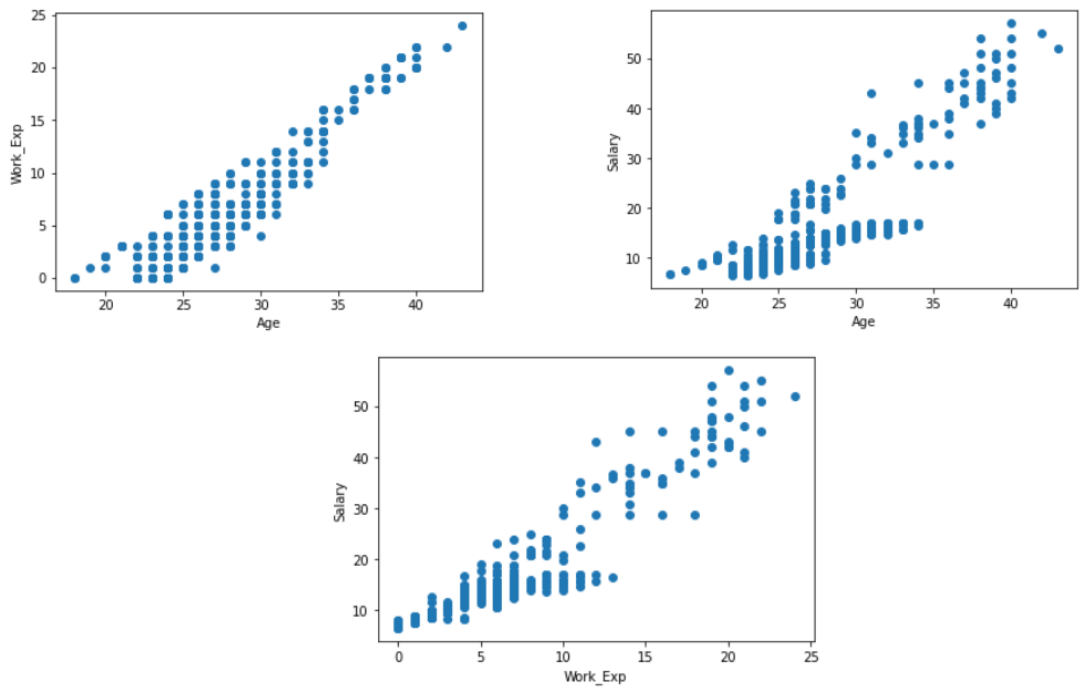


Figure 4: Scatterplot of numeric vs numeric variables

Inference:

We see that as the age increase, employee working experience increases, and also as the work exp. Increase the Salary of employee also increases.

Count plot between Categorical Vs Target variable

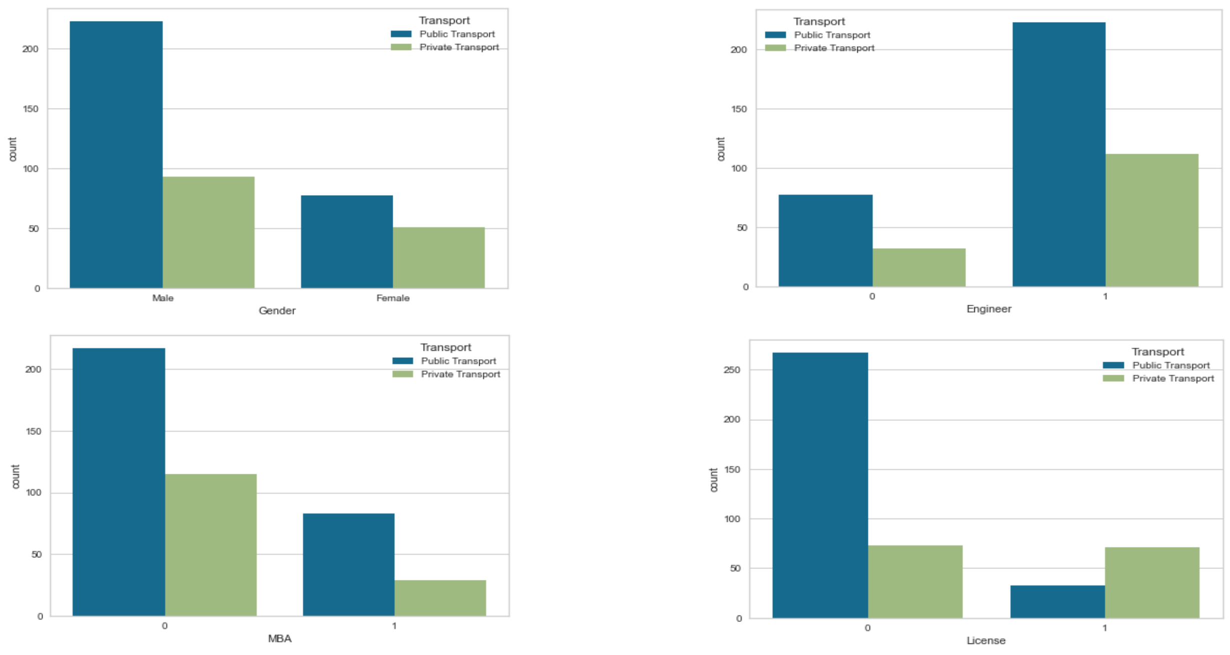


Figure 5: Count plot of Categorical Vs Target Variables

Inference:

- We see that among Male gender, most the male employees are using Public Transport. Same with female employee, most of them are using public transport.
- Employee who are engineers are commuting mostly with public transport.
- As in our dataset we have maximum non-MBA's, we see that they are using public transport more than private. Same trend is seen with employees who are holding MBA degree are mostly using public transport.
- Since most of the employees are not having driving license, they are bound to use Public Transport.

Numeric Vs Target Variable

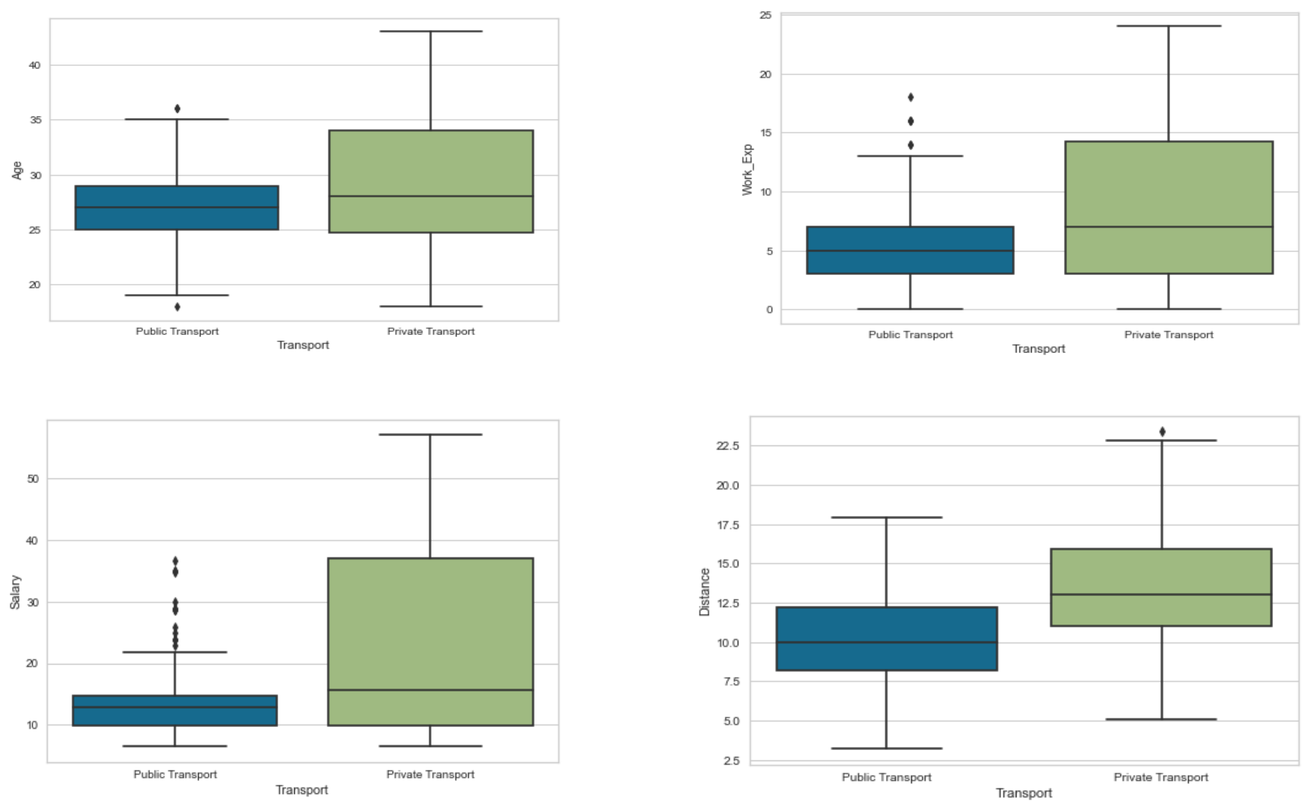


Figure 6: Box plot of Numeric Vs Target Variable

Inference:

- Employee in the age range of 25-35 are mostly using Private transport.
- Also, employee with working experience of years between 3 to 13 are also using Private transport.
- Employees earning 10.0 unit to almost 40.0 units are using Private transport for commuting.
- Distance also plays a vital role in choice of transport. Max distance range travelled by employees are between 10.0 to almost 17.0 km are opting for private transport than public transport.

Multi Variate Analysis

Analyzing the relation between continuous variable using Pairplot and Correlation Heatmap.

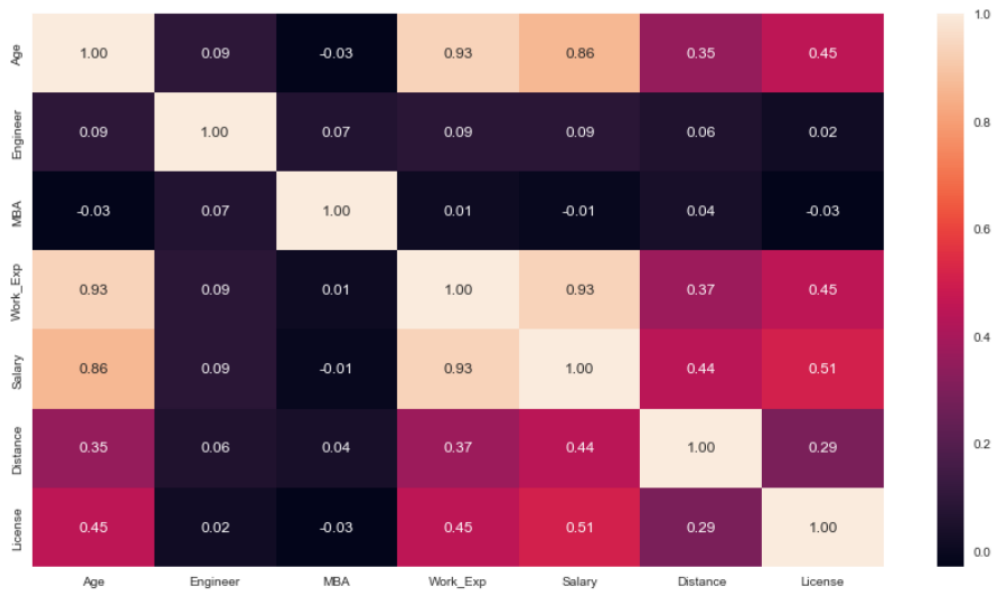


Figure 7: Correlation Plot

Inference:

- Age is showing highly positive correlation with Salary and Work_Exp variable and vice versa.
- Distance is showing very low positive correlation with Salary, Work_Exp and Age and vice versa.

Note -For practical purposes correlations in the range of [-0.4, 0.4] are not considered to be important.

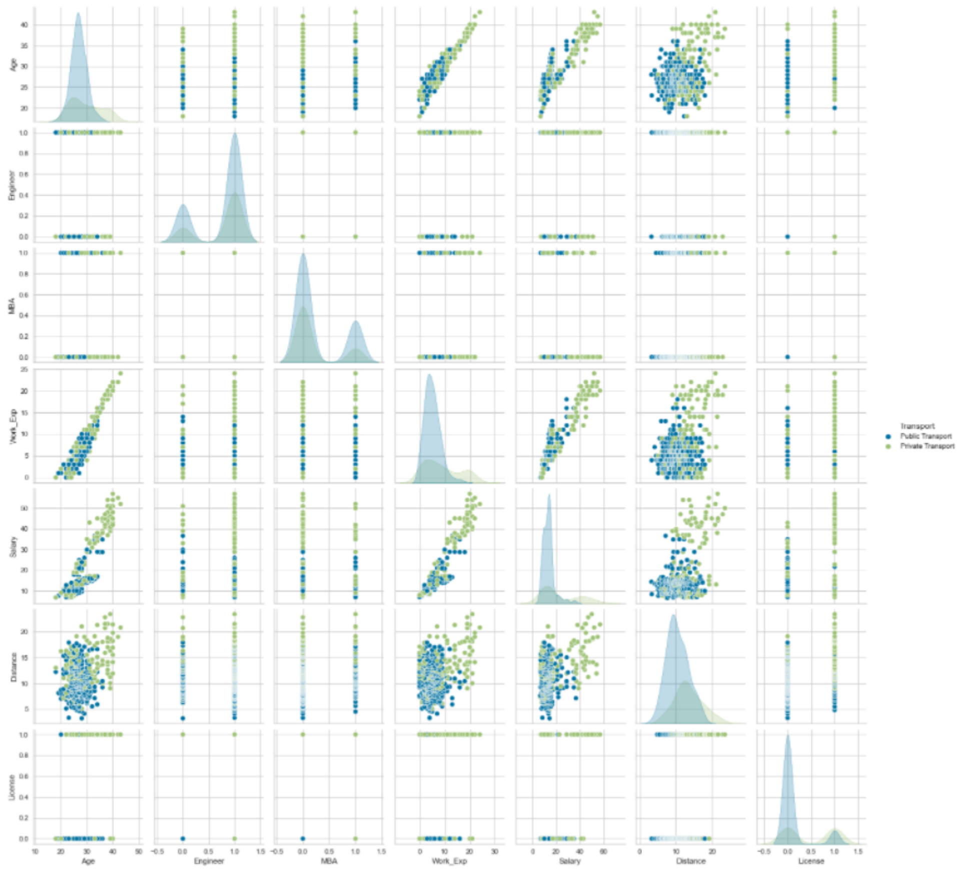
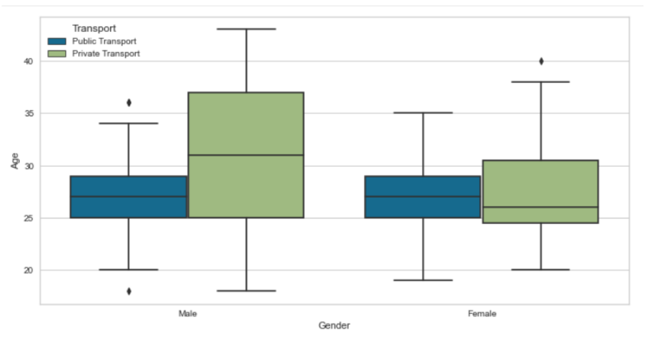
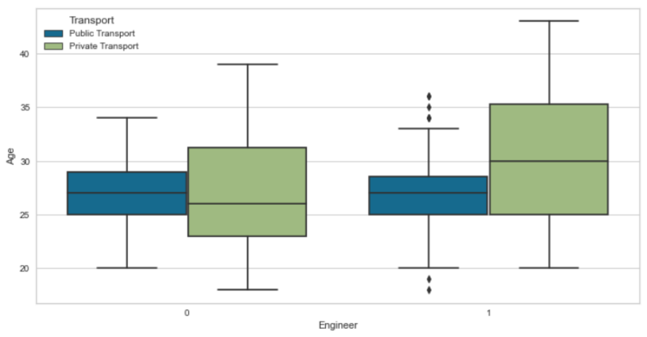


Figure 8: Pairplot

- We observe in this Pairplot that density distribution of data is showing that employee are opting for public transport then private transport.

Multivariate Analysis between Multiple Variables with Target variable as hue





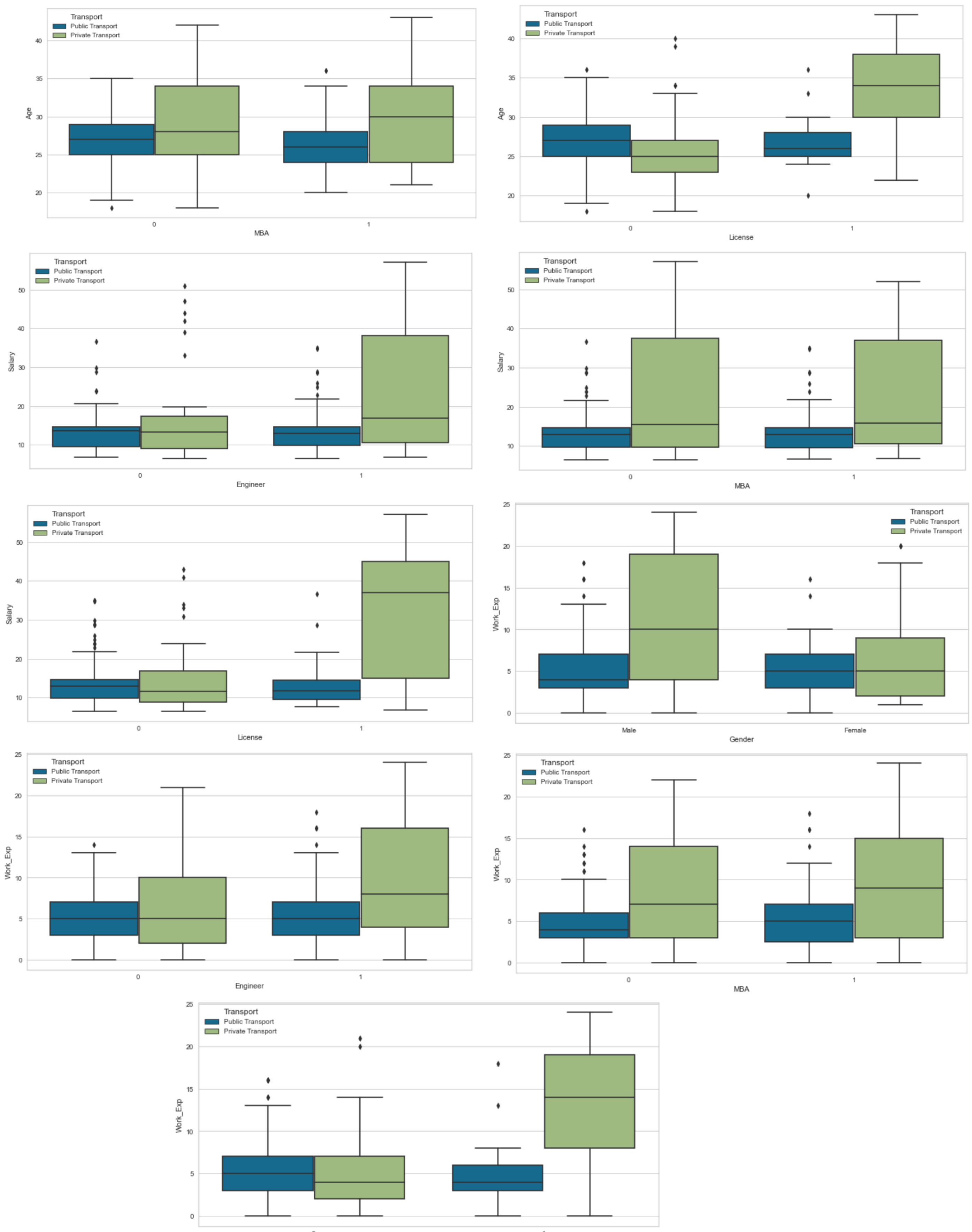


Figure 9: Boxplot showing Relation with Multiple Variables

Inference: Using the graph, we can compare the range and distribution of the Gender, Engineer, MBA, License for Public and Private Transport. We observe that there is a greater variability for Private Transport among all the variable as well as few outliers.

Also, since the notches in the boxplots do not overlap, we can conclude that with 95% confidence, that the true medians do differ in all the cases.

