
# MACHINE LEARNING PROJECT

SUJIT KUMAR

# Build a machine learning model, to predict the employees transport preference

## Table of contents

**Problem 1:**

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 finalized.

**Data Dictionary**

| | |
|---|---|
| Age | Age of the Employee in Years |
| Gender | Gender of the Employee |
| Engineer | For Engineer =1, Non-Engineer =0 |
| MBA | For MBA =1, Non-MBA =0 |
| Work Exp | Experience in years |
| Salary | Salary in Lakhs per Annum |
| Distance | Distance in Kms from Home to Office |
| license | If Employee has Driving Licence -1, If not, then 0 |
| Transport | Mode of Transport |

**Project Objective:**

The Objective of the report is to explore the dataset "Transport.csv" in Python (JUPYTER NOTEBOOK) and generate insights about the dataset. This exploration report will consist of the following:

- Importing the dataset in jupyter notebook.
- Understanding the structure of dataset.
- Exploratory Data analysis
- Graphical exploration
- Prediction using various machine learning models
- Insights from the dataset

**Assumptions:**

Predictive modelling is the general concept of building a model that can make predictions. Typically, such a model includes a machine learning algorithm that learns certain properties from a training dataset to make those predictions. Pattern classification is to assign discrete class labels to observations as outcomes of a prediction.

Machine learning model predictions allow businesses to make highly accurate guesses as to the likely outcomes of a question based on historical data, which can be about all kinds of things. These provide the business with insights that result in tangible business value.

**1.1 Read the dataset. Do the descriptive statistics and do the null value condition check. Write an inference on it.**

**Dataset:** Transport.csv

| | Age | Gender | Engineer | MBA | Work Exp | Salary | Distance | license | Transport |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 18 | Male | 1 | 0 | 0 | 6.8 | 12.2 | 0 | Public Transport |
| **1** | 18 | Male | 0 | 0 | 0 | 6.7 | 13.0 | 0 | Private Transport |
| **2** | 19 | Female | 1 | 0 | 1 | 7.5 | 8.1 | 0 | Public Transport |
| **3** | 20 | Male | 1 | 0 | 2 | 8.5 | 7.0 | 0 | Public Transport |
| **4** | 20 | Female | 0 | 1 | 1 | 8.5 | 7.9 | 0 | Public Transport |

## Information on dataset:

```
<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
```

### Inference

- There are 444 rows and 9 columns
- Numerical Columns: Age , Engineer , MBA , Work Exp , Salary , Distance and license
- Non-Numerical Columns: Transport and Gender.
- There are no null values

## Summary of the dataset:

| | count | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|
| **Age** | 444.0 | 27.747748 | 4.416710 | 18.0 | 25.0 | 27.0 | 30.000 | 43.0 |
| **Engineer** | 444.0 | 0.754505 | 0.430866 | 0.0 | 1.0 | 1.0 | 1.000 | 1.0 |
| **MBA** | 444.0 | 0.252252 | 0.434795 | 0.0 | 0.0 | 0.0 | 1.000 | 1.0 |
| **Work Exp** | 444.0 | 6.299550 | 5.112098 | 0.0 | 3.0 | 5.0 | 8.000 | 24.0 |
| **Salary** | 444.0 | 16.238739 | 10.453851 | 6.5 | 9.8 | 13.6 | 15.725 | 57.0 |
| **Distance** | 444.0 | 11.323198 | 3.606149 | 3.2 | 8.8 | 11.0 | 13.425 | 23.4 |
| **license** | 444.0 | 0.234234 | 0.423997 | 0.0 | 0.0 | 0.0 | 0.000 | 1.0 |

| | count | unique | top | freq |
|---|---|---|---|---|
| **Gender** | 444 | 2 | Male | 316 |
| **Transport** | 444 | 2 | Public Transport | 300 |

## Inference

- Gender and Transport variable have 2 unique values
- Transport: Public and Private
- Public Transport count is 300 which is more than Private Transport count
- gender: Male and Female
- Male employees are more than Female employees
- Age is continuous variable
- Minimum age of employee is 18
- Maximum age of employee is 43
- Average age of employee is 27
- Maximum distance between home and office is 23.4km
- Average distance between home and office is 11km
- Minimum distance between home and office is 3.2km

## Duplicates:

```
Number of duplicate rows = 0
```

There are no duplicates in the data frame.

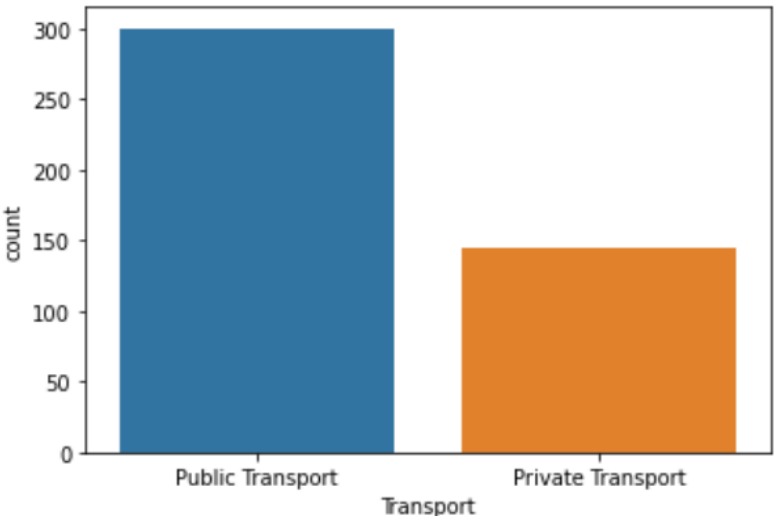
## Skewness:

```
Age          0.955276
Engineer    -1.186708
MBA          1.144763
Work Exp     1.352840
Salary       2.044533
Distance     0.539851
license      1.259293
dtype: float64
```