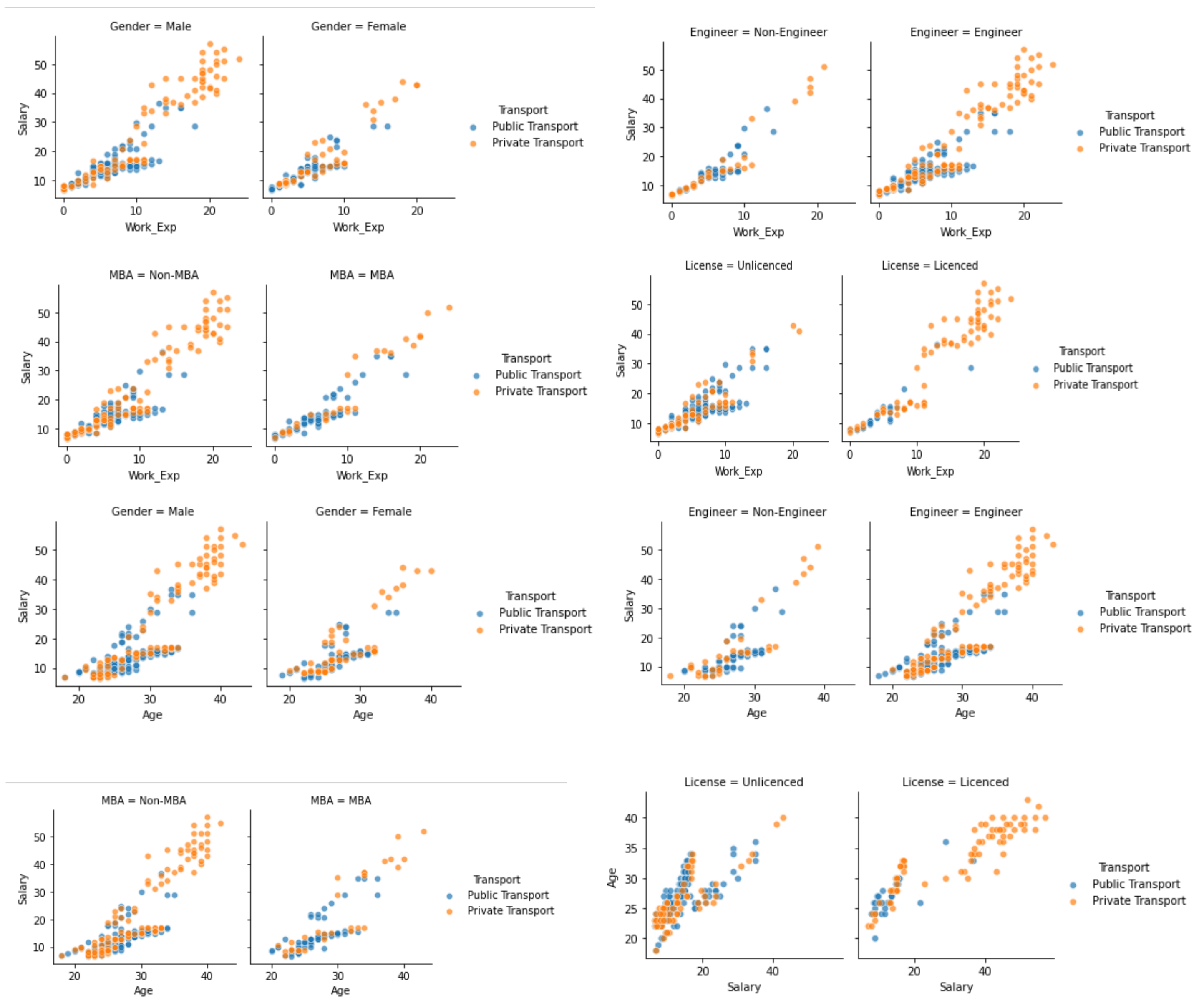


Figure 10: Scatterplot showing Relation with Multiple Variables

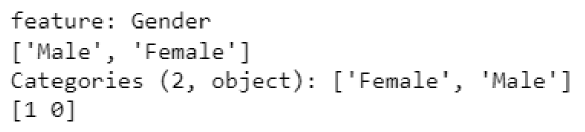
Inference: The above plot shows Multi-plot grid for plotting conditional relationship between the variables. A scatter plot of two variables would help in determining the relationship between the variables how the increase or decrease in one affects the other. Keeping hue as Transport the above plots are showing the variables are affecting each other. Like, as age increases salary of employee increases and showing how the choice of Transport also differ.

EDA Insight: Our Transport dataset has record of 444.
With Male of 316 and female 128 counts. Age of the employee vary from 18yrs to 43yrs.
Total no of engineers among those employees are 335 and rest are non-engineers i.e 109.
Total MBA employees are 111 and rest 332 are not MBA's. Work_Exp of employee varies from 0-24 years. Hence, Salary of employee ranges from 6.5unit to 57units.

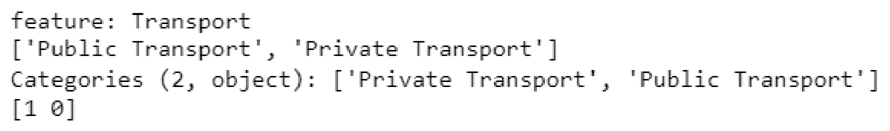
Mostly male who are holding degree of engineer and MBA, having high salary bracket are commuting with their private transports with higher distance between home and office. And those with moderate salary bracket and moderate distance are opting for public transport though they are holding driving license. Maybe company can investigate these employees who are license holder are opting for public transport. There are 104 employee who are having driving license out of which 71 employees use private transport and rest 33 public transport. We also see that 300 employees are using public transport and rest are using private transport. The complete EDA is showing that employee are opting more for public transport than private.

1.3) Encode the data (having string values) for Modelling. Is Scaling necessary here or not? Data Split: Split the data into train and test (70:30).

The performance of a machine learning model not only depends on the model and the hyperparameters but also on how we process and feed different types of variables to the model. Since most machine learning models only accept numerical variables, pre-processing the categorical variables becomes a necessary step. We need to convert these categorical variables to numbers such that the model can understand and extract valuable information.
We have done label encoding in our data, where it is simply converting each value in a column to a number. For e.g., Non-MBA-0 And MBA-1… We have done the same using pd.categorical().codes. as below:

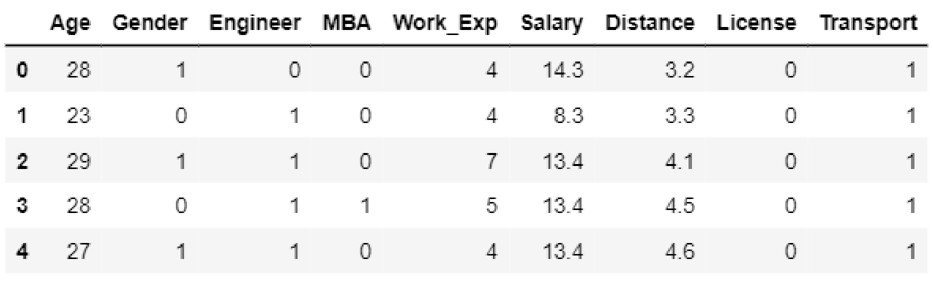


```
feature: Gender
['Male', 'Female']
Categories (2, object): ['Female', 'Male']
[1 0]
```



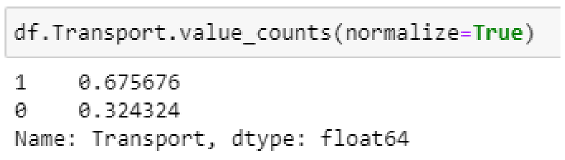
```
feature: Transport
['Public Transport', 'Private Transport']
Categories (2, object): ['Private Transport', 'Public Transport']
[1 0]
```

Let’s also check few rows of the data to confirm the conversion.



	Age	Gender	Engineer	MBA	Work_Exp	Salary	Distance	License	Transport
0	28	1	0	0	4	14.3	3.2	0	1
1	23	0	1	0	4	8.3	3.3	0	1
2	29	1	1	0	7	13.4	4.1	0	1
3	28	0	1	1	5	13.4	4.5	0	1
4	27	1	1	0	4	13.4	4.6	0	1

Let’s see the unique values in our Target column



```
df.Transport.value_counts(normalize=True)
```

```
1    0.675676
0    0.324324
Name: Transport, dtype: float64
```

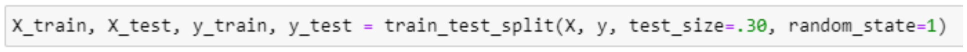
Here, 1-Public Transport and
0-Private Transport.

Dataset Preparation (splitting and normalization)

When using machine learning algorithms, we should always split our data into a training set and test set. The data set consists of record of 444 employees in total. To train and test our model we will be using 444 records. We will split the data in Train and Test dataset in 70:30 ratio.

- Training data is the **data you use to train an algorithm or machine learning model to predict the outcome you design your model to predict**.
- Test data is used to measure the performance, such as accuracy or efficiency, of the algorithm you are using to train the machine.

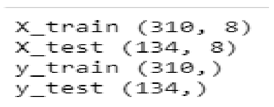
We split the data into training and test sets in Python using scikit-learn's built-in train_test_split()



```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.30, random_state=1)
```

The **test_size** refers to the number of observations that we want to put in the training data and the test data. If we specify a test_size of 0.3, our test_size will be 30 percent of the original data, therefore leaving the other 70 percent as training data.

The **random_state** is a parameter that allows you to obtain the same results every time the code is run. **train_test_split()** makes a random split in the data, which is problematic for reproducing the results. Therefore, it's common to use random_state. The choice of value in random_state is arbitrary. Below we can see the shape of our Train and Test dataset.



```
X_train (310, 8)
X_test (134, 8)
y_train (310,)
y_test (134,)
```

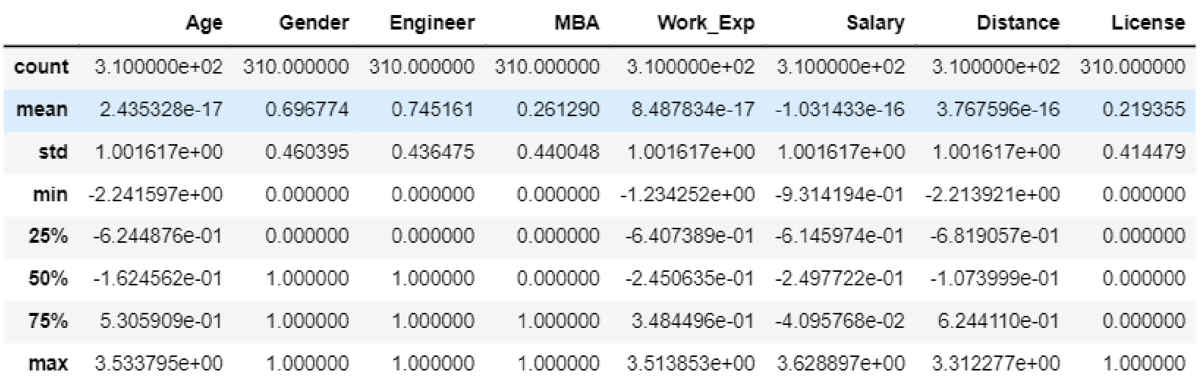
Is Scaling necessary here?

Scaling is a technique to standardize the independent features present in the data in a fixed range. It is performed during the data pre-processing to handle highly varying magnitudes or values or units. If feature scaling is not done, then a machine learning al gorithm tends to weigh greater values, higher and consider smaller values as the lower values, regardless of the unit of the values.

Here, in our problem we are going to use StandardScaler. Why?

- Most of our independent variable are in '1' and '0'.
- Our Dataset is very small in shape.
- The outliers present in the date will not be affected by standardization

To standardize our data, we need to import the *StandardScalar* from the sklearn library and apply it to our dataset.



	Age	Gender	Engineer	MBA	Work_Exp	Salary	Distance	License
count	3.100000e+02	310.000000	310.000000	310.000000	3.100000e+02	3.100000e+02	3.100000e+02	310.000000
mean	2.435328e-17	0.696774	0.745161	0.261290	8.487834e-17	-1.031433e-16	3.767596e-16	0.219355
std	1.001617e+00	0.460395	0.436475	0.440048	1.001617e+00	1.001617e+00	1.001617e+00	0.414479
min	-2.241597e+00	0.000000	0.000000	0.000000	-1.234252e+00	-9.314194e-01	-2.213921e+00	0.000000
25%	-6.244876e-01	0.000000	0.000000	0.000000	-6.407389e-01	-6.145974e-01	-6.819057e-01	0.000000
50%	-1.624562e-01	1.000000	1.000000	0.000000	-2.450635e-01	-2.497722e-01	-1.073999e-01	0.000000
75%	5.305909e-01	1.000000	1.000000	1.000000	3.484496e-01	-4.095768e-02	6.244110e-01	0.000000
max	3.533795e+00	1.000000	1.000000	1.000000	3.513853e+00	3.628897e+00	3.312277e+00	1.000000

We can notice how scaling the features brings everything into perspective. The features are now more comparable and will have a similar effect on the learning models.

It’s now time to train some machine learning algorithms on our data to compare the effects of different scaling techniques on the performance of the algorithm. Let us see the effect of scaling on 4 algorithms in particular: Logistic Regression, K-Nearest Neighbours, Bagging, and Boosting.

1.4) Apply Logistic Regression. Interpret the inferences of both models.

Logistic regression is one of the most popular machine learning algorithms for binary classification. This is because it is a simple algorithm that performs very well on a wide range of problems.

Logistic Regression is **a Machine Learning algorithm which is used for the classification problems**, it is a predictive analysis algorithm and based on the concept of probability. The hypothesis of logistic regression tends it to limit the cost function between 0 and 1.

For the first iteration, let us build the Logistic Regression model with default Hyperparameters. We will import LogisticRegression from **sklearn.linear_model** and fit the regressor over the training data. The performance of the model can be evaluated by calculating the **model score**. Depending upon the nature of fit i.e. over fit or under fit, we will go ahead with Hyperparameter tuning to arrive at better-fitted model.

Results from the first iteration –

```
Train Accuracy is : 0.7967741935483871

Test Accuracy is : 0.8208955223880597

Train ROC-AUC score is : 0.8319193061840121

Test ROC-AUC score is : 0.818840579710145


Logistic Regression Classification report Train set :
              precision    recall  f1-score   support

           0       0.78      0.53      0.63       102
           1       0.80      0.93      0.86       208

    accuracy                           0.80       310
   macro avg       0.79      0.73      0.75       310
weighted avg       0.79      0.80      0.78       310


Logistice Regression Classification report Test set :
              precision    recall  f1-score   support

           0       0.78      0.60      0.68        42
           1       0.83      0.92      0.88        92

    accuracy                           0.82       134
   macro avg       0.81      0.76      0.78       134
weighted avg       0.82      0.82      0.81       134
```

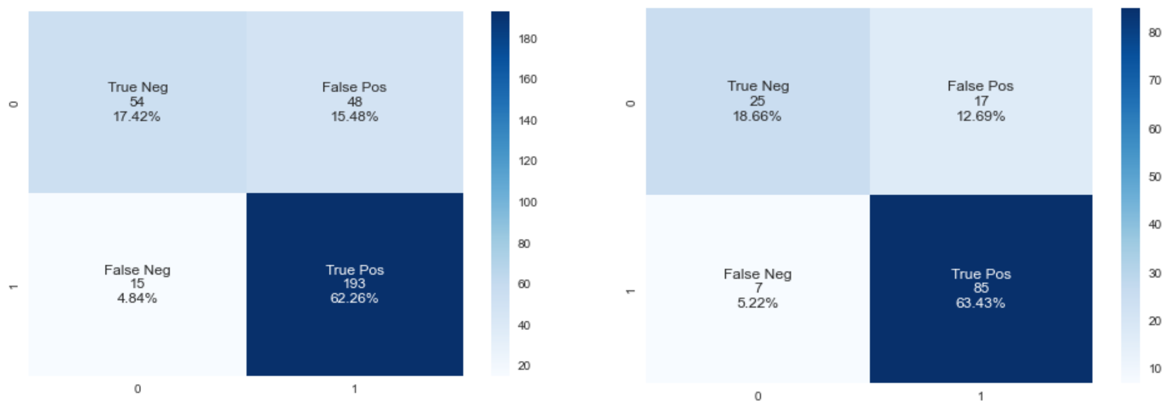


Figure 11: Classification report and Confusion Matrices for Train & Test Set- Logistic Regression Model

Observation: *We can see that the* ***training accuracy*** *of* ***Logistic Regression*** *with default hyperparameters is 80% accuracy. Whereas when we fit the same model on* ***testing data****, the accuracy is 82% approx.* The classifier made a total of 310 predictions with Train dataset. (e.g., 310 employees were being tested whether they will choose public or private transport).

Negative Class = No(0)=Private Transport,   Positive Class = Yes(1)=Public Transport
True Positive:
It means the actual value and also the predicted values are the same.
In our case the actual value is also Public Transport and the model prediction is also Public Transport.
If you observe for the TP cell the positive value is the same for Actual and predicted i.e. 193 for train set and 85 in test set.

False Negative:
This means the actual value is positive in our case it is public transport but the model has predicted it as negative i.e., private transport. So the model has given the wrong prediction, It was supposed to give a positive(public transport) but it has given a negative(private transport) so whatever the negative output we got is false, hence the name False Negative i.e. 15 in train set and 7 in test set.

False Positive:
This means the actual value is negative in our case it is private transport but the model has predicted it as positive i.e., public transport. So the model has given the wrong prediction, it was supposed to give negative(private transport) but it has given positive(public transport) so whatever the positive output we got is false, hence the name False Positive i.e. 48 in train set and 17 in test set.

True Negative:
It means the actual value and also the predicted values are the same. In our case, the actual values are also private transport and the Prediction is also private transport i.e. 54 in train set and 25 in test set.

Note: This turned out to be pretty descent classifier for our dataset considering the model is saying the most preferred transport is public by the employees.

```
Logistic Regression Classifier Train: ROC AUC=0.832
Logistic Regression classifier test: ROC AUC=0.819
```

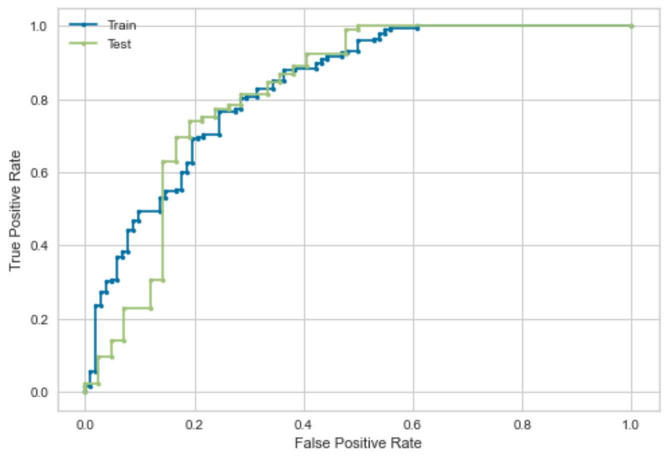


Figure 12: ROC AUC plot for Train & Test Set- Logistic Regression Model

This plot is commonly used to summarizes the performance of a classifier over all possible thresholds. It is generated by plotting the **True Positive Rate** (y-axis) against the **False Positive Rate** (x-axis) as you vary the threshold for assigning observations to a given class.

Observation: From the above ROC AUC curve we can see that with default parameter tuning, the ***accuracy*** of the Train and Test model are approx. equal i.e. for Train it is 83% and Test it is 82%.

Let’s look at the Feature Importance,

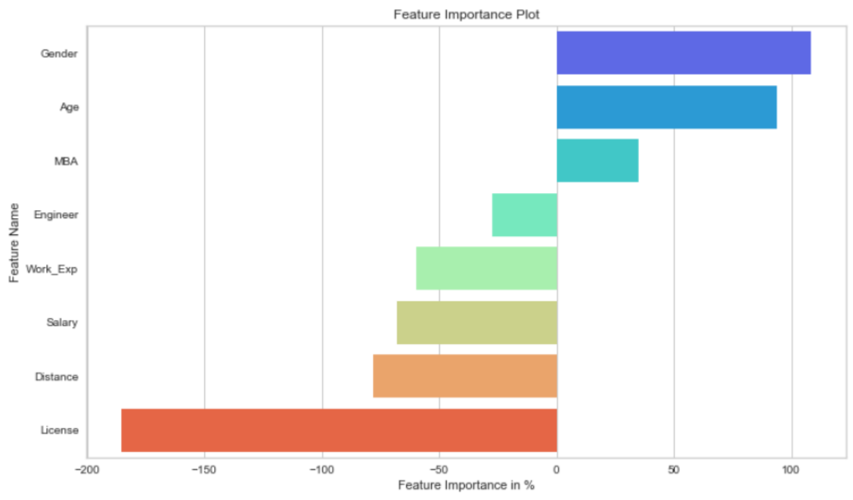


Figure 13: Feature Importance using Logistic Regression Classifier

Feature: 0, Score: 0.93857	Age
Feature: 1, Score: 1.08562	Gender
Feature: 2, Score: -0.27267	Engineer
Feature: 3, Score: 0.34690	MBA
Feature: 4, Score: -0.59732	Work_Exp
Feature: 5, Score: -0.67838	Salary
Feature: 6, Score: -0.78262	Distance
Feature: 7, Score: -1.85525	License

Observation: As we know that this is a classification problem with classes 0 and 1. Notice that the coefficients are both positive and negative. The positive scores indicate a feature that predicts class 1(Public Transport), whereas the negative scores indicate a feature that predicts class 0(Private Transport). In this plot Gender, Age and MBA is showing the outcome of Public transport whereas Engineer, Work_Exp, Salary, Distance, and License are playing role in the outcome of private transport.