## Inference

- Skewness is a measure of the asymmetry of the probability distribution of a real-valued random variable about its mean.
- All variables have positive skewness except Engineer variable.
- Engineer has more skewness

## Unique values for categorical variables :-

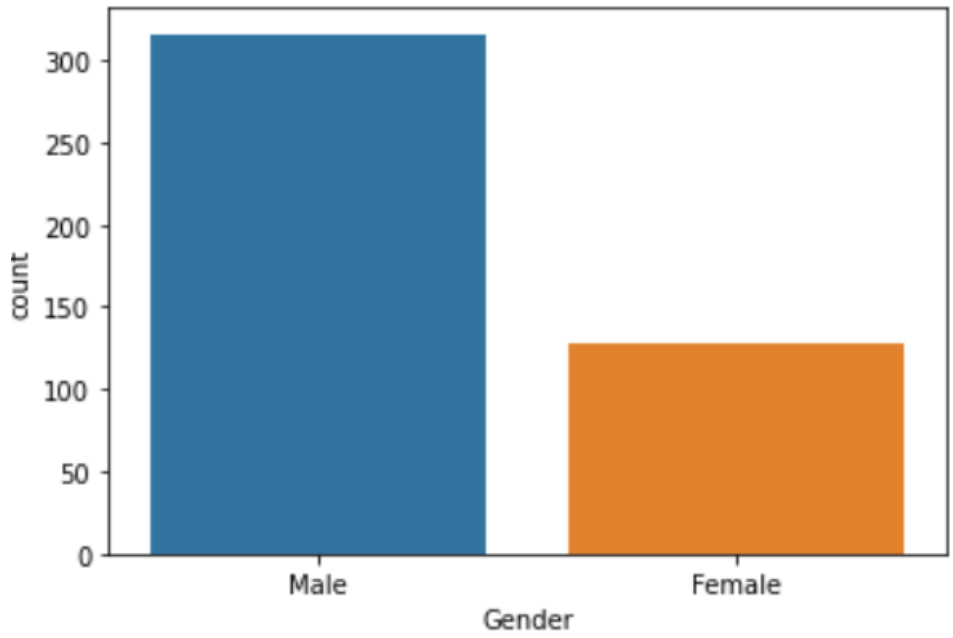



```
Public Transport     0.675676
Private Transport    0.324324
Name: Transport, dtype: float64
```

## Inference

- nearly 67% employees use public transport and only 32% employees use private transport.

## Gender

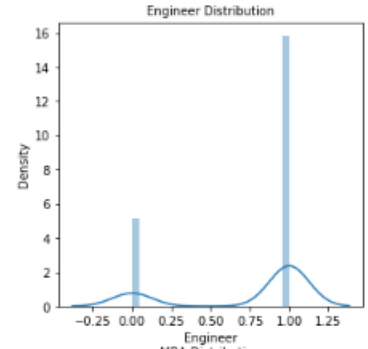


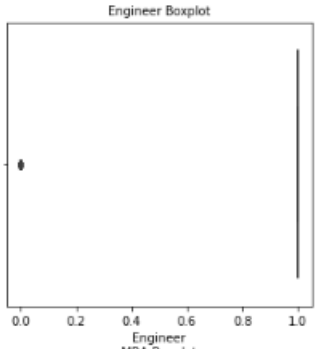

```
Male      0.711712
Female    0.288288
Name: Gender, dtype: float64
```

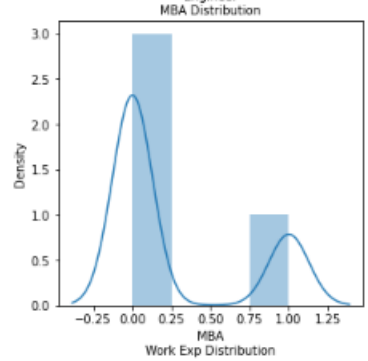
## Inference

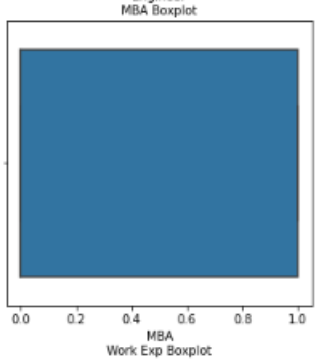
- 71% employees are male , and 28% employees are female.