In 2nd Iteration we will use hyperparameters in Logistic Regression Classifier Model to check the performance of our Dataset.

Hyperparameter chosen for Tuning:

solver*{‘newton-cg’, ‘lbfgs’, ‘liblinear’, ‘sag’, ‘saga’}, default=’lbfgs’*

Algorithm to use in the optimization problem. Default is ‘lbfgs’. To choose a solver, you might want to consider the following aspects:

- For small datasets, ‘liblinear’ is a good choice, whereas ‘sag’ and ‘saga’ are faster for large ones;
- For multiclass problems, only ‘newton-cg’, ‘sag’, ‘saga’ and ‘lbfgs’ handle multinomial loss;
- ‘liblinear’ is limited to one-versus-rest schemes.

penalty*{‘l1’, ‘l2’, ‘elasticnet’, ‘none’}, default=’l2’*

Specify the norm of the penalty:

- 'none': no penalty is added;
- 'l2': add a L2 penalty term and it is the default choice;
- 'l1': add a L1 penalty term;
- 'elasticnet': both L1 and L2 penalty terms are added.

C *float, default=1.0*

- Inverse of regularization strength; must be a positive float. Like in support vector machines, smaller values specify stronger regularization.

Result from the Tuned Model:

```
Train and Test Accuracy, ROC-AUC curve scores after Hyper Tunning

Train Accuracy is : 0.8129032258064516

Test Accuracy is : 0.7985074626865671

Train ROC-AUC score is : 0.8023190045248869

Test ROC-AUC score is : 0.8253105590062112


Logistic Regression-Tuned Classification report Train set :
              precision    recall  f1-score   support

           0       0.92      0.47      0.62       102
           1       0.79      0.98      0.88       208

    accuracy                           0.81       310
   macro avg       0.86      0.73      0.75       310
weighted avg       0.83      0.81      0.79       310


Logistice Regression-Tuned Classification report Test set :
              precision    recall  f1-score   support

           0       0.86      0.43      0.57        42
           1       0.79      0.97      0.87        92

    accuracy                           0.80       134
   macro avg       0.82      0.70      0.72       134
weighted avg       0.81      0.80      0.78       134
```

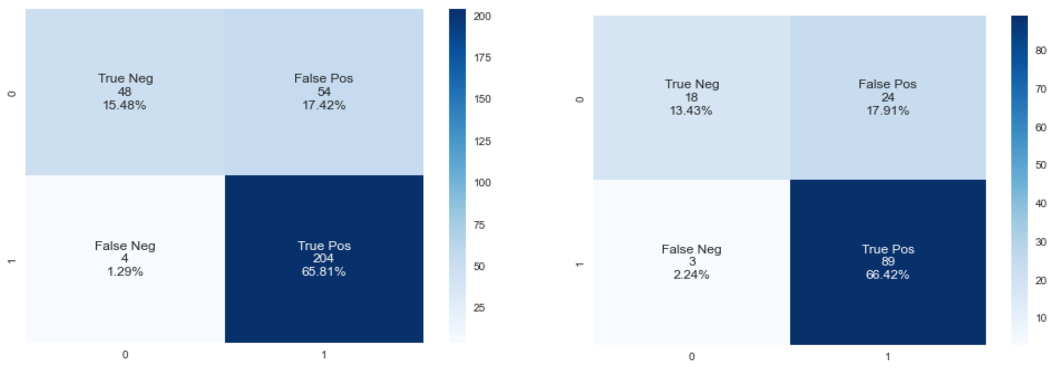


Figure 14: Classification Report and Confusion Matrices for Train & Test Set- Logistic Regression Model-Tuned

Observations:
Best parameters we got: 'C': 10, 'penalty': 'l2', 'solver': 'liblinear'}

*Now, when we applied the **GridSearchCV** algorithm to our model which is used to select the **best hyperparameters**, we see that the training accuracy is same i.e. 80% approx and testing accuracy has increased to 84%. The **training accuracy** is **unaffected** after tuning the hyperparameters. Still giving a good fit for our dataset.*

- **true positives (TP):** These are cases in which we predicted public transport, and employees do prefer public transport. 204 times in Train and 89 times in Test dataset.
- **true negatives (TN):** We predicted private transport, and actually employee do prefer private transport. 48 times in Train set and 18 times in test dataset.
- **false positives (FP):** We predicted public transport, but actually employee prefers private transport. (Also known as a "Type I error."). 54 times in Train set and 24 times in test set.
- **false negatives (FN):** We predicted private transport, but they actually prefer public transport. (Also known as a "Type II error.") 4 times in train set and 3 times in test set.

Note: This turned out to be pretty decent classifier even with hyperparameters for our dataset considering the model is saying the most preferred transport is public by the employees.

```
Logistic Regression-Tuned Classifier Train: ROC AUC=0.802
Logistic Regression-Tuned classifier test: ROC AUC=0.825
```

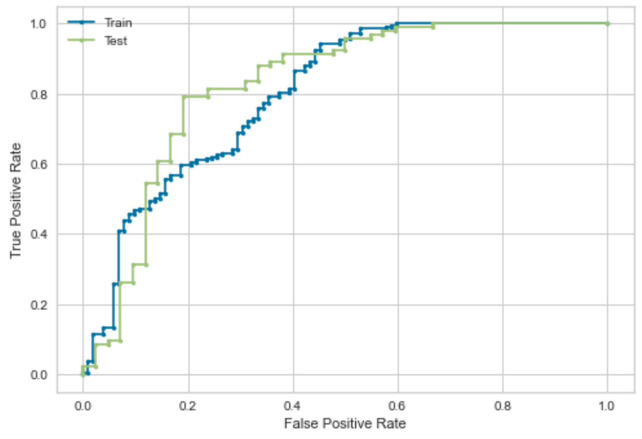


Figure 15: ROC AUC plot for Train & Test Set- Logistic Regression Model-Tuned

From the ROC-AUC curve we can see that after hyperparameter tuning the ***accuracy*** of the model has slightly decreased to a certain extent against the one when we used default hyperparameters. The ***accuracy*** of the Train and Test model are approx. equal i.e. for Train it is 83% and Test it is 82%.

Note: Building Model with hyperparameters, has slightly increase the performance of the model on our dataset.

Model Evaluation:

Model Name	Training Data	Test Data	Train AUC-ROC Score	Test AUC-ROC Score
Logistic Regression Model	0.796774	0.820896	0.831919	0.818841
Logistic Regression Model-Tuned	0.812903	0.798507	0.802319	0.825311

1.5) Apply KNN Model. Interpret the inferences of each model.

KNN: The abbreviation KNN stands for "**K-Nearest Neighbour**". It is a supervised machine learning algorithm. The algorithm can be used to solve both classification and regression problem statements. The number of nearest neighbours to a new unknown variable that has to be predicted or classified is denoted by the symbol 'K'. The KNN algorithm employs the same principle. Its aim is to locate all of the closest neighbours around a new unknown data point in order to figure out what class it belongs to. **It's a distance-based approach**.

In the above section 1.3), we separate the data into training and test data. Which is needed for objective model evaluation.
Now we proceed to fit a KNN model on the training data using scikit-learn.

To fit a model from scikit-learn, we start by creating a model of the correct class. At this point, we also need to choose the values for our hyperparameters. For the kNN algorithm, we need to choose the value for k, which is called **n_neighbors(default=5;** Number of neighbours to use by default for **kneighbors** queries.) in the scikit-learn implementation.
*Refer to Python code book for the same.

For the first iteration, we are building the kNN model with default hyperparameters. Depending upon the nature of the fit that is overfit and underfit we will go ahead with the pattern to arrive at better fitted model.

Result of 1st Iteration:

```
Train Accuracy is : 0.8387096774193549

Test Accuracy is : 0.7835820895522388

Train ROC-AUC score is : 0.923006221719457

Test ROC-AUC score is : 0.7553053830227743


Classification report Train set :
              precision    recall  f1-score   support

           0       0.83      0.64      0.72       102
           1       0.84      0.94      0.89       208

    accuracy                           0.84       310
   macro avg       0.84      0.79      0.80       310
weighted avg       0.84      0.84      0.83       310

Classification report Test set :
              precision    recall  f1-score   support

           0       0.71      0.52      0.60        42
           1       0.81      0.90      0.85        92

    accuracy                           0.78       134
   macro avg       0.76      0.71      0.73       134
weighted avg       0.78      0.78      0.77       134
```

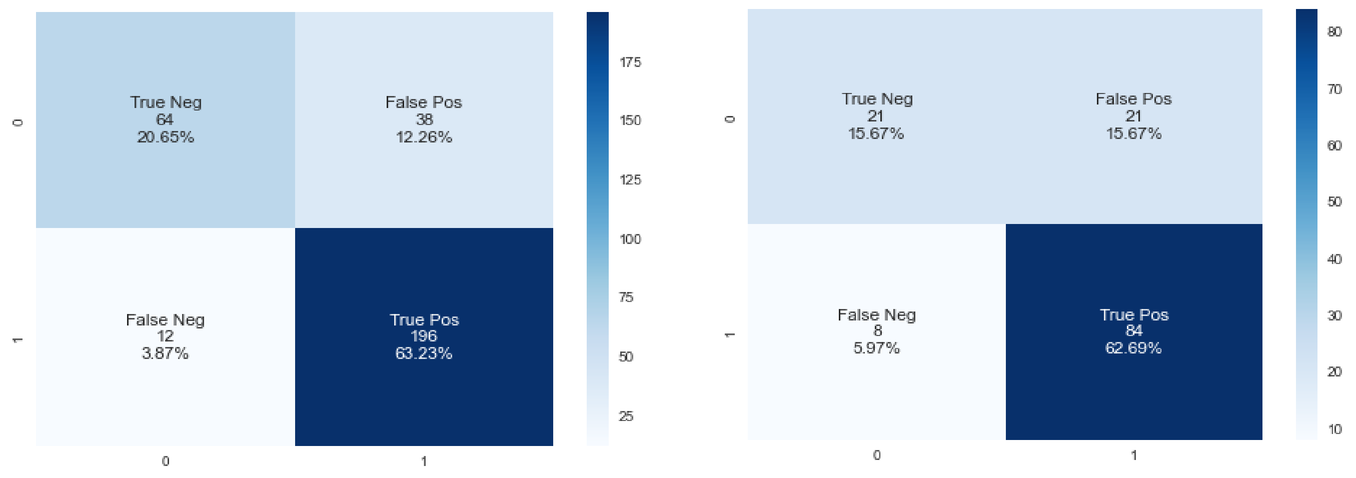


Figure 16: Classification Report and Confusion Matrices for Train & Test Set-KNN Model

Observation on Confusion Matrix:

- True Positive (TP) = 196; meaning 196 public transport class data points were correctly classified by the model.
- True Negative (TN) = 65; meaning 65 private transport class data points were correctly classified by the model
- False Positive (FP) = 37; meaning 37 private transport class data points were incorrectly classified as belonging to the public transport class by the model
- False Negative (FN) = 13; meaning 13 public transport class data points were incorrectly classified as belonging to the private transport class by the model

Likewise for Test Dataset: TP=84, TN=21, FP=21, FN=8

*We can see that the **training accuracy** of **KNN Classifier** with default hyperparameters is **84%** accuracy. Whereas when we fit the same model on **testing data**, the accuracy reduces to **78%** approx. Clearly the **model has overfit the training data** and also the **variance is very high** i.e. the model is behaving differently for different samples.*

```
Knn Model Train: ROC AUC=0.924
Knn Model test: ROC AUC=0.742
```

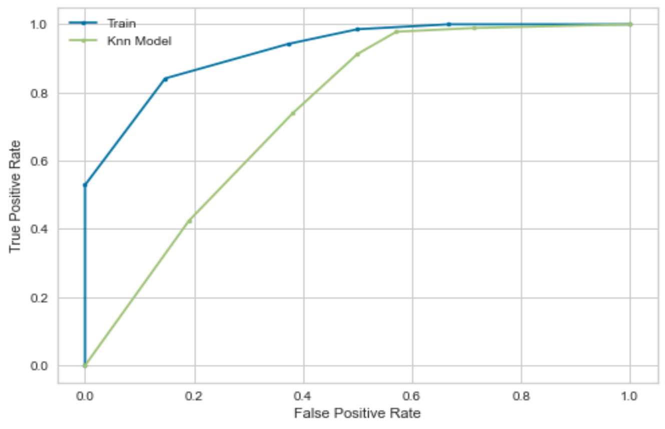


Figure 17: ROC AUC for Train & Test Set-KNN Model

Observation: From the above ROC AUC curve we can see that with default parameter tuning, the ***accuracy*** of the Train model is higher than that of Test dataset i.e. for Train it is 92% and Test it is 74%.

Let's Look at the feature importance in KNN Model

Feature: 0, Score: 0.02774	Age
Feature: 1, Score: 0.02710	Gender
Feature: 2, Score: 0.00645	Engineer
Feature: 3, Score: -0.00516	MBA
Feature: 4, Score: 0.02387	Work_Exp
Feature: 5, Score: 0.02968	Salary
Feature: 6, Score: 0.06323	Distance
Feature: 7, Score: 0.02452	License

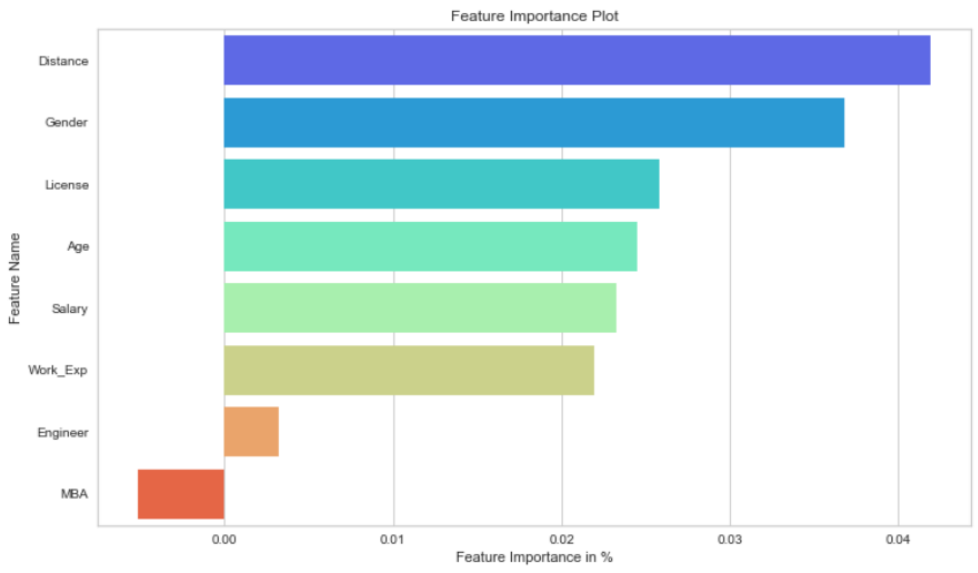


Figure 18: Feature Importance using KNN Classifier

From above plot we can see the feature importance of the varaiables. We see that except MBA rest variable are contibuting towards the prediction of public transport.

Now we must choose the set of Hyperparameters that will find the best value of k, we are going to use a tool called GridSearchCV. This is a tool that is often used for tuning hyperparameters of machine learning models. In our case, it will help by automatically finding the best value of k for your dataset.
GridSearchCV is available in scikit-learn, and it has the benefit of being used in almost the exact same way as the scikit-learn models.

Hyperparameters chosen for tuning-

- **n_neighbors*int, default=5***
  - Number of neighbors to use by default for **kneighbors** queries.
- **weights*{'uniform', 'distance'} or callable, default='uniform'***

  Weight function used in prediction. Possible values:

'uniform' : uniform weights. All points in each neighborhood are weighted equally.

'distance' : weight points by the inverse of their distance. in this case, closer neighbors of a query point will have a greater influence than neighbors which are further away.

[callable] : a user-defined function which accepts an array of distances, and returns an array of the same shape containing the weights.

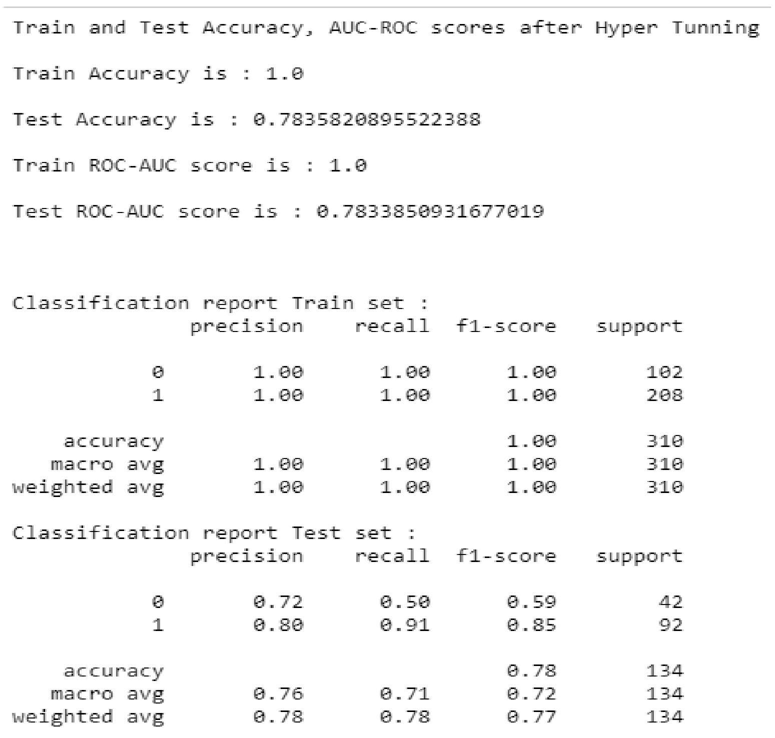
- **algorithm*{'auto', 'ball_tree', 'kd_tree', 'brute'}, default='auto'***

  Algorithm used to compute the nearest neighbors:

  - 'ball_tree' will use **BallTree**
  - 'kd_tree' will use **KDTree**
  - 'brute' will use a brute-force search.
  - 'auto' will attempt to decide the most appropriate algorithm based on the values passed to **fit** method.

  Note: fitting on sparse input will override the setting of this parameter, using brute force.
- **leaf_size*int, default=30***
  - Leaf size passed to BallTree or KDTree. This can affect the speed of the construction and query, as well as the memory required to store the tree. The optimal value depends on the nature of the problem.