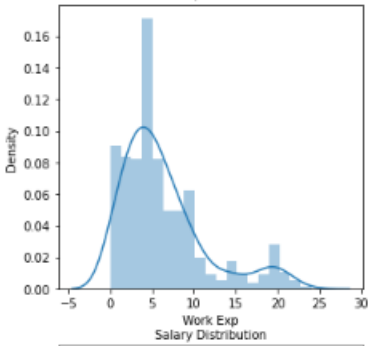
## 1.2 Perform Univariate and Bivariate Analysis. Do exploratory data analysis. Check for Outliers.

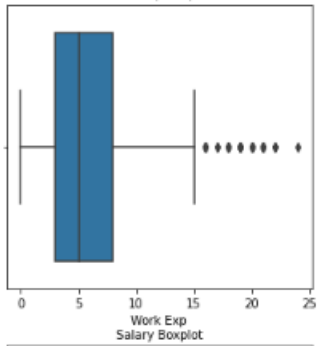


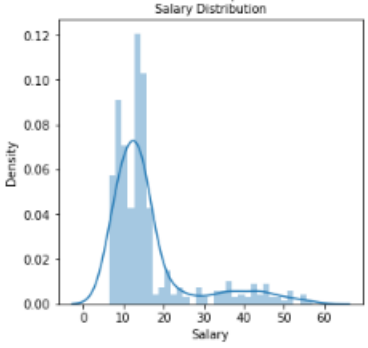



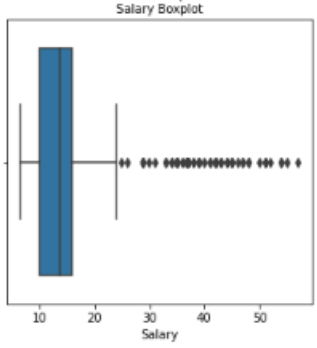



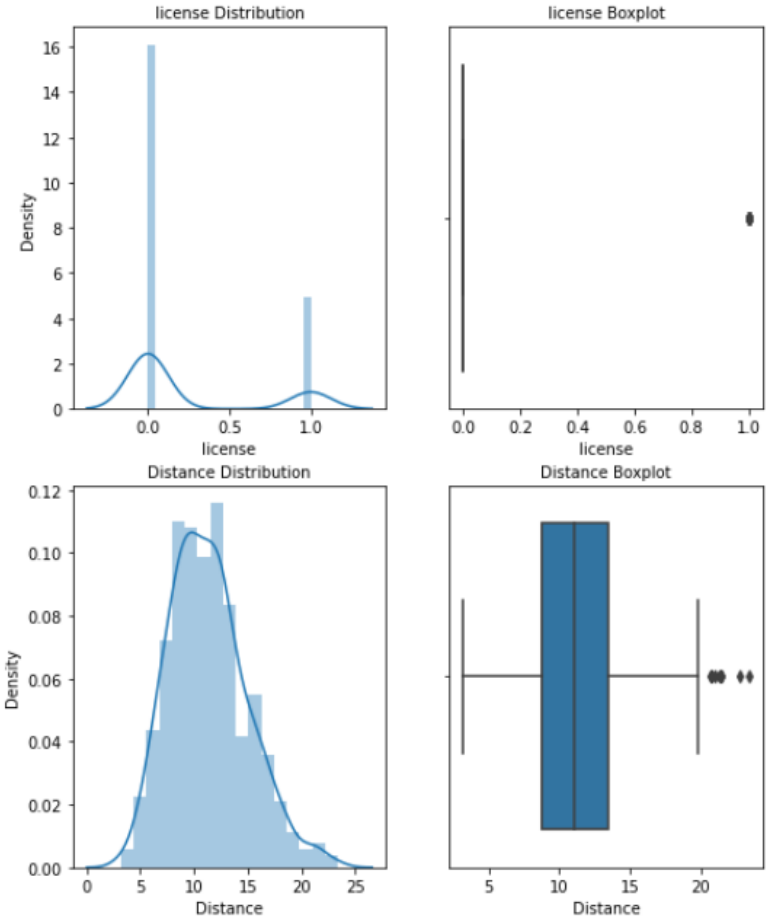



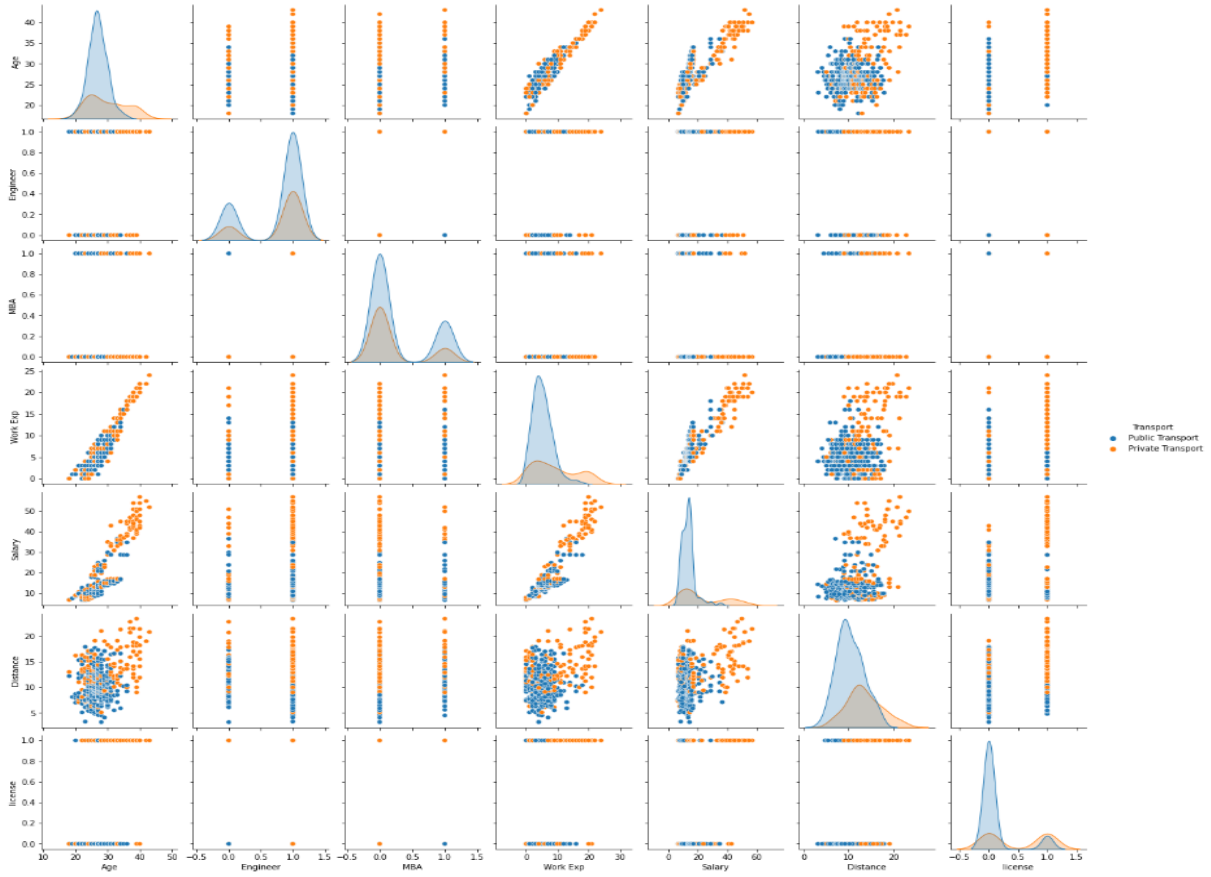