Result from the Tuned Model:



```
Train and Test Accuracy, AUC-ROC scores after Hyper Tunning

Train Accuracy is : 1.0

Test Accuracy is : 0.7835820895522388

Train ROC-AUC score is : 1.0

Test ROC-AUC score is : 0.7833850931677019


Classification report Train set :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       102
           1       1.00      1.00      1.00       208

    accuracy                           1.00       310
   macro avg       1.00      1.00      1.00       310
weighted avg       1.00      1.00      1.00       310

Classification report Test set :
              precision    recall  f1-score   support

           0       0.72      0.50      0.59        42
           1       0.80      0.91      0.85        92

    accuracy                           0.78       134
   macro avg       0.76      0.71      0.72       134
weighted avg       0.78      0.78      0.77       134
```

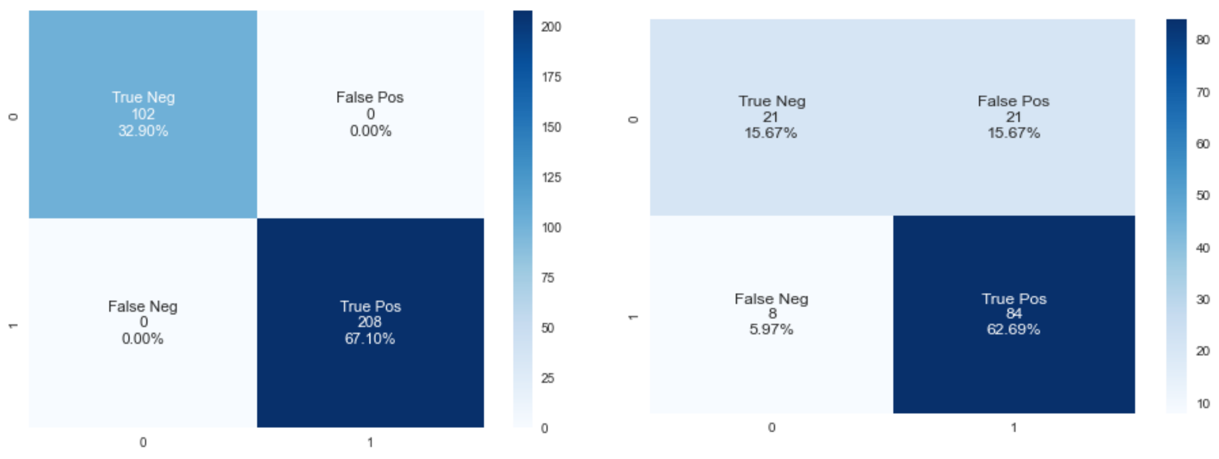


Figure 19: Classification Report and Confusion Matrices for Train & Test Set-KNN_Tuned Model

Inference:

The best parameters are {'algorithm': 'auto', 'leaf_size': 15, 'n_neighbors': 9, 'weights': 'distance'}.
We observed that after doing hyperparameter tuning using KNN model on our dataset, the model is showing extreme overfitting.
We achieved accuracy of 100% which is too good and test accuracy is showing 78%, hence major Overfitting.
Using GridSearchCV has not given us appropriate value of k and model performance.

Observation on Confusion Matrix:

- True Positive (TP) = 208; meaning 208 public transport class data points were correctly classified by the model.
- True Negative (TN) = 102; meaning 102 private transport class data points were correctly classified by the model
- False Positive (FP) = 0; meaning model did not predict anything incorrectly.
- False Negative (FN) = 0; meaning model did not predict anything incorrectly.

Likewise for Test Dataset: TP=84, TN=21, FP=21, FN=8

```
Knn Model Train: ROC AUC=1.000
Knn Model test: ROC AUC=0.783
```

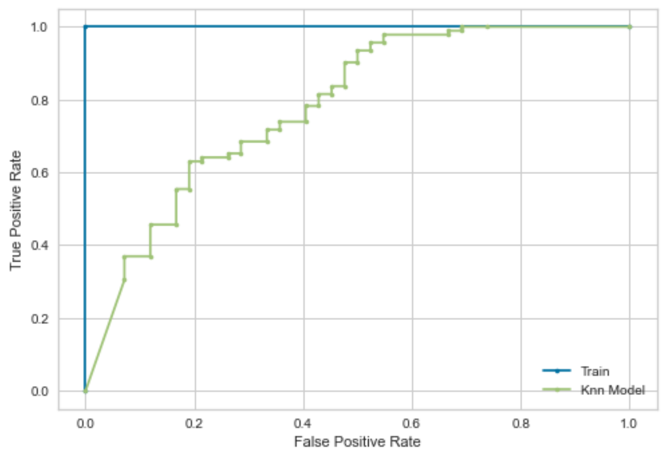


Figure 20: ROC AUC for Train & Test Set-KNN Model-Tuned

From the above ROC AUC curve, we can see that with hyperparameter tuning, the ***accuracy*** of the Train model is higher than that of Test dataset i.e. for Train it is 100% and Test it is 78%. GridSearchCV has clearly not helped the model to perform well.

KNN Model Evaluation

Model Name	Training Data	Test Data	Train AUC-ROC Score	Test AUC-ROC Score
KNN	0.83871	0.783582	0.923501	0.741718
KNN-Tuned	1.0	0.783582	1.0	0.783385

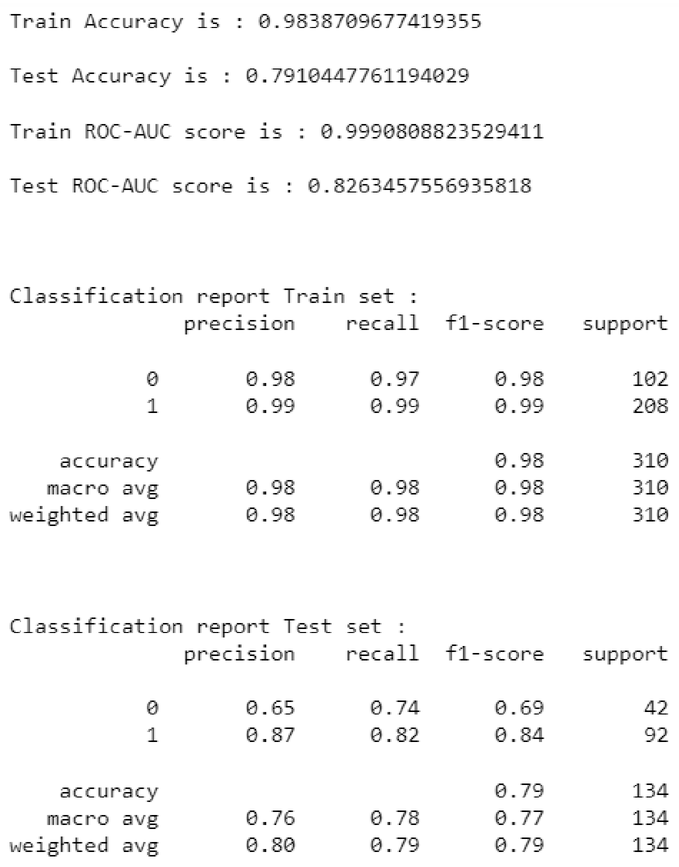
1.6) Bagging and Boosting, Model Tuning.

A Bagging classifier.

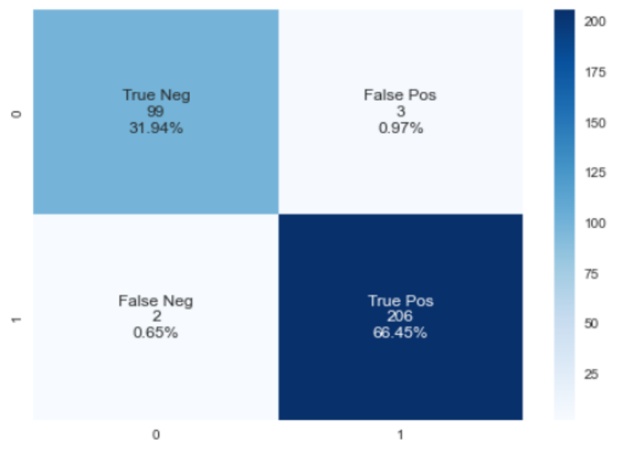
A Bagging classifier is an ensemble meta-estimator that fits base classifiers each on random subsets of the original dataset and then aggregate their individual predictions (either by voting or by averaging) to form a final prediction. Such a meta-estimator can typically be used as a way to reduce the variance of a black-box estimator (e.g., a decision tree), by introducing randomization into its construction procedure and then making an ensemble out of it. Bagging methods come in many flavours but mostly differ from each other by the way they draw random subsets of the training set. When samples are drawn with replacement, then the method is known as **Bagging**.

For the first iteration, we are building the Bagging model with default parameters. Depending upon the nature of the fit that is overfit and underfit we will go ahead with the pattern to arrive at better fitted model.

1st Iteration Result: We will us scikit-learn to apply bagging" BaggingClassifier()" to our dataset.



```
Train Accuracy is : 0.9838709677419355

Test Accuracy is : 0.7910447761194029

Train ROC-AUC score is : 0.9990808823529411

Test ROC-AUC score is : 0.8263457556935818


Classification report Train set :
              precision    recall  f1-score   support

           0       0.98      0.97      0.98       102
           1       0.99      0.99      0.99       208

    accuracy                           0.98       310
   macro avg       0.98      0.98      0.98       310
weighted avg       0.98      0.98      0.98       310


Classification report Test set :
              precision    recall  f1-score   support

           0       0.65      0.74      0.69        42
           1       0.87      0.82      0.84        92

    accuracy                           0.79       134
   macro avg       0.76      0.78      0.77       134
weighted avg       0.80      0.79      0.79       134
```



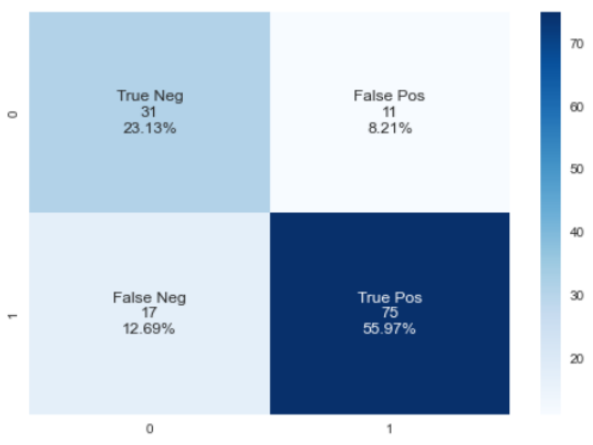


Figure 21: Classification Report and Confusion Matrices for Train & Test Set- Bagging Model

Observation on Confusion Matrix:

- True Positive (TP) = 206; meaning 206 public transport class data points were correctly classified by the model.
- True Negative (TN) = 99; meaning 99 private transport class data points were correctly classified by the model
- False Positive (FP) = 3; meaning 3 private transport class data points were incorrectly classified as belonging to the public transport class by the model
- False Negative (FN) = 2; meaning 2 public transport class data points were incorrectly classified as belonging to the private transport class by the model

Likewise for Test Dataset: TP=75, TN=31, FP=11, FN=17

Observations:
We can see that the ***training accuracy*** of ***Bagging Classifier*** with default hyperparameters is ***98%*** accuracy. Whereas when we fit the same model on ***testing data***, the accuracy reduces to ***79%*** approx. Clearly the ***model has majorly overfitted the training data*** and also the ***variance is very high*** i.e. the model is behaving differently for different samples.

```
Bagging Classifier Train: ROC AUC=0.999
Bagging classifier test: ROC AUC=0.826
```

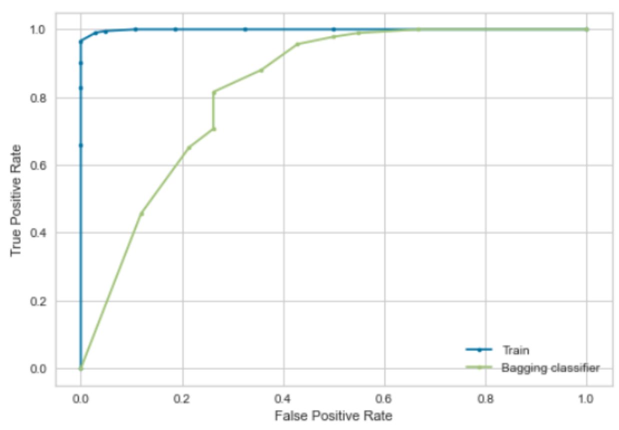


Figure 22: ROC AUC for Train & Test Set- Bagging Model

Observation: From the above ROC AUC curve we can see that with default parameter, the ***accuracy*** of the Train model is higher than that of Test dataset i.e. for Train it is ~100% and Test it is 83%.

Let's look at the Feature Importance.

Feature: 0, Score: 0.12426887	Age
Feature: 1, Score: 0.05653219	Gender
Feature: 2, Score: 0.01599047	Engineer
Feature: 3, Score: 0.01562708	MBA
Feature: 4, Score: 0.08450377	Work_Exp
Feature: 5, Score: 0.3584507	Salary
Feature: 6, Score: 0.22108971	Distance
Feature: 7, Score: 0.12353721	License

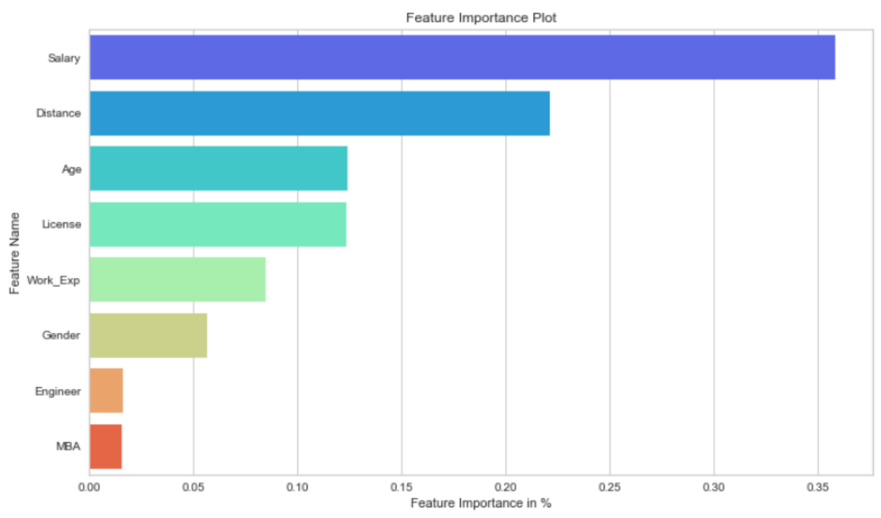


Figure 23: Feature Importance using Bagging Classifier

From above plot we can understand that few top features that are important to predict public transport are salary, distance, age, license, Work_Exp. And less contributing features are Gender, Engineer, MBA.

Hyperparameters chosen for tuning-

base_estimator*object, default=None*

The base estimator to fit on random subsets of the dataset. If None, then the base estimator is a **DecisionTreeClassifier**.

In this problem we are taking base estimator as **LogisticRegression, DecisionTreeClasiifier and RandomForestClassifier**.

n_estimators*int, default=10*

The number of base estimators in the ensemble.

max_samples*int or float, default=1.0*

The number of samples to draw from X to train each base estimator (with replacement by default).

- If int, then draw max_samples samples.
- If float, then draw max_samples * X.shape[0] samples.

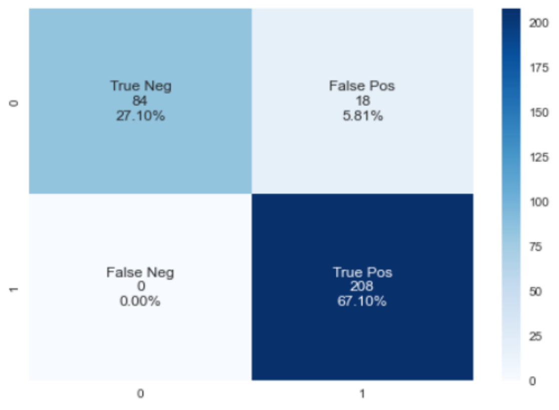
max_features*int or float, default=1.0*

The number of features to draw from X to train each base estimator (without replacement by default).

- If int, then draw max_features features.
- If float, then draw max_features * X.shape[1] features.

Result from Tuned Model:

```
Train Accuracy is : 0.9419354838709677

Test Accuracy is : 0.7686567164179104

Train ROC-AUC score is : 0.9986331070889893

Test ROC-AUC score is : 0.8422619047619048


Bagging-Tuned Classification report Train set :
              precision    recall  f1-score   support

           0       1.00      0.82      0.90       102
           1       0.92      1.00      0.96       208

    accuracy                           0.94       310
   macro avg       0.96      0.91      0.93       310
weighted avg       0.95      0.94      0.94       310


Bagging-Tuned Classification report Test set :
              precision    recall  f1-score   support

           0       0.68      0.50      0.58        42
           1       0.80      0.89      0.84        92

    accuracy                           0.77       134
   macro avg       0.74      0.70      0.71       134
weighted avg       0.76      0.77      0.76       134
```



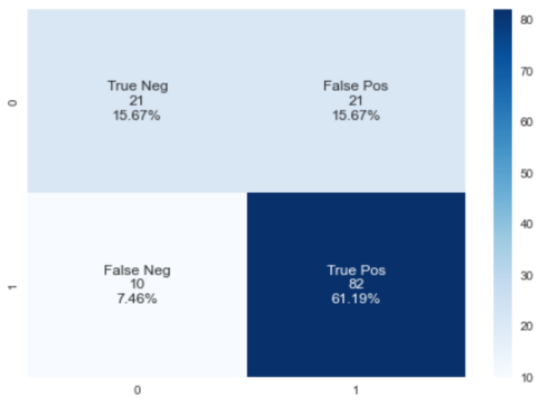


Figure 24: Classification Report and Confusion Matrices for Train & Test Set- Bagging Model-Tuned

Observation:

We got best parameters while tuning the model as {'base_estimator': DecisionTreeClassifier(), 'max_features': 0.5, 'max_samples': 0.5, 'n_estimators': 100, 'random_state': 100}.

Now, when we applied the ***GridSearchCV*** algorithm to our model which is used to select the ***best hyperparameters***, we see that the training accuracy has reduced to 94% aprox and testing accuracy has also decreased to 77% approx. The ***training accuracy*** has ***reduced*** after tuning the hyperparameters, but it has also slightly ***reduced the overfitting*** in our model. Although the ***difference between training and testing accuracy is still high*** which suggests there is still situation of overfitting, but this ***may vary for different samples***.

```
Bagging Classifier-Tuned Train: ROC AUC=0.999
Bagging classifier-Tuned test: ROC AUC=0.842
```

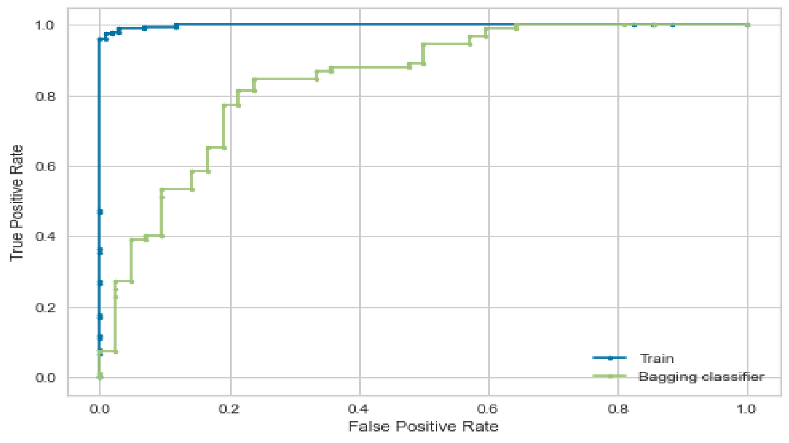


Figure 25: ROC AUC for Train & Test Set- Bagging Model-Tuned

From the above ROC AUC curve, we can see that with hyperparameter tuning, the ***accuracy*** of the Train and Test dataset has not shown any drastic drop or increment in the scores i.e. for Train it is ~ 100% and Test it is 84%. GridSearchCV has clearly not helped the model to perform well.

Model Evaluation:

Model Name	Training Data	Test Data	Train AUC-ROC Score	Test AUC-ROC Score
Bagging Model	0.983871	0.791045	0.999081	0.826346
Bagging Model-Tuned	0. .941935	0.768657	0.998633	0.842262

Note: Overall, Bagging is not showing a good-fit for our database. The model is showing major overfitting problem on using Bagging Classifier.

What is Boosting?

Boosting refers to a class of machine learning ensemble algorithms where models are added sequentially and later models in the sequence correct the predictions made by earlier models in the sequence. Boosting algorithm can track the model who failed the accurate prediction. Boosting algorithms are less affected by the overfitting problem.

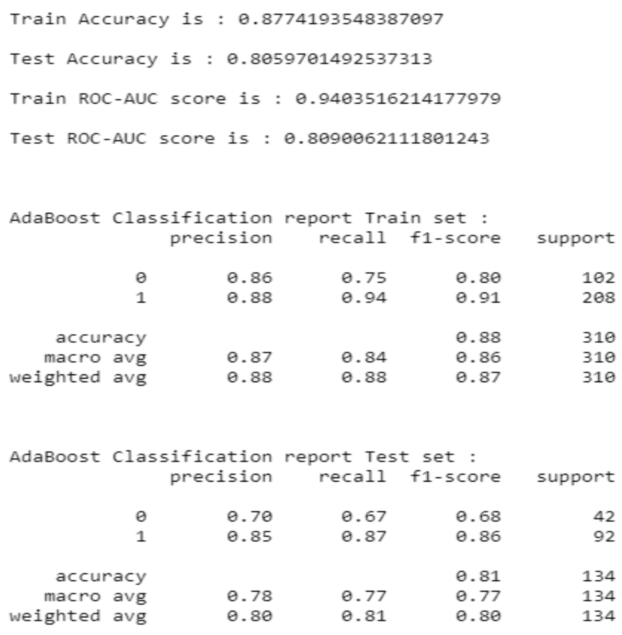
Here, we will build model with 1)AdaBoost, short for "*Adaptive Boosting*". And 2) Gradient Boosting

Let's build the **AdaBoost** model with default parameters. Let's create the AdaBoost Model using Scikit-learn. AdaBoost uses Decision Tree Classifier as default Classifier.

"The most important parameters are base_estimator, n_estimators, and learning_rate."

- **base_estimator:** It is a weak learner used to train the model. It uses DecisionTreeClassifier as default weak learner for training purpose. We can also specify different machine learning algorithms.
- **n_estimators:** Number of weak learners to train iteratively.
- **learning_rate:** It contributes to the weights of weak learners. It uses 1 as a default value.

1st Iteration Result:



```
Train Accuracy is : 0.8774193548387097

Test Accuracy is : 0.8059701492537313

Train ROC-AUC score is : 0.9403516214177979

Test ROC-AUC score is : 0.8090062111801243


AdaBoost Classification report Train set :
              precision    recall  f1-score   support

           0       0.86      0.75      0.80       102
           1       0.88      0.94      0.91       208

    accuracy                           0.88       310
   macro avg       0.87      0.84      0.86       310
weighted avg       0.88      0.88      0.87       310


AdaBoost Classification report Test set :
              precision    recall  f1-score   support

           0       0.70      0.67      0.68        42
           1       0.85      0.87      0.86        92

    accuracy                           0.81       134
   macro avg       0.78      0.77      0.77       134
weighted avg       0.80      0.81      0.80       134
```

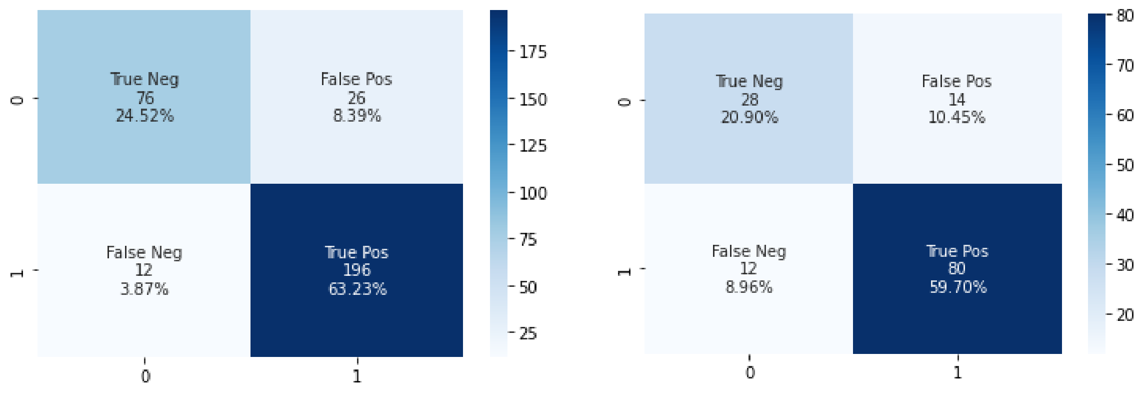


Figure 26: Classification Report and Confusion Matrices for Train & Test Set- AdaBoost Model

Observation on Confusion Matrix:

- True Positive (TP) = 196; meaning 196 public transport class data points were correctly classified by the model.
- True Negative (TN) = 76; meaning 76 private transport class data points were correctly classified by the model

- False Positive (FP) = 26; meaning 26 private transport class data points were incorrectly classified as belonging to the public transport class by the model
- False Negative (FN) = 12; meaning 12 public transport class data points were incorrectly classified as belonging to the private transport class by the model

  Likewise for Test Dataset: TP=80, TN=28, FP=14, FN=12

Observation:
We can see that the **training accuracy** of **AdaBoost Classifier** with default hyperparameters is 88**%** accuracy. Whereas when we fit the same model on **testing data**, the accuracy reduces to 81**%** approx. Clearly the **model overfits the training data,** we see that the model is behaving differently on train and test dataset.

```
AdaBoost Classifier Train: ROC AUC=0.940
AdaBoost classifier test: ROC AUC=0.809
```

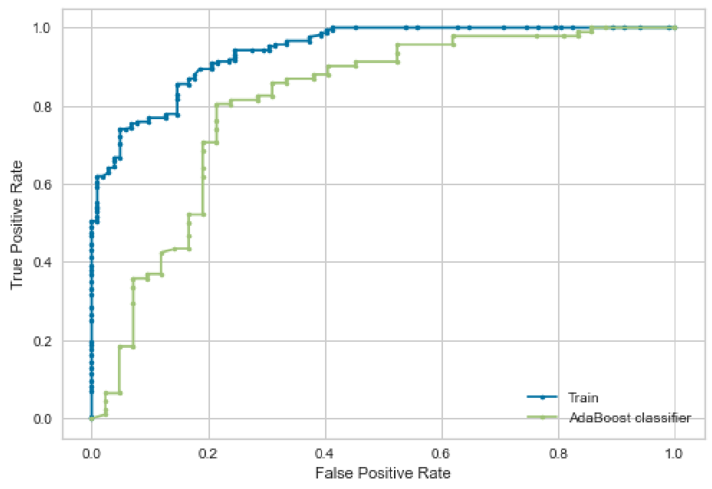


Figure 27: ROC-AUC curve of Train & Test Set- AdaBoost Model

Observations:

*From the ROC curve we can see that with default parameter the **accuracy** of the model is showing 94% for Train dataset and 81% for Test dataset.*

Lets look at the feature importance using AdaBoostClassifier.

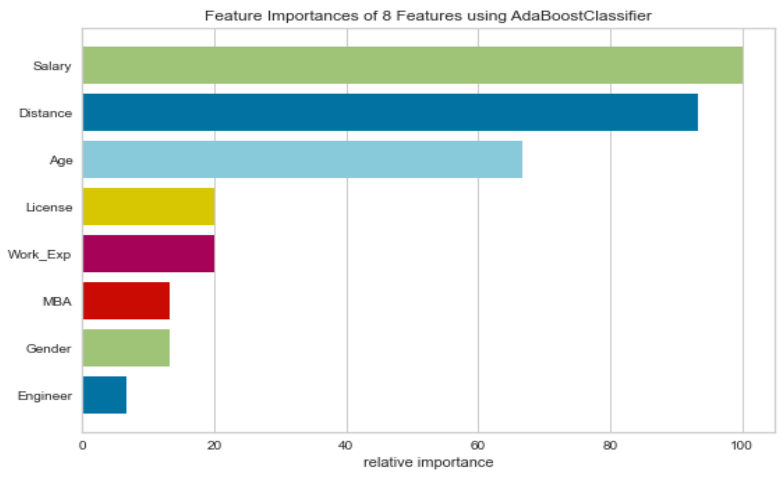


Figure 28: Feature Importance using AdaBoostClassifer

Feature: 0, Score: 0.20000	Age
Feature: 1, Score: 0.04000	Gender
Feature: 2, Score: 0.02000	Engineer
Feature: 3, Score: 0.04000	MBA
Feature: 4, Score: 0.06000	Work_Exp
Feature: 5, Score: 0.30000	Salary
Feature: 6, Score: 0.28000	Distance
Feature: 7, Score: 0.06000	License

Obesrvation: We observe from above feature importance plot that variable Salary, Distance and Age are big contributor towards choice of transport.

Hyper parameter Tuning:

While doing Hyperparameter tuning in AdaBoost we are using Different Base Learners.
Here, we will use RandomForestClassifier as a base estimator. We can use any ML learner as base estimator if it accepts sample weight such as Decision Tree.

Result from Tuned Model:

```
Train Accuracy is : 1.0

Test Accuracy is : 0.8208955223880597

Train ROC-AUC score is : 1.0

Test ROC-AUC score is : 0.8231107660455486


AdaBoost Classification report Train set :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       102
           1       1.00      1.00      1.00       208

    accuracy                           1.00       310
   macro avg       1.00      1.00      1.00       310
weighted avg       1.00      1.00      1.00       310


AdaBoost Classification report Test set :
              precision    recall  f1-score   support

           0       0.74      0.67      0.70        42
           1       0.85      0.89      0.87        92

    accuracy                           0.82       134
   macro avg       0.80      0.78      0.79       134
weighted avg       0.82      0.82      0.82       134
```

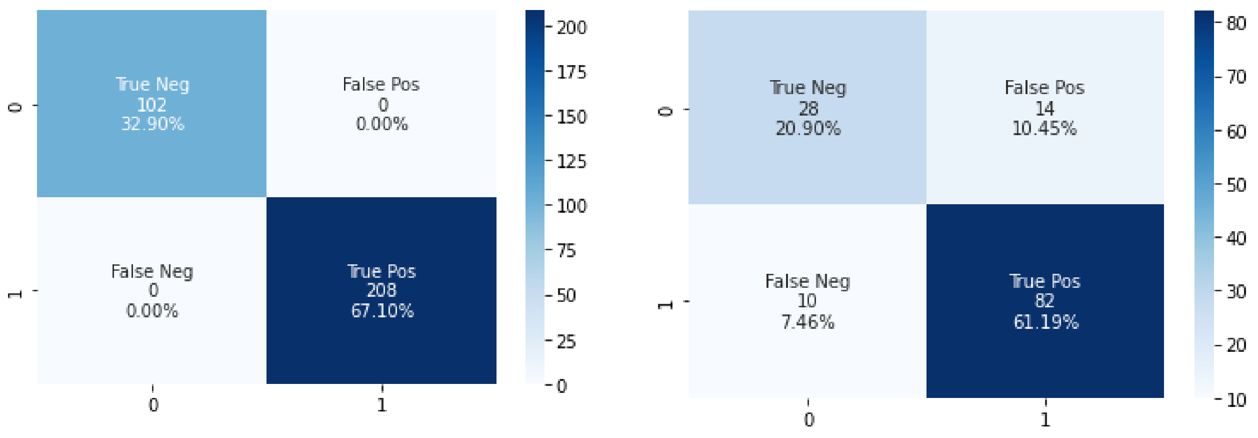


Figure 29: Classification Report and Confusion Matrices for Train & Test Set- AdaBoost Model-Tuned

Observation:

Now, when we applied the different base learner in the algorithm to our model which is used to select the ***best hyperparameters***, we see that the training accuracy has increase to 100% and testing accuracy has increased to 82% approx. Clearly the ***model has overfit the training data*** and also the ***variance is very high*** i.e. the model is behaving too good on the training data and comparatively not that well on test data. Using Hyperparameters in AdaBoost has not helped the model to perform well at all. As compared to this AdaBoost has performed well with default parameters.

```
AdaBoost-Tuned Classifier Train: ROC AUC=1.000
AdaBoost-Tuned classifier test: ROC AUC=0.823
```

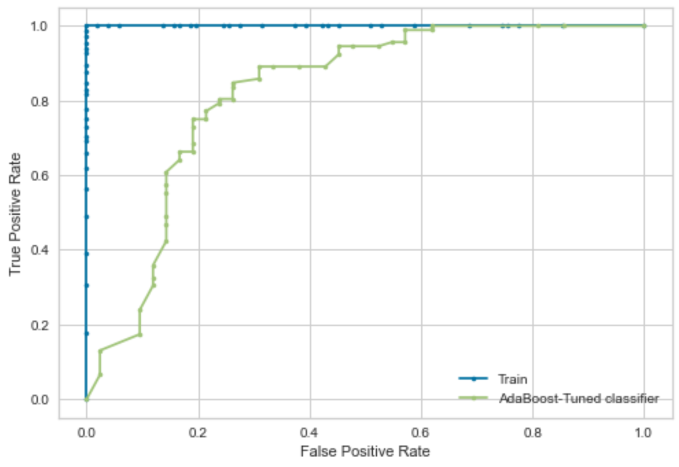


Figure 30: ROC-AUC curve of Train & Test Set- AdaBoost Model-Tuned

Observations:
From the ROC curve we can see that after hyperparameter tuning the ***accuracy*** *of the model has* ***increased*** *to 100%. Though Test score is showing ok accuracy score.*

Model Evaluation:

Model Name	Training Data	Test Data	Train AUC-ROC Score	Test AUC-ROC Score
ADaBoost Model	0.877419	0.80597	0.940352	0.809006
ADaBoost-Tuned Model	1	0.820896	1	0.823111

Conclusion: AdaBoost Classifier with default parameter showing slight overfitting but with hyperparameter it was not a good-fit Model for our Transport Dataset.

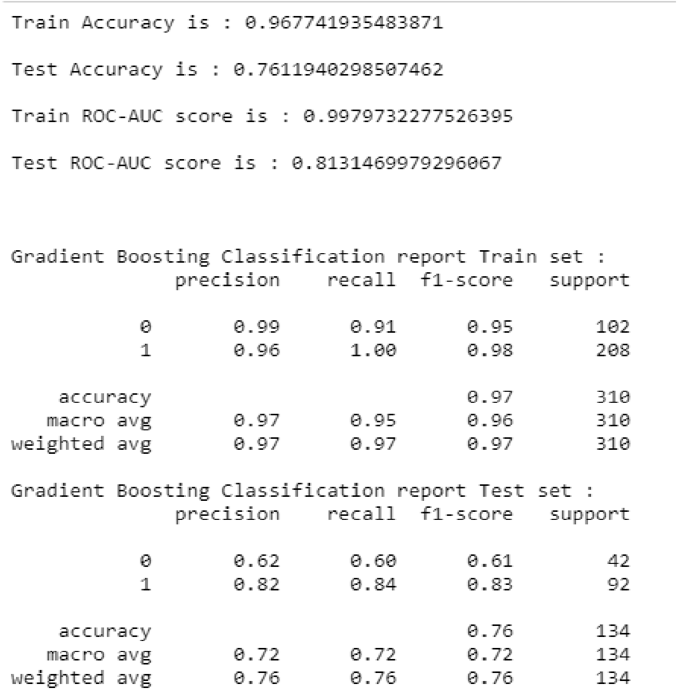
Model Building with Gradient Boosting Classifier.

1. It is extremely powerful machine learning classifier.
2. Accepts various types of inputs that make it more flexible.
3. It can be used for both regression and classification.
4. It gives you features important for the output.

Basic parameters of Gradient Boosting Algorithm

- **number of trees** (*n_estimators*; *def: 100*)
- **learning rate** (*learning_rate*; *def: 0.1*) — Scales the contribution of each tree as discussed before. There is a trade-off between learning rate and number of trees. Commonly used values of learning rate lie between 0.1 to 0.3
- **maximum depth** (*max_depth; def: 3*) — Maximum depth of each estimator. It limits the number of nodes of the decision trees

1st Iteration result using default parameters:



```
Train Accuracy is : 0.967741935483871

Test Accuracy is : 0.7611940298507462

Train ROC-AUC score is : 0.9979732277526395

Test ROC-AUC score is : 0.8131469979296067


Gradient Boosting Classification report Train set :
              precision    recall  f1-score   support

           0       0.99      0.91      0.95       102
           1       0.96      1.00      0.98       208

    accuracy                           0.97       310
   macro avg       0.97      0.95      0.96       310
weighted avg       0.97      0.97      0.97       310

Gradient Boosting Classification report Test set :
              precision    recall  f1-score   support

           0       0.62      0.60      0.61        42
           1       0.82      0.84      0.83        92

    accuracy                           0.76       134
   macro avg       0.72      0.72      0.72       134
weighted avg       0.76      0.76      0.76       134
```

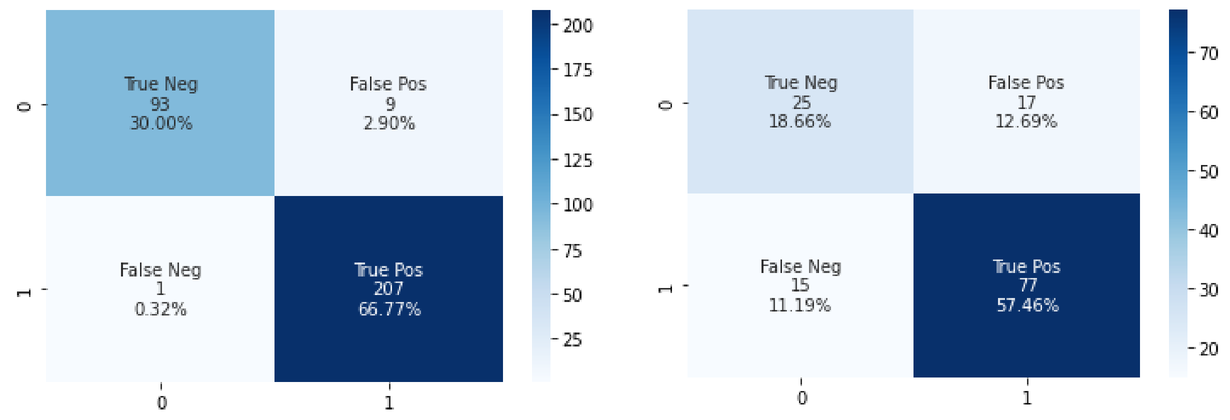


Figure 31: Classification Report and Confusion Matrices for Train & Test Set- Gradient Boosting Model

Observation: *We can see that the* **training accuracy** *of* **Gradient Boosting Classifier** *with default hyperparameters is* **97%** *accuracy. Whereas when we fit the same model on* **testing data**, *the accuracy reduces to* **76%** *approx. Clearly the* **model has overfit the training data** *and also the* **variance is very high** *i.e. the model is behaving differently for different samples.*

Observation on Confusion Matrix:

- True Positive (TP) = 207; meaning 207 public transport class data points were correctly classified by the model.
- True Negative (TN) = 93; meaning 93 private transport class data points were correctly classified by the model.
- False Positive (FP) = 9; meaning 9 private transport class data points were incorrectly classified as belonging to the public transport class by the model
- False Negative (FN) = 1; meaning 1 public transport class data points were incorrectly classified as belonging to the private transport class by the model

Likewise for Test Dataset: TP=77, TN=25, FP=17, FN=15

```
Gradient Boosting Classifier Train: ROC AUC=0.998
Gradient Boosting classifier test: ROC AUC=0.813
```

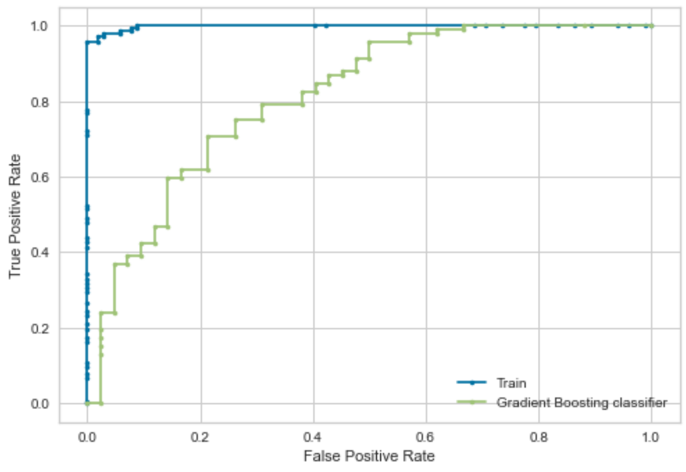


Figure 32: ROC-AUC curve of Train & Test Set-Gradient Boosting Model

Observations:

*From the ROC curve we can see that with default parameter the model is showing **accuracy** of 99% i.e. ~100% for Train dataset and 81% for Test dataset. Performing too good on the dataset.*

Let us look at the Feature Importance with Gradient Boosting algorithm

Feature: 0, Score: 0.14419	Age
Feature: 1, Score: 0.04799	Gender
Feature: 2, Score: 0.00477	Engineer
Feature: 3, Score: 0.00669	MBA
Feature: 4, Score: 0.10523	Work_Exp
Feature: 5, Score: 0.31975	Salary
Feature: 6, Score: 0.20720	Distance
Feature: 7, Score: 0.16418	License

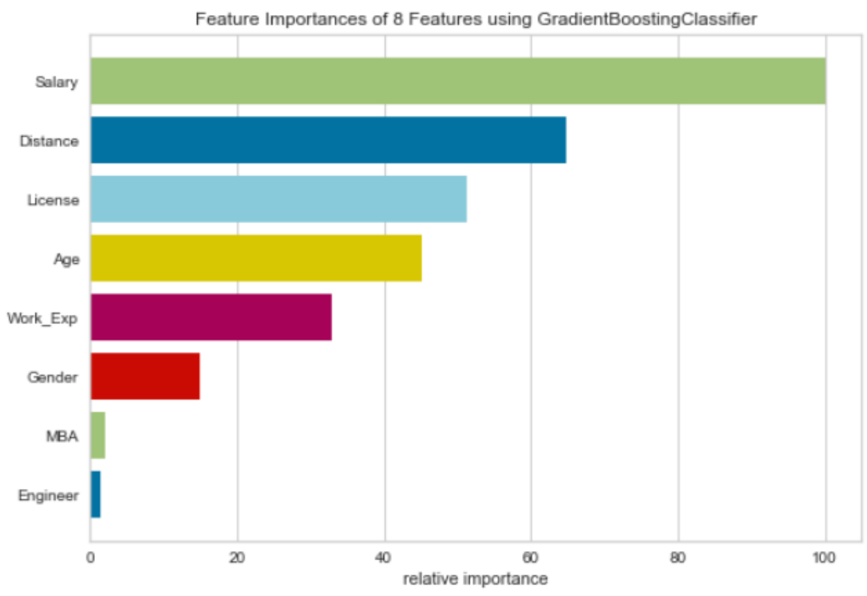


Figure 33: Feature Importance using Gradient Boosting Classifier

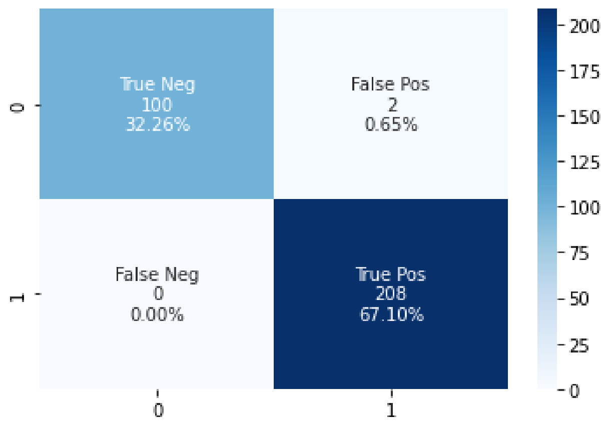
Observation: From above Importance plot we observe that Feature Salar, Distance, License, Age, and to an extent Work_Exp is contributing towards selection of Transport.