## Inference

- only age variable is normally distributed and other variables has multimodal skewness seen
- all variable have an outliers except MBA have outliers

## Bivariate Analysis:

## Pair plot between variables:




## Inference

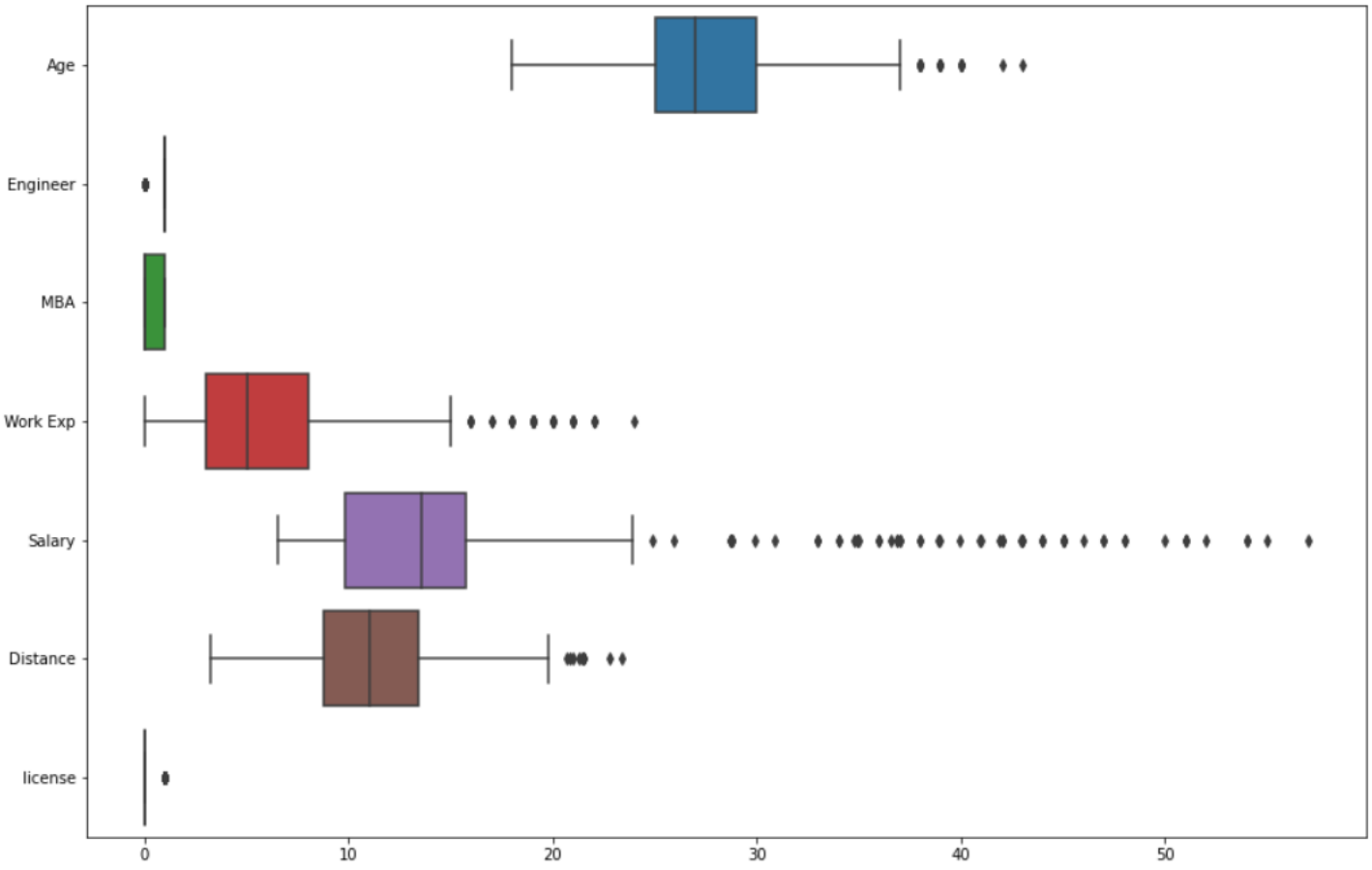
- There is a linear relationship between variables

## Correlation Heatmap Plot

**Inference**

- There is mostly positive correlation between the variables
- The highest positive correlation is seen between “Work Exp” and “Age” (93%) with nearly similar results seen from “Work Exp” and “Salary”(93%).
- The highest negative correlation is seen between “MBA” and “Age” (29%) with nearly similar results seen from “MBA” and “license” (27%).
- Overall the magnitude of correlations between the variables are very less.

**Boxplot :-**




**Inference :-**

1. There are outliers in all the variables except MBA. Ensemble techniques can handle the outliers.
2. Hence, Outliers are not treated for now, we will keep the data as it is.

## 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).

### Converting Object variables to categorical variables:

```
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]
```

| | Age | Gender | Engineer | MBA | Work Exp | Salary | Distance | license | Transport |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 18 | 1 | 1 | 0 | 0 | 6 | 12 | 0 | 1 |
| **1** | 18 | 1 | 0 | 0 | 0 | 6 | 13 | 0 | 0 |
| **2** | 19 | 0 | 1 | 0 | 1 | 7 | 8 | 0 | 1 |
| **3** | 20 | 1 | 1 | 0 | 2 | 8 | 7 | 0 | 1 |
| **4** | 20 | 0 | 0 | 1 | 1 | 8 | 7 | 0 | 1 |

### Inference

- Codes are an array of integers which are the positions of the actual values in the categories array.
- Here Transport and Gender are categorical variables are now converted into integers using codes
- All the variables in the data frame are integers

### Scaling of the data

- Most of the times, your dataset will contain features highly varying in magnitudes, units, and range. But since, most of the machine learning algorithms use Euclidean distance between two data points in their computations, this is a problem.
- Differences in the scales across input variables may increase the difficulty of the problem being modelled.
- This means that you are transforming your data so that it fits within a specific scale, like 0-100 or 0-1
- Usually, the distance-based methods (E.g.: KNN) would require scaling as it is sensitive to extreme difference and can cause a bias.
- tree-based method uses split method (E.g.: Decision Trees) would not require scaling in general as it's unnecessary.
- In this dataset, Age , Work Exp , Salary , Distance are only continuous variable and rest of the variables have 1 and 2. Age , Work Exp , Salary , Distance variable are only scaled because they are continuous variables
- The method of scaling performed only on the Age , Work Exp , Salary , Distance variable is the StandardScaler.

All the model prediction are done with scaled data

## Train-Test Split

Separating independent (train) and dependent (test)variables for the linear regression model

X = independent (train) variables

Y = dependent (test)variables

```
The training set for the independent variables: (310, 8)
The training set for the dependent variable: (310, )
The test set for the independent variables: (134, 8)
The test set for the dependent variable: (134, )
```

Inference

spilting the dataset into train and test set to build Logistic regression and LDA model (70:30)

X_train :70% of data randomly chosen from the 8 columns. These are training independent variables

X_test :30% of data randomly chosen from the 8 columns. These are test independent variables

y_train :70% of data randomly chosen from the "Transport" column. These are training independent variables

y_test :30% of data randomly chosen from the " Transport " columns. These are test dependent variable.

## 1.4 Apply Logistic Regression and LDA (linear discriminant analysis).

## Logistic Regression Model

Logistic regression is a fundamental classification technique.It belongs to the group of linear classifiers and is somewhat similar to polynomial and linear regression.It is the go-to method for binary classification problems (problems with two class values).

Two libraries of Logistic regression

1. sklearn
2. statsmodel

Here for the model sklearn library is used

## Applying GridSearchCV for Logistic Regression

### The probabilities on the training set

| | 0 | 1 |
|---|---|---|
| 0 | 0.064534 | 0.935466 |
| 1 | 0.083540 | 0.916460 |
| 2 | 0.191774 | 0.808226 |
| 3 | 0.617038 | 0.382962 |
| 4 | 0.241908 | 0.758092 |

### The probabilities on the test set

| | 0 | 1 |
|---|---|---|
| 0 | 0.314766 | 0.685234 |
| 1 | 0.325143 | 0.674857 |
| 2 | 0.451972 | 0.548028 |
| 3 | 0.117414 | 0.882586 |
| 4 | 0.116482 | 0.883518 |

## Fit the model to the training set

We now fit our model to the GridSearchCV for Logistic Regression model by training the model with our independent variable and dependent variables.