HyperParameter Tuning in Gradient Boosting

Tuning n_estimators and Learning rate
n_estimators is the number of trees (weak learners) that we want to add in the model. There are no optimum values for learning rate as low values always work better, given that we train on sufficient number of trees. A high number of trees can be computationally expensive that's why I have taken few numbers of trees here.

Result from Tuned Model:

```
Train Accuracy is : 0.9935483870967742

Test Accuracy is : 0.7761194029850746

Train ROC-AUC score is : 1.0

Test ROC-AUC score is : 0.8535196687370601


Gradient Boosting Tuned Classification report Train set :
              precision    recall  f1-score   support

           0       1.00      0.98      0.99       102
           1       0.99      1.00      1.00       208

    accuracy                           0.99       310
   macro avg       1.00      0.99      0.99       310
weighted avg       0.99      0.99      0.99       310

Gradient Boosting Tuned Classification report Test set :
              precision    recall  f1-score   support

           0       0.65      0.62      0.63        42
           1       0.83      0.85      0.84        92

    accuracy                           0.78       134
   macro avg       0.74      0.73      0.74       134
weighted avg       0.77      0.78      0.77       134
```



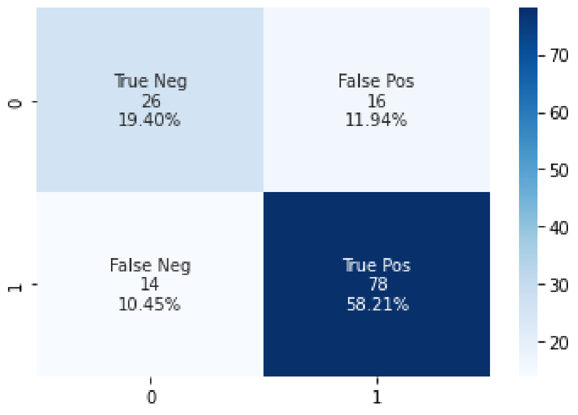


Figure 34: Classification Report and Confusion Matrices for Train & Test Set- Gradient Boosting Model-Tuned

Observation: Output showing best parameter as {'learning_rate': 0.05, 'max_depth': 5, 'min_samples_split': 12, 'n_estimators': 100, 'random_state': 100}

*Now, when we applied the **GridSearchCV** algorithm to our model which has used the above **best hyperparameters**, We can see that the* **training accuracy** *of* **Gradient Boosting Classifier** *with hyperparameters is* **99%** *accuracy. Whereas when we fit the same model* *on* **testing data**, *the accuracy reduces to* **78%** *approx. Clearly the* **model has overfit the training data** *and also the* **variance is very high** *i.e. the model is behaving differently for different samples.*

```
Gradient Boosting Classifier-Tuned Train: ROC AUC=1.000
Gradient Boosting classifier-Tuned test: ROC AUC=0.852
```

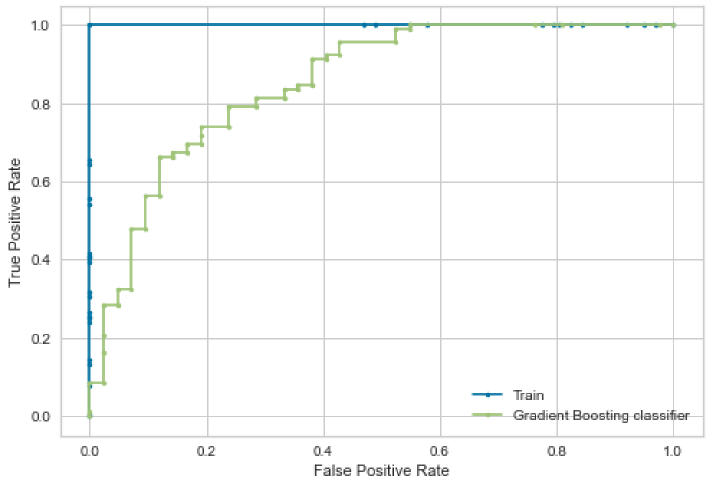


Figure 35: ROC-AUC curve of Train & Test Set-Gradient Boosting Model-Tuned

From above ROC curve AUC curve, we don't observe a good performance by the Model. After tuning with hyperparameters we are getting Train accuracy as ~100% and Test accuracy as 85%.

Model Evaluation:

Model Name	Training Data	Test Data	Train AUC-ROC Score	Test AUC-ROC Score
Gradient Boosting Model	0.967742	0.761194	0.997973	0.813147
Gradient Boosting Model-Tuned	0.993548	0.776119	1	0.85352

Conclusion: Gradient Boosting was not a good-fit Model for our Transport Dataset.

1.7 Performance Metrics: Check the performance of Predictions on Train and Test sets using Accuracy, Confusion Matrix, Plot ROC curve and get ROC_AUC score for each model. Final Model - Compare all models on the basis of the performance metrics in a structured tabular manner. Describe on which model is best/optimized.

Performance Metrics:

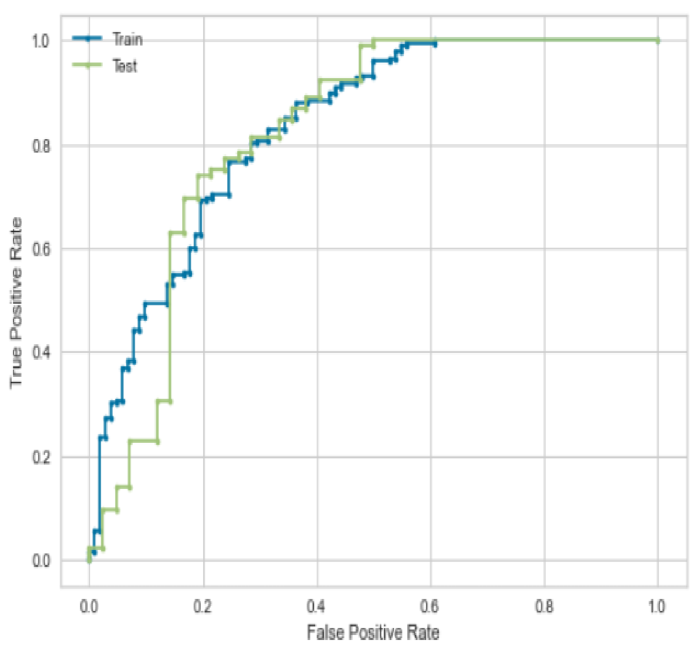
Performance metrics based on following methods:

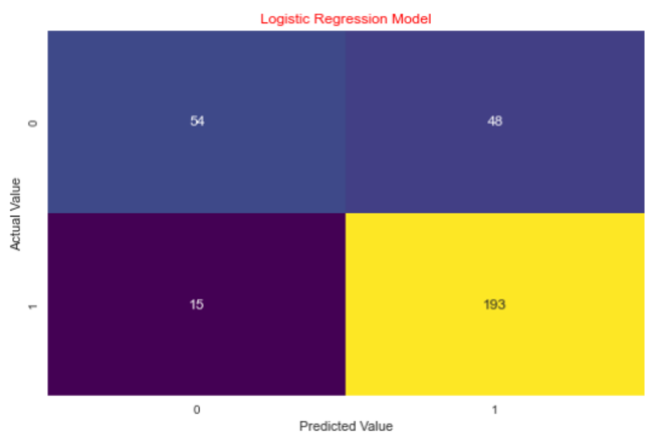
- Model Accuracy.
- Confusion Matrix.
- Classification Report.
- Area Under ROC curve and AUC score.

Model 1: Logistic Regression

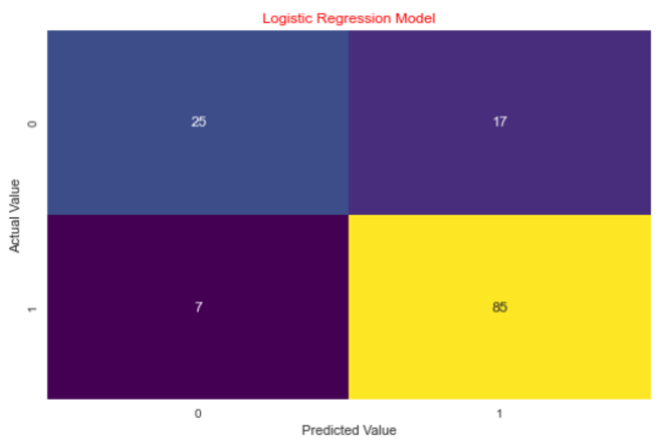
```
Train Accuracy is : 0.7967741935483871

Test Accuracy is : 0.8208955223880597

Train ROC-AUC score is : 0.8319193061840121

Test ROC-AUC score is : 0.818840579710145


Logistic Regression Classification report Train set :
              precision    recall  f1-score   support

           0       0.78      0.53      0.63       102
           1       0.80      0.93      0.86       208

    accuracy                           0.80       310
   macro avg       0.79      0.73      0.75       310
weighted avg       0.79      0.80      0.78       310


Logistice Regression Classification report Test set :
              precision    recall  f1-score   support

           0       0.78      0.60      0.68        42
           1       0.83      0.92      0.88        92

    accuracy                           0.82       134
   macro avg       0.81      0.76      0.78       134
weighted avg       0.82      0.82      0.81       134
```

```
Logistic Regression Classifier Train: ROC AUC=0.832
Logistic Regression classifier test: ROC AUC=0.819
```





Confusion Matrix-Train

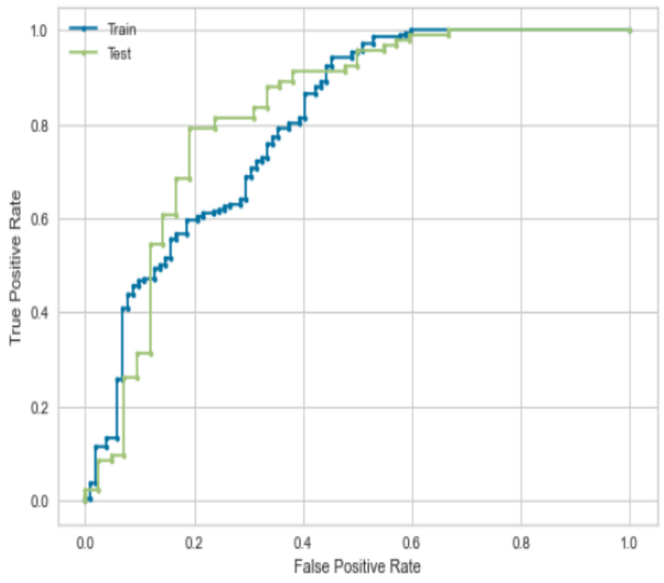


Confusion Matrix-Test

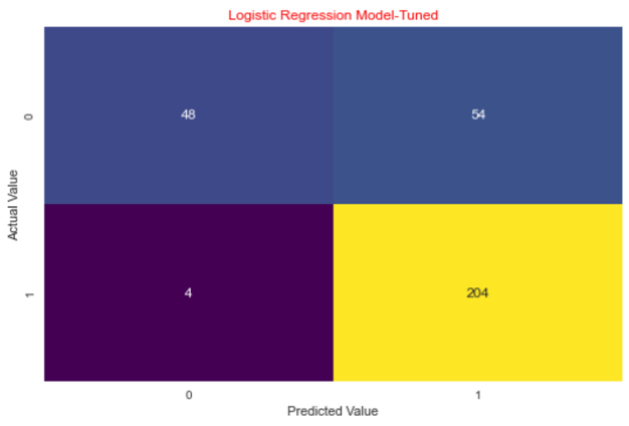
Model 2: Logistic Regression-Tuned

```
Train Accuracy is : 0.8129032258064516

Test Accuracy is : 0.7985074626865671

Train ROC-AUC score is : 0.8023190045248869

Test ROC-AUC score is : 0.8253105590062112


Logistic Regression-Tuned Classification report Train set :
              precision    recall  f1-score   support

           0       0.92      0.47      0.62       102
           1       0.79      0.98      0.88       208

    accuracy                           0.81       310
   macro avg       0.86      0.73      0.75       310
weighted avg       0.83      0.81      0.79       310


Logistice Regression-Tuned Classification report Test set :
              precision    recall  f1-score   support

           0       0.86      0.43      0.57        42
           1       0.79      0.97      0.87        92

    accuracy                           0.80       134
   macro avg       0.82      0.70      0.72       134
weighted avg       0.81      0.80      0.78       134
```

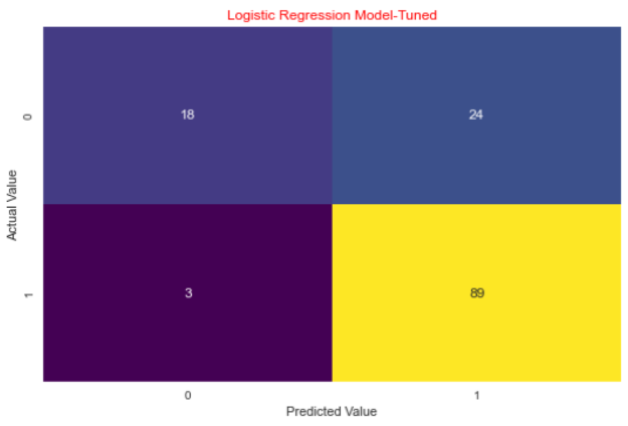
```
Logistic Regression-Tuned Classifier Train: ROC AUC=0.802
Logistic Regression-Tuned classifier test: ROC AUC=0.825
```



Confusion Matrix-Train



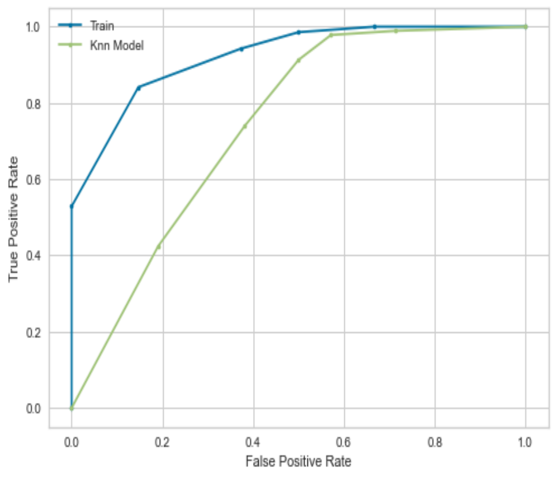
Confusion Matrix-Test



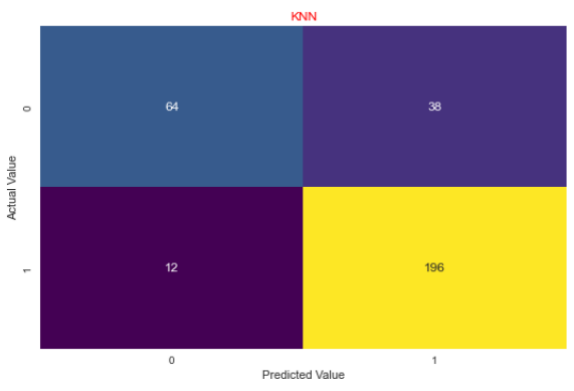
Model 3: KNN:

```
Train Accuracy is : 0.8387096774193549

Test Accuracy is : 0.7835820895522388

Train ROC-AUC score is : 0.923006221719457

Test ROC-AUC score is : 0.7553053830227743


Classification report Train set :
              precision    recall  f1-score   support

           0       0.83      0.64      0.72       102
           1       0.84      0.94      0.89       208

    accuracy                           0.84       310
   macro avg       0.84      0.79      0.80       310
weighted avg       0.84      0.84      0.83       310
Classification report Test set :
              precision    recall  f1-score   support

           0       0.71      0.52      0.60        42
           1       0.81      0.90      0.85        92

    accuracy                           0.78       134
   macro avg       0.76      0.71      0.73       134
weighted avg       0.78      0.78      0.77       134
```

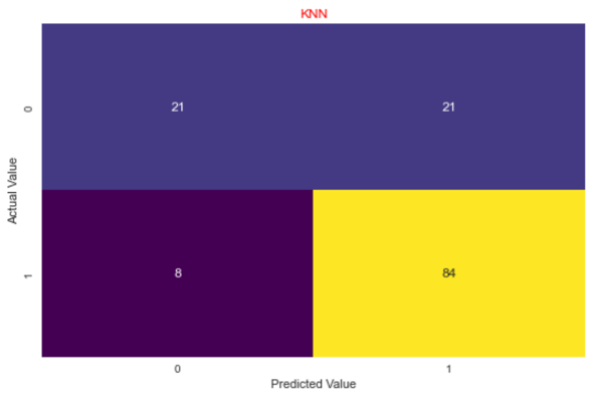
```
Knn Model Train: ROC AUC=0.924
Knn Model test: ROC AUC=0.742
```



Confusion Matrix-Train



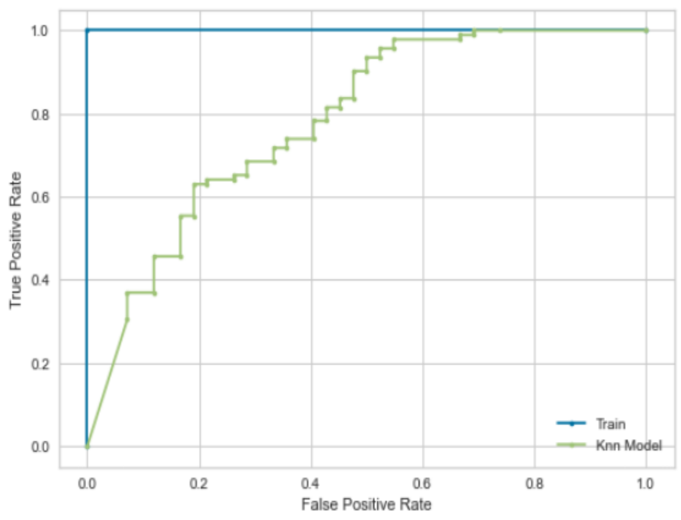
Confusion Matrix-Test



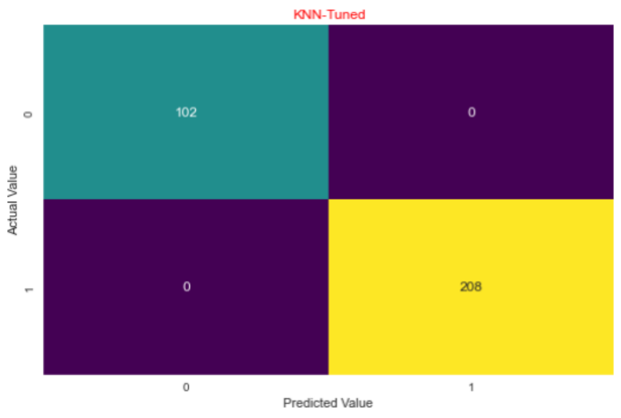
Model 4: KNN-Tuned

```
Train Accuracy is : 1.0

Test Accuracy is : 0.7835820895522388

Train ROC-AUC score is : 1.0

Test ROC-AUC score is : 0.7833850931677019


Classification report Train set :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       102
           1       1.00      1.00      1.00       208

    accuracy                           1.00       310
   macro avg       1.00      1.00      1.00       310
weighted avg       1.00      1.00      1.00       310

Classification report Test set :
              precision    recall  f1-score   support

           0       0.72      0.50      0.59        42
           1       0.80      0.91      0.85        92

    accuracy                           0.78       134
   macro avg       0.76      0.71      0.72       134
weighted avg       0.78      0.78      0.77       134
```