Inference

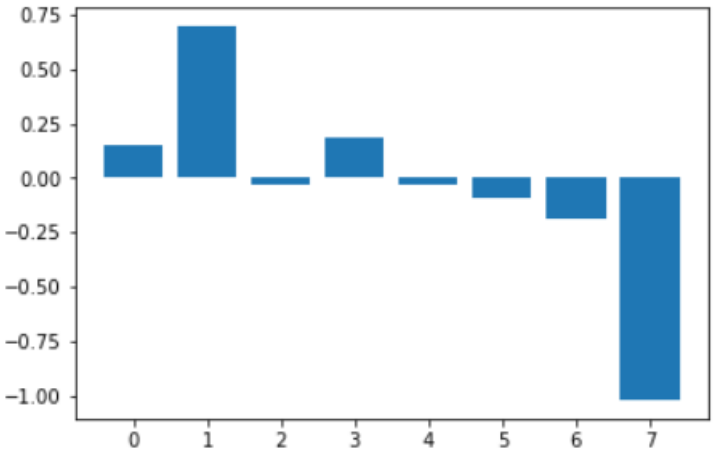
1. Using GridsearchCV, we input various parameters like 'max_iter', 'penalty',solver', 'tol' which will helps usto find best grid for prediction of the better model
2. max_iter is an integer (100 by default) that defines the maximum number of iterations by the solverduring model fitting.
3. solver is a string ('liblinear' by default) that decides what solver to use for fitting the model. Other optionsare 'newton-cg', 'lbfgs', 'sag', and 'saga'.
4. penalty is a string ('l2' by default) that decides whether there is regularization and which approach to use.Other options are 'l1', 'elasticnet', and 'none'.
5. bestgrid:{'max_iter': 1000, 'penalty': 'l2', 'solver': 'saga', 'tol': 1e-05}#Accuracy score of training data:78.3%
6. Accuracy score of test data:85.8%

## The coefficients for each of the independent attributes

```
The coefficient for Age is 0.1524314424700661
The coefficient for Gender is 0.6957119478314426
The coefficient for Engineer is -0.033138209061856584
The coefficient for MBA is 0.18799260055668557
The coefficient for Work Exp is -0.0349568856041856
The coefficient for Salary is -0.09146408470874967
The coefficient for Distance is -0.1914062006757103
The coefficient for license is -1.0204769664913549
```

Feature Importance Graphs

```
Feature: 0, Score: 0.15243
Feature: 1, Score: 0.69571
Feature: 2, Score: -0.03314
Feature: 3, Score: 0.18799
Feature: 4, Score: -0.03496
Feature: 5, Score: -0.09146
Feature: 6, Score: -0.19141
Feature: 7, Score: -1.02048
```




## Inference

- The coefficients for each of the independent attributes
- The sign of a regression coefficient tells you whether there is a positive or negative correlation between each independent variable the dependent variable. A positive coefficient indicates that as the value of the independent variable increases, the mean of the dependent variable also tends to increase. A negative coefficient suggests that as the independent variable increases, the dependent variable tends to decrease.
- Gender have more positive coefficient. A positive coefficient indicates that as the value of the independent variable increases, the mean of the dependent variable also tends to increase.

### Linear Discriminant Analysis

- Linear Discriminant Analysis (LDA) is a dimensionality reduction technique which is commonly used for the supervised classification problems.
- It is used for modeling differences in groups i.e., separating two or more classes. It is used to project the features in higher dimension space into a lower dimension space.
- library used in LDA is sklearn

**The probabilities on the training set**

| | 0 | 1 |
|---|---|---|
| 0 | 0.032931 | 0.967069 |
| 1 | 0.047642 | 0.952358 |
| 2 | 0.173447 | 0.826553 |
| 3 | 0.568737 | 0.431263 |
| 4 | 0.167712 | 0.832288 |

**The probabilities on the test set**

| | 0 | 1 |
|---|---|---|
| 0 | 0.233293 | 0.766707 |
| 1 | 0.282662 | 0.717338 |
| 2 | 0.447156 | 0.552844 |
| 3 | 0.142922 | 0.857078 |
| 4 | 0.061066 | 0.938934 |

Applying GridSearchCV for LDA

- Using GridsearchCV, we input various parameters like 'max_iter', 'penalty',solver', 'tol ' which will helps us to find best grid for prediction of the better model
- max_iter is an integer (100 by default) that defines the maximum number of iteration s by the solver during model fitting.
- solver is a string ('liblinear' by default) that decides what solver to use for fitting the model. Other options are 'newton-cg', 'lbfgs', 'sag', and 'saga'.
- here 'solver':['svd', 'lsqr', 'eigen'] are used with others parameters has default
- 'svd': Singular value decomposition (default). Does not compute the covariance matri x, therefore this solver is recommended for data with many features.
- 'lsqr': Least squares solution. Can be combined with shrinkage or custom covariance estimator.
- 'eigen': Eigenvalue decomposition. Can be combined with shrinkage or custom covari ance estimator.
- bestgrid:{'solver': 'svd'}
- Training Data Class Prediction with a cut-off value of 0.5
- Test Data Class Prediction with a cut-off value of 0.5
- Accuracy score of training data:78 %
- Accuracy score of test data:85%

## 1.5 Apply KNN Model. Interpret the results.

- KNN is a non-parametric and lazy learning algorithm. Non-parametric means there is no assumption for underlying data distribution. In KNN, K is the number of nearest neighbors. The number of neighbors is the core deciding factor

KNN has the following basic steps:

- Calculate distance
- Find closest neighbors
- Transport foremployees

we will be using popular scikit-learn package.

- The k-nearest neighbor algorithm is imported from the scikit-learn package.
- Create feature and target variables.
- Split data into training and test data.
- Generate a k-NN model using neighbors value.
- Train or fit the data into the model.
- Predict the future.

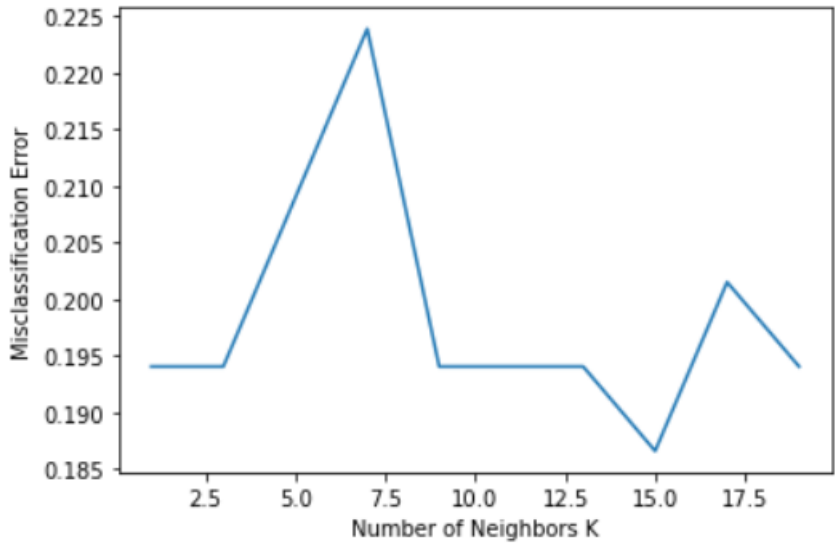
## KNN classifier model.

- First, import the K-Neighbors Classifier module and create KNN classifier object by passing argumentnumber of neighbors in K-Neighbors Classifier() function.
- Then, fit your model on the train set using fit() and perform prediction on the test set using predict().
- Let us build KNN classifier model for k=15.

misclassification error

```
[0.19402985074626866,
 0.19402985074626866,
 0.20895522388059706,
 0.22388059701492535,
 0.19402985074626866,
 0.19402985074626866,
 0.19402985074626866,
 0.18656716417910446,
 0.20149253731343286,
 0.19402985074626866]
```

## Plot misclassification error vs k (with k value on X-axis)




- The number of neighbors(K) in KNN is a hyperparameter that you need choose at the time of model building. You can think of K as a controlling variable for the prediction model.
- N_neighbors = 15

## The probabilities on the training set

| | 0 | 1 |
|---|---|---|
| 0 | 0.133333 | 0.866667 |
| 1 | 0.333333 | 0.666667 |
| 2 | 0.066667 | 0.933333 |
| 3 | 0.133333 | 0.866667 |
| 4 | 0.333333 | 0.666667 |

## The probabilities on the test set

| | 0 | 1 |
|---|---|---|
| 0 | 0.000000 | 1.000000 |
| 1 | 0.400000 | 0.600000 |
| 2 | 0.133333 | 0.866667 |
| 3 | 0.266667 | 0.733333 |
| 4 | 0.066667 | 0.933333 |

- Accuracy score of training data:80.6 %
- Accuracy score of test data:81.3%

## 1.6 Model Tuning, Bagging (Random Forest should be applied for Bagging) and Boosting.

### Model tuning

Tuning is the process of maximizing a model's performance without overfitting or creating too high of a variance. In machine learning, this is accomplished by selecting appropriate "hyperparameters."

Hyperparameters can be thought of as the "dials" or "knobs" of a machine learning model. Choosing an appropriate set of hyperparameters is crucial for model accuracy, but can be computationally challenging. Hyperparameters differ from other model parameters in that they are not learned by the model automatically through training methods. Instead, these parameters must be set manually. Many methods exist for selecting appropriate hyperparameters.

### Grid Search

Grid Search, also known as parameter sweeping, is one of the most basic and traditional methods of hyperparametric optimization. This method involves manually defining a subset of the hyperparametric space and exhausting all combinations of the specified hyperparameter subsets. Each combination's performance is then evaluated, typically using cross-validation, and the best performing hyperparametric combination is chosen.

### Bagging with randomforest

### 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 to reduce the variance of a black-box estimator (e.g., RandomForest), by introducing randomization into its construction procedure and then making an ensemble out of it.
Bagging and random forests are "bagging" algorithms that aim to reduce the complexity of models that overfit the training data.
Bagging (Random Forest should be applied for Bagging)
Inference
set the hyper parameters in randomforest classifier
N_estimators (only used in Random Forests) is the number of decision trees used in making the forest (default = 100).
Max_depth is an integer that sets the maximum depth of the tree. The default is None, which means the nodes are expanded until all the leaves are pure
Min_samples_split is the minimum number of samples required to split an internal node.