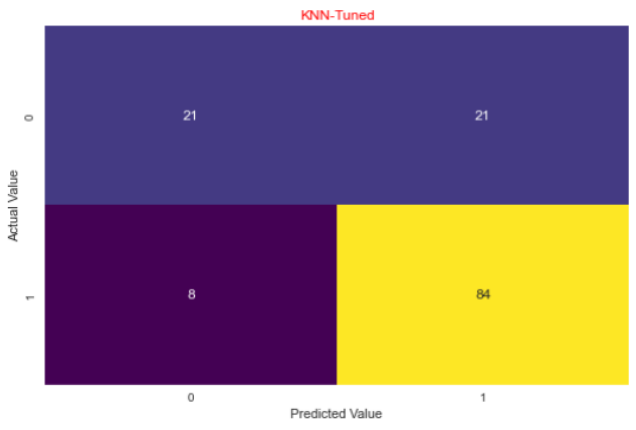
```
Knn Model Train: ROC AUC=1.000
Knn Model test: ROC AUC=0.783
```



Confusion Matrix-Train



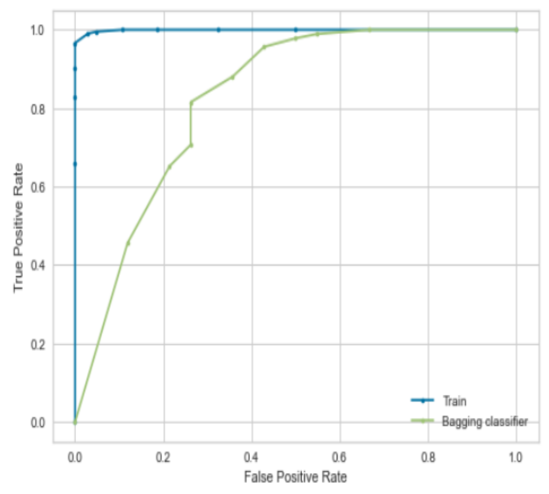
Confusion Matrix-Test



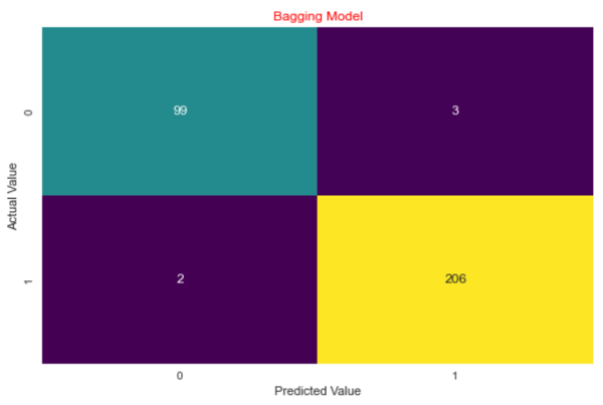
Model 5: Bagging

```
Train Accuracy is : 0.9838709677419355

Test Accuracy is : 0.7910447761194029

Train ROC-AUC score is : 0.9990808823529411

Test ROC-AUC score is : 0.8263457556935818


Classification report Train set :
              precision    recall  f1-score   support

           0       0.98      0.97      0.98       102
           1       0.99      0.99      0.99       208

    accuracy                           0.98       310
   macro avg       0.98      0.98      0.98       310
weighted avg       0.98      0.98      0.98       310


Classification report Test set :
              precision    recall  f1-score   support

           0       0.65      0.74      0.69        42
           1       0.87      0.82      0.84        92

    accuracy                           0.79       134
   macro avg       0.76      0.78      0.77       134
weighted avg       0.80      0.79      0.79       134
```

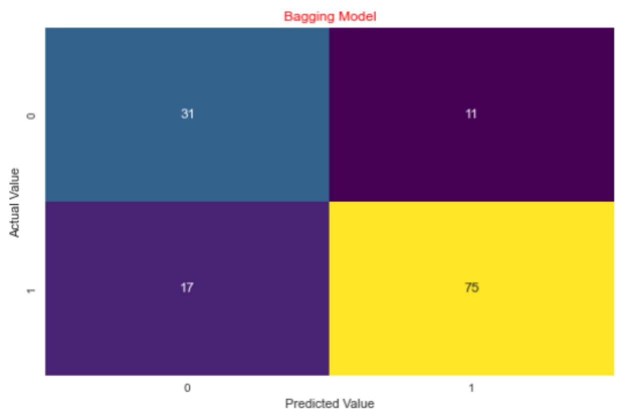
```
Bagging Classifier Train: ROC AUC=0.999
Bagging classifier test: ROC AUC=0.826
```



Confusion Matrix-Train



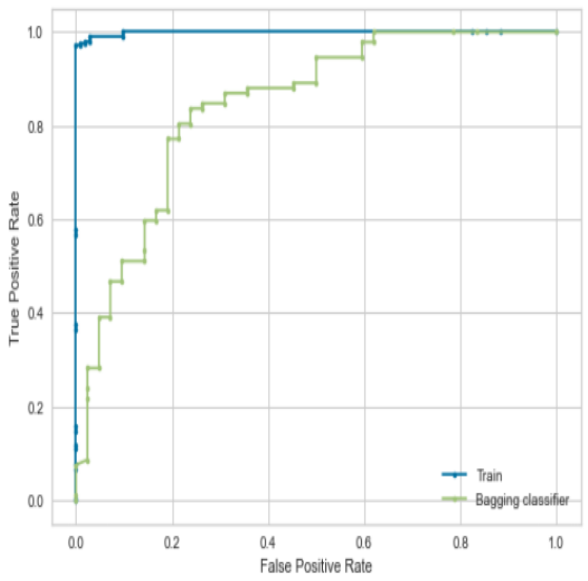
Confusion Matrix-Test



Model 6: Bagging-Tuned

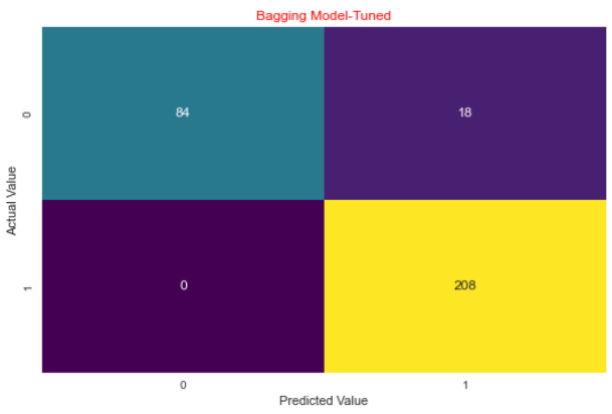
```
Train Accuracy is : 0.9419354838709677

Test Accuracy is : 0.7686567164179104

Train ROC-AUC score is : 0.9986331070889893

Test ROC-AUC score is : 0.8422619047619048


Bagging-Tuned Classification report Train set :
              precision    recall  f1-score   support

           0       1.00      0.82      0.90       102
           1       0.92      1.00      0.96       208

    accuracy                           0.94       310
   macro avg       0.96      0.91      0.93       310
weighted avg       0.95      0.94      0.94       310


Bagging-Tuned Classification report Test set :
              precision    recall  f1-score   support

           0       0.68      0.50      0.58        42
           1       0.80      0.89      0.84        92

    accuracy                           0.77       134
   macro avg       0.74      0.70      0.71       134
weighted avg       0.76      0.77      0.76       134
```

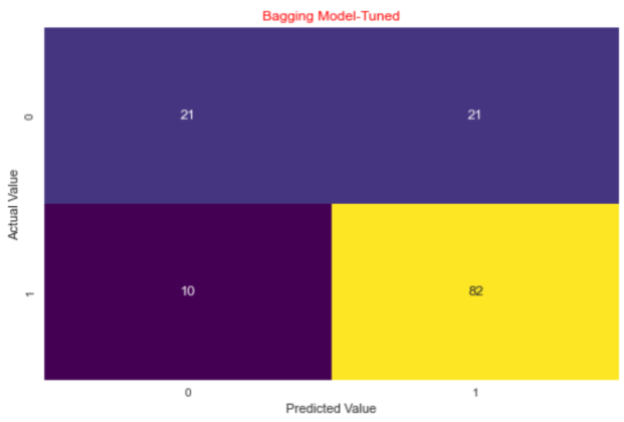
```
Bagging Classifier-Tuned Train: ROC AUC=0.999
Bagging classifier-Tuned test: ROC AUC=0.842
```



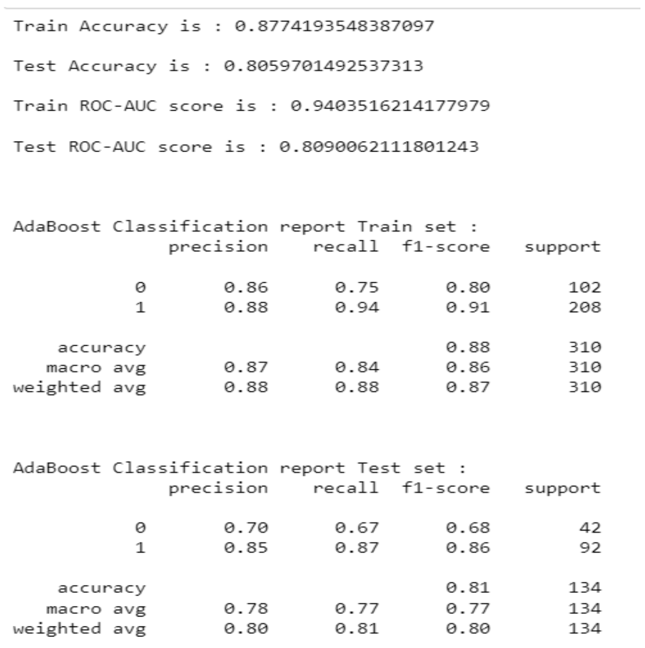
Confusion Matrix-Train

Confusion Matrix-Test



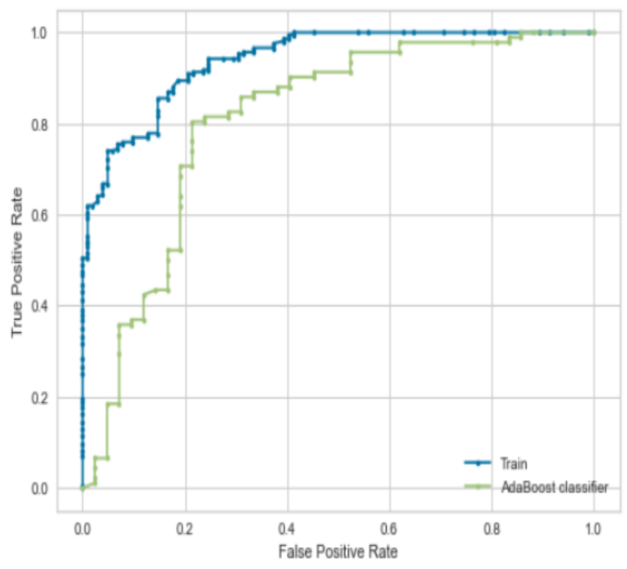


Model 7: AdaBoost



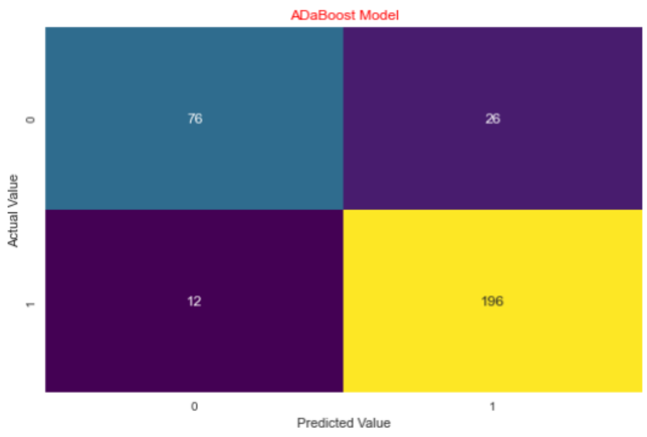
```
Train Accuracy is : 0.8774193548387097

Test Accuracy is : 0.8059701492537313

Train ROC-AUC score is : 0.9403516214177979

Test ROC-AUC score is : 0.8090062111801243


AdaBoost Classification report Train set :
              precision    recall  f1-score   support

           0       0.86      0.75      0.80       102
           1       0.88      0.94      0.91       208

    accuracy                           0.88       310
   macro avg       0.87      0.84      0.86       310
weighted avg       0.88      0.88      0.87       310


AdaBoost Classification report Test set :
              precision    recall  f1-score   support

           0       0.70      0.67      0.68        42
           1       0.85      0.87      0.86        92

    accuracy                           0.81       134
   macro avg       0.78      0.77      0.77       134
weighted avg       0.80      0.81      0.80       134
```

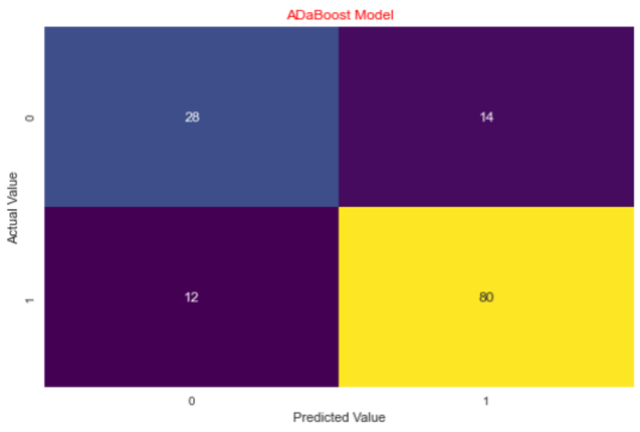
```
AdaBoost Classifier Train: ROC AUC=0.940
AdaBoost classifier test: ROC AUC=0.809
```



Confusion Matrix-Train

Confusion Matrix-Test

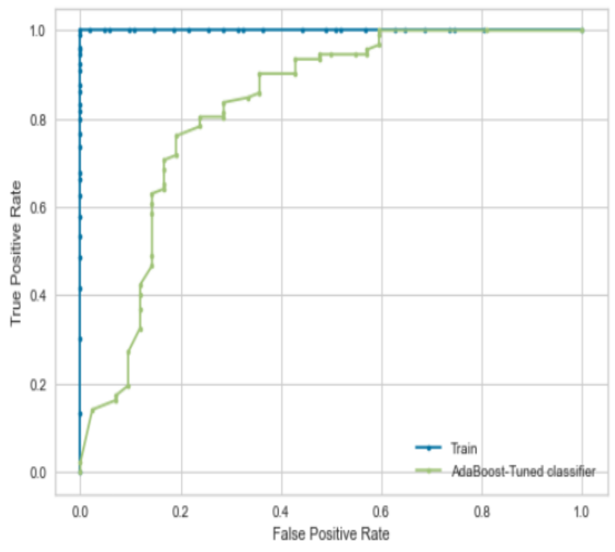




Model 8: AdaBoost-Tuned

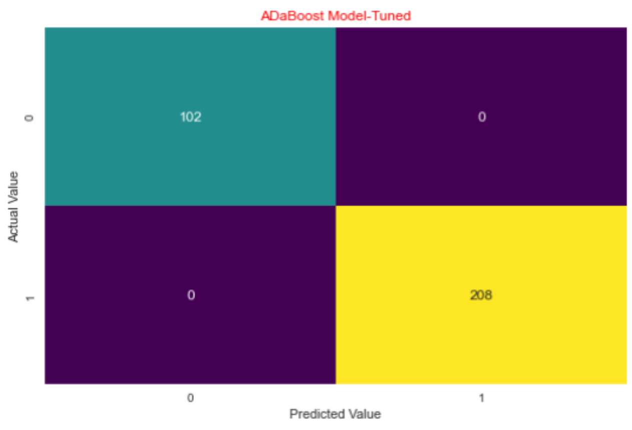
```
Train Accuracy is : 1.0

Test Accuracy is : 0.8134328358208955

Train ROC-AUC score is : 0.9999999999999999

Test ROC-AUC score is : 0.8255693581780539


AdaBoost Classification report Train set :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       102
           1       1.00      1.00      1.00       208

    accuracy                           1.00       310
   macro avg       1.00      1.00      1.00       310
weighted avg       1.00      1.00      1.00       310


AdaBoost Classification report Test set :
              precision    recall  f1-score   support

           0       0.73      0.64      0.68        42
           1       0.85      0.89      0.87        92

    accuracy                           0.81       134
   macro avg       0.79      0.77      0.78       134
weighted avg       0.81      0.81      0.81       134
```

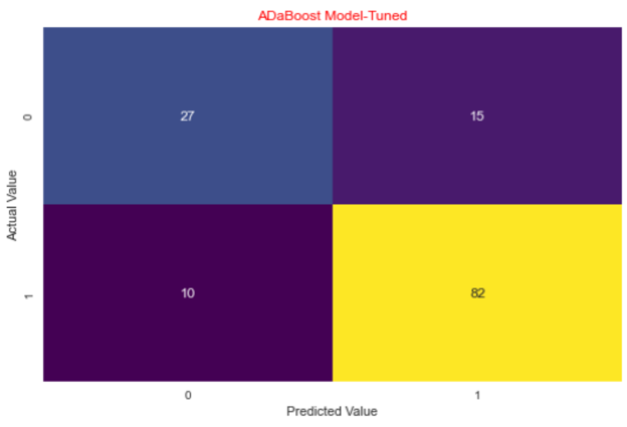
```
AdaBoost-Tuned Classifier Train: ROC AUC=1.000
AdaBoost-Tuned classifier test: ROC AUC=0.826
```



Confusion Matrix-Train

Confusion Matrix-Test

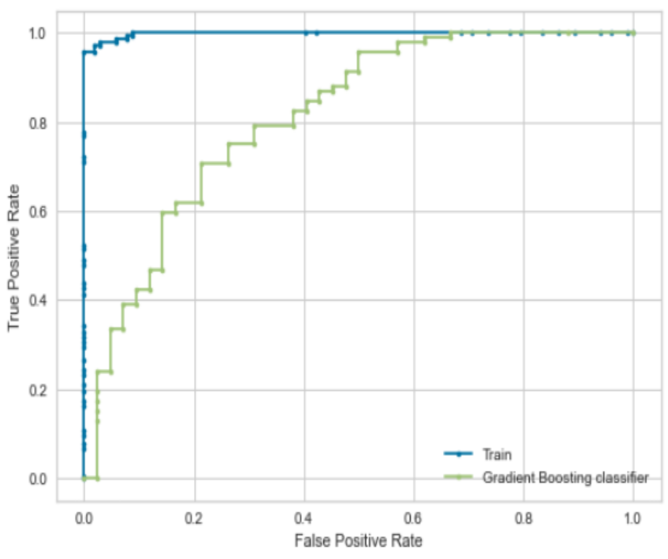




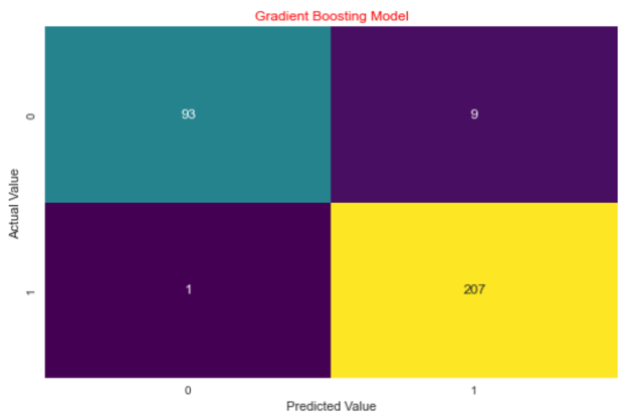
Model 9: Gradient Boosting

```
Train Accuracy is : 0.967741935483871

Test Accuracy is : 0.7611940298507462

Train ROC-AUC score is : 0.9979732277526395

Test ROC-AUC score is : 0.8126293995859213


Gradient Boosting Classification report Train set :
              precision    recall  f1-score   support

           0       0.99      0.91      0.95       102
           1       0.96      1.00      0.98       208

    accuracy                           0.97       310
   macro avg       0.97      0.95      0.96       310
weighted avg       0.97      0.97      0.97       310

Gradient Boosting Classification report Test set :
              precision    recall  f1-score   support

           0       0.62      0.60      0.61        42
           1       0.82      0.84      0.83        92

    accuracy                           0.76       134
   macro avg       0.72      0.72      0.72       134
weighted avg       0.76      0.76      0.76       134
```

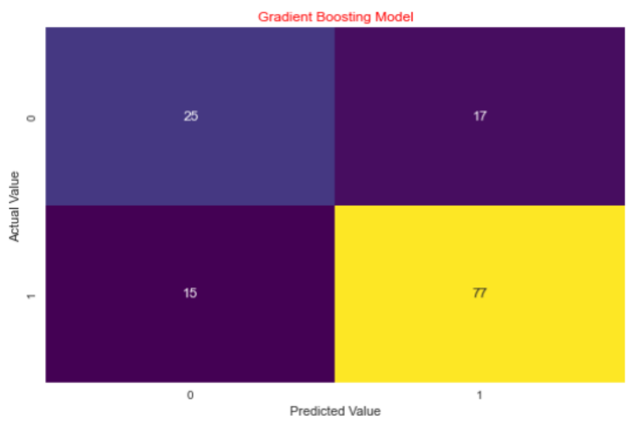
```
Gradient Boosting Classifier Train: ROC AUC=0.998
Gradient Boosting classifier test: ROC AUC=0.813
```



Confusion Matrix-Train



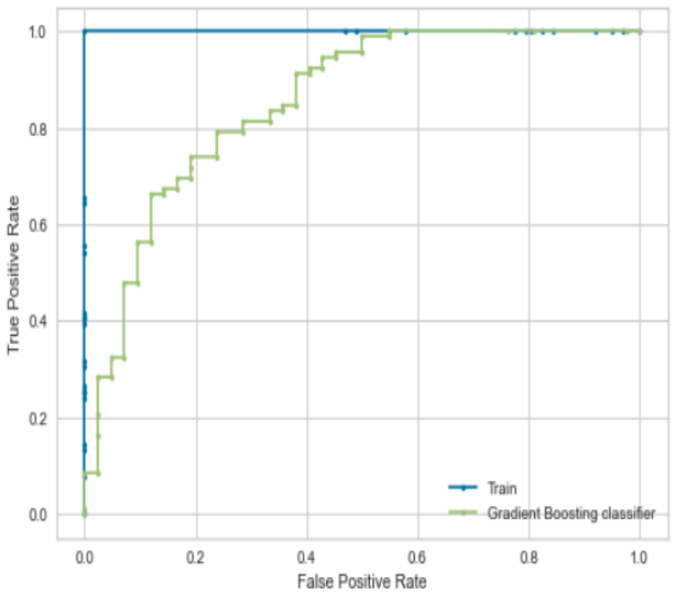
Confusion Matrix-Test



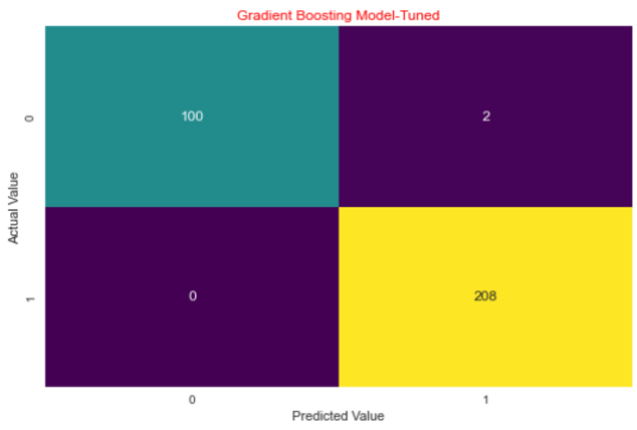
Model 10: Gradient Boosting-Tuned

```
Train Accuracy is : 0.9935483870967742

Test Accuracy is : 0.7835820895522388

Train ROC-AUC score is : 1.0

Test ROC-AUC score is : 0.8530020703933747


Gradient Boosting Tuned Classification report Train set :
              precision    recall  f1-score   support

           0       1.00      0.98      0.99       102
           1       0.99      1.00      1.00       208

    accuracy                           0.99       310
   macro avg       1.00      0.99      0.99       310
weighted avg       0.99      0.99      0.99       310

Gradient Boosting Tuned Classification report Test set :
              precision    recall  f1-score   support

           0       0.67      0.62      0.64        42
           1       0.83      0.86      0.84        92

    accuracy                           0.78       134
   macro avg       0.75      0.74      0.74       134
weighted avg       0.78      0.78      0.78       134
```

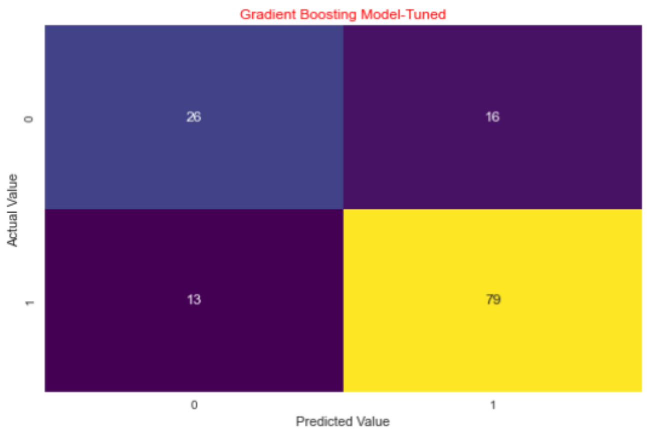
```
Gradient Boosting Classifier-Tuned Train: ROC AUC=1.000
Gradient Boosting classifier-Tuned test: ROC AUC=0.853
```



Confusion Matrix-Train



Confusion Matrix-Test



Model Comparison- There are 10 different type of model we build with scaled dataset. The basis on which models are evaluated are known as performance metrics, which are:

- Accuracy
- Recall
- Precision
- F1score
- ROC-AUC score

Model with default parameters:

	Logistic Regression-Train	Logistic Regression-Test	KNN-Train	KNN-Test	Bagging-Train	Bagging-Test	ADaBoost-Train	ADaBoost-Test	Gradient Boosting-Train	Gradient Boosting-Test
Accuracy	0.796774	0.820896	0.83871	0.783582	0.983871	0.791045	0.877419	0.80597	0.967742	0.761194
Recall	0.927885	0.923913	0.942308	0.913043	0.990385	0.815217	0.942308	0.869565	0.995192	0.836957
Precision	0.80083	0.833333	0.837607	0.8	0.985646	0.872093	0.882883	0.851064	0.958333	0.819149
F1 score	0.859688	0.876289	0.886878	0.852792	0.98801	0.842697	0.911628	0.860215	0.976415	0.827957
ROC_AUC score	0.83	0.82	0.92	0.74	1	0.83	0.94	0.81	1	0.81

Table 6: Default Model Comparison of Train and Test Set

From Above Comparison:

- Based on Accuracy-Logistic Regression, KNN and AdaBoost Model performs better than other models.
- Based on Recall- Logistic Regression performs better than other models.
- Based on Precision- Logistic Regression performs better than other models.
- Based on F1 score- Logistic Regression performs better than other models.
- Based on ROC-AUC score-Again Logistic Regression performs better than other models.

Note: Though in few models AUC score is 100% i.e. model is predicting 100% correct but unable to get that 100% prediction in test model. Hence selecting the model whose AUC score of train and test are not differing much to each other.

Model with HyperParameters:

	Logistic Regression Tuned-Train	Logistic Regression Tuned-Test	KNN Tuned-Train	KNN Tuned-Test	Bagging Tuned-Train	Bagging Tuned-Test	ADaBoost Tuned-Train	ADaBoost Tuned-Test	Gradient Boosting Tuned-Train	Gradient Boosting Tuned-Test
Accuracy	0.812903	0.798507	1	0.783582	0.941935	0.768657	1	0.813433	0.993548	0.783582
Recall	0.980769	0.967391	1	0.913043	1	0.891304	1	0.891304	1	0.858696
Precision	0.790698	0.787611	1	0.8	0.920354	0.796117	1	0.845361	0.990476	0.831579
F1 score	0.875536	0.868293	1	0.852792	0.958525	0.841026	1	0.867725	0.995215	0.84492
ROC_AUC score	0.800000	0.83	1	0.78	1	0.84	1	0.83	1	0.85

Table 7: Hyperparameter-Tuned Model Comparison of Train and Test Set

From Above Comparison:

- Based on Accuracy- Logistic Regression with Hyperparameters has performed better than other models.
- Based on Recall- Logistic Regression with Hyperparameters has performed better than other models.
- Based on Precision- Logistic Regression with Hyperparameters has performed better than other models.
- Based on F1 score- Logistic Regression with Hyperparameters has performed better than other models.
- Based on ROC-AUC score-Again Logistic Regression with Hyperparameters has performed better than other models.
- Note: Though in few models AUC score is 100% i.e. model is predicting 100% correct but unable to get that 100% prediction in test model. Hence selecting the model whose AUC score of train and test are not differing much to each other.

Observation: From above 2 tables, it can be observed that using so many models with default parameter and hyperparameter, all the model has not performed to well on our dataset. Some worked great on Train set but failed to work great on test set. And has quite a gap in their performance metrics values.

Though, few model did performed well, like Logistic Regression (with default parameter & hyperparameters), KNN(default parameters) and AdaBoost(default parameters).

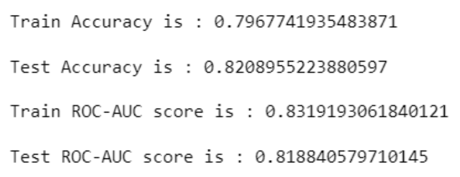
Among these, the model which perform better is……

Final Model- LOGISTIC REGRESSION.

1.8) Based on your analysis and working on the business problem, detail out appropriate insights and recommendations to help the management solve the business objective.

Based on our Analysis we have selected Logistic Regression as the best model for our dataset. We got below result using default parameter and hyperparameter:

Logistic Regression Model with default parameters:



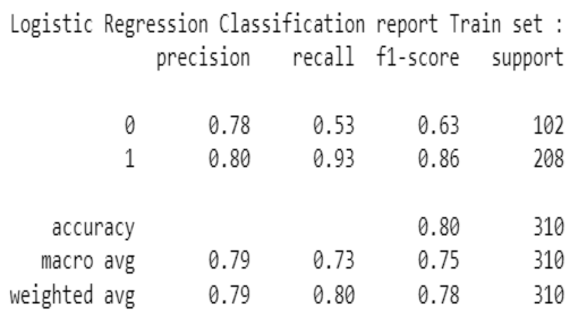
```
Train Accuracy is : 0.7967741935483871

Test Accuracy is : 0.8208955223880597

Train ROC-AUC score is : 0.8319193061840121

Test ROC-AUC score is : 0.818840579710145
```

Importance feature Identified by the model are:

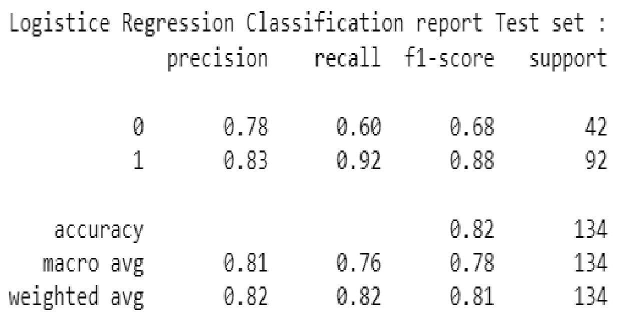
Feature: Age, Score: 0.93857
Feature: Gender, Score: 1.08562
Feature: Engineer, Score: -0.27267
Feature: MBA, Score: 0.34690
Feature: Work_Exp, Score: -0.59732
Feature: Salary, Score: -0.67838
Feature: Distance, Score: -0.78262
Feature: License, Score: -1.85525

License, Age, Gender and Distance showing high probability towards predicting employee will be opting for public transportation. And the rest features showing comparatively less effect towards probability of employees opting for public transportation.

If we look at the classification report of the model:



```
Logistic Regression Classification report Train set :
              precision    recall  f1-score   support

           0       0.78      0.53      0.63       102
           1       0.80      0.93      0.86       208

    accuracy                           0.80       310
   macro avg       0.79      0.73      0.75       310
weighted avg       0.79      0.80      0.78       310
```



```
Logistice Regression Classification report Test set :
              precision    recall  f1-score   support

           0       0.78      0.60      0.68        42
           1       0.83      0.92      0.88        92

    accuracy                           0.82       134
   macro avg       0.81      0.76      0.78       134
weighted avg       0.82      0.82      0.81       134
```

Inference for the Train Set:

•For predicting employees who will opt for private transportation.

oPrecision (78%) –78% of prediction of employees who will opt for private transport are correct.

oRecall (53%) –53% of employees who will opt for private transport are correctly predicted.

•For predicting employees who opt for public transport

oPrecision (80%) –80% of prediction of employees who will opt for public transport are correct

oRecall (93%) –93% of employees who will opt for public transport are correctly predicted

Note: Overall accuracy of the model- 80% of the total prediction are correct.

Inference for the Test Set:

•For predicting employees who will opt for private transportation.

oPrecision (78%) –78% of prediction of employees who will opt for private transport are correct.

oRecall (60%) –60% of employees who will opt for private transport are correctly predicted.

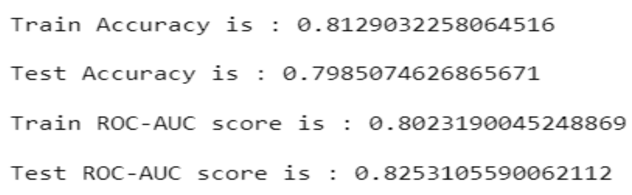
•For predicting employees who opt for public transport

oPrecision (83%) –83% of prediction of employees who will opt for public transport are correct

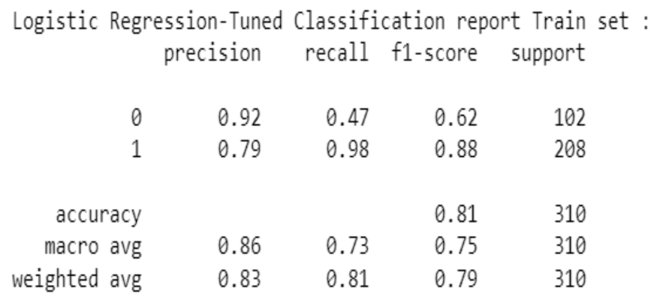
oRecall (92%) –92% of employees who will opt for public transport are correctly predicted

Note: Overall accuracy of the model- 82% of the total prediction are correct.

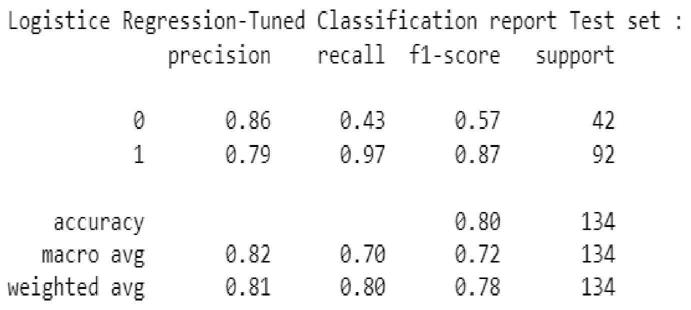
Logistic Regression Model with hyperparameters:



```
Train Accuracy is : 0.8129032258064516

Test Accuracy is : 0.7985074626865671

Train ROC-AUC score is : 0.8023190045248869

Test ROC-AUC score is : 0.8253105590062112
```



```
Logistic Regression-Tuned Classification report Train set :
              precision    recall  f1-score   support

           0       0.92      0.47      0.62       102
           1       0.79      0.98      0.88       208

    accuracy                           0.81       310
   macro avg       0.86      0.73      0.75       310
weighted avg       0.83      0.81      0.79       310
```



```
Logistice Regression-Tuned Classification report Test set :
              precision    recall  f1-score   support

           0       0.86      0.43      0.57        42
           1       0.79      0.97      0.87        92

    accuracy                           0.80       134
   macro avg       0.82      0.70      0.72       134
weighted avg       0.81      0.80      0.78       134
```

Inference for the Train Set:

•For predicting employees who will opt for private transportation.

oPrecision (92%) –92% of prediction of employees who will opt for private transport are correct.

oRecall (47%) –47% of employees who will opt for private transport are correctly predicted.

•For predicting employees who opt for public transport

oPrecision (79%) –79% of prediction of employees who will opt for public transport are correct

oRecall (98%) –98% of employees who will opt for public transport are correctly predicted

Note: Overall accuracy of the model- 81% of the total prediction are correct.

Inference for the Test Set:

•For predicting employees who will opt for private transportation.

oPrecision (86%) –86% of prediction of employees who will opt for private transport are correct.

oRecall (43%) –43% of employees who will opt for private transport are correctly predicted.

•For predicting employees who opt for public transport

oPrecision (79%) –79% of prediction of employees who will opt for public transport are correct

oRecall (97%) –97% of employees who will opt for public transport are correctly predicted

Note: Overall accuracy of the model- 80% of the total prediction are correct.

Observation: If we see model the outputs we don't see much of differences in the scores. Hence, our Selection of this model as final model is the right decision to make.

If ABC company is looking for a model, which can help them to target potential employee, and thus helping them to reduce transport company acquisition cost than model with higher precision makes sense. As in this correctness of predicting employee, opting for public transport is more important. Lower the percentage means that the company might lose money on the respective lead.

If the company is looking to acquire higher number of employees irrespective of cost of acquisition than recall makes more sense. Here they want to maximise prediction of employee opting for public transport.

As ABC company wants to focus on providing maximum benefits to its employee base and thus the company would be interested in model which will more accurately predicted employees who will opt for public transport. Therefore, as compared to other performance measure recall is most critical.

From Logistic Regression Model (Feature Analysis)

- Age, Gender and MBA the continuous variable which have a positive influence on employees for opting public Transport
- Engineer, Work_Exp, Salary, Distance, License are not showing positive influence on employee for opting public Transport.
- If the Gender is Male the probability of employee taking public transport is more. In EDA also while doing Visualization analysis, we inferred the same.
- Surprisingly salary is not the criterion for individual to opt for Public or Private Transport.
- Long distance and higher salary, higher working experience employee are mostly opting for private transport.
- If we look at license feature, employee with no license are maximum employees opting for public transport.

Insight and Recommendation:

Our main Business objective is "To build a model to understand how do the employees of ABC Consulting prefer to commute presently (between home and office)."

Based on above various model building and analysis, we are recommending ABC company " The Logistic Regression Model", and below are the recommendation which will help them to understand why they should use this model to achieve their goal.

- Using Logistic Regression model for Training the model and using it for predictions is very simple, and it does not require a lot of engineering overhead for maintenance.
- The algorithm is extremely efficient. Fast training times combined with low computational requirements make logistic regression easy to scale, even when the data volume and speed increase.
- Logistic regression is selected because it demonstrated the best results in speed and accuracy.
- For logistic regression, it is easy to find out which variables affect the result of the predictions more and which ones less.
- Logistic regression can separate two classes of users easily. Based on the data, the model will help the company to decide the group of employees who will convert from public transport to transportation provided by the company.
- Logistic Regression will increase our overall accuracy of predicting employees who are more likely to use the transportation provided by the company.
- Company can focus on employees with no license and prefer commuting with public transport from home to office.
- Also, those employees who lives nearby office between 3km to 18km are mostly commuting with public transport can be the target employees for availing the benefit of company transportation.

Reference sites: https://machinelearningmastery.com/calculate-feature-importance-with-python/
https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization/?
https://www.scikit-yb.org/en/latest/api/model_selection/importances.html
https://www.scikit-yb.org/en/latest/api/model_selection/importances.html
https://machinelearningmastery.com/calculate-feature-importance-with-python/
https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization/

Thank you