Min_samples_leaf defines the minimum number of samples needed at each leaf. The default input here is 1.

We now fit randomforest classifier model to the bagging model by training the model with our independent variable and dependent variables.

At this point, you have the classification model defined.

**The probabilities on the training set**

| Column1 | 0 | 1 |
|---|---|---|
| 0 | 0.115405 | 0.884595 |
| 1 | 0.311753 | 0.688247 |
| 2 | 0.173228 | 0.826772 |
| 3 | 0.295061 | 0.704939 |
| 4 | 0.331104 | 0.668896 |

**The probabilities on the test set**

| Column1 | 0 | 1 |
|---|---|---|
| 0 | 0.257569 | 0.742431 |
| 1 | 0.273913 | 0.726087 |
| 2 | 0.291294 | 0.708706 |
| 3 | 0.299722 | 0.700278 |
| 4 | 0.134457 | 0.865543 |

Accuracy score of training data:77%

Accuracy score of test data:83.5%

## Boosting

Boosting is an ensemble strategy that is consecutively builds on weak learners in order to generate one final strong learner. A weak learner is a model that may not be exactly accurate or may not take many predictors into account. By building a weak model, making conclusions about the various feature importance's and parameters, and then using those conclusions to build a new, stronger model, Boosting can effectively convert weak learners into a strong learner.

## AdaBoost (Adaptive Boosting):

AdaBoost uses decision stumps as weak learners. A Decision Stump is a Decision Tree model that only splits off at one level, ergo the final prediction is based off only one feature. When AdaBoost makes its first Decision Stump, all observations are weighted evenly.

To correct previous error, when moving to the second Decision Stump, the observations that were classified incorrectly now carry more weight than the observations that were correctly classified. AdaBoost continues this strategy until the best classification model is built.

- GridSearchCV ADA boosting
- Using GridsearchCV, we input various parameters like {'algorithm', 'learning_rate', 'n_estimators'} which will helps us to find best grid for prediction of the better model
- N_estimators is the maximum number of estimators at which boosting is terminated. If a perfect fit is reached, the algo is stopped. The default here is 50.
- Learning_rate is the rate at which we are adjusting the weights of our model with respect to the loss gradient.
- The SAMME.R algorithm typically converges faster than SAMME, achieving a lower test error with fewer boosting iterations.
- bestgrid: {'algorithm': 'SAMME.R', 'learning_rate': 0.3, 'n_estimators': 51}

**The probabilities on the training set**

| | 0 | 1 |
|---|---|---|
| 0 | 0.425810 | 0.574190 |
| 1 | 0.488283 | 0.511717 |
| 2 | 0.480466 | 0.519534 |
| 3 | 0.494553 | 0.505447 |
| 4 | 0.486134 | 0.513866 |

**The probabilities on the test set**

| | 0 | 1 |
|---|---|---|
| 0 | 0.485813 | 0.514187 |
| 1 | 0.489563 | 0.510437 |
| 2 | 0.493919 | 0.506081 |
| 3 | 0.443662 | 0.556338 |
| 4 | 0.482314 | 0.517686 |

- Accuracy score of training data:86.1%
- Accuracy score of test data:80.5%

## Gradient Boosting

## Gradient Boosting for classification.

GB builds an additive model in a forward stage-wise fashion; it allows for the optimization of arbitrary differentiable loss functions. In each stage n_classes_ regression trees are fit on the negative gradient of the binomial or multinomial deviance loss function. Binary classification is a special case where only a single regression tree is induced.

- Using GridsearchCV, we input various parameters like {'criterion', 'loss', 'max_features', 'min_samples_split', 'n_estimators} which will helps us to find best grid for prediction of the better model
- loss*{'deviance', 'exponential'}, default='deviance'*
  The loss function to be optimized. 'deviance' refers to deviance (= logistic regression) for classification with probabilistic outputs.
- criterion*{'friedman_mse', 'mse', 'mae'}, default='friedman_mse'*
- The function to measure the quality of a split. Supported criteria are 'friedman_mse' for the mean squared error with improvement score by Friedman, 'mse' for mean squared error, and 'mae' for the mean absolute error.
- n_estimators*int, default=100*
- The number of boosting stages to perform. Gradient boosting is fairly robust to over-fitting so a large number usually results in better performance.
- **min_samples_split*int or float, default=2***
- The minimum number of samples required to split an internal node:If int, then consider min_samples_split as the minimum number.
- **max_features*{'auto', 'sqrt', 'log2'}, int or float, default=None***
- The number of features to consider when looking for the best split:
- If int, then consider max_features features at each split.
- best_params: {'criterion': 'friedman_mse', 'loss': 'exponential', 'max_features': 6, 'min_samples_split': 30, 'n_estimators': 51}
- best_estimator:GradientBoostingClassifier(loss='exponential', max_features=6, min_samples_split=30, n_estimators=51)

### The probabilities on the training set

| | 0 | 1 |
|---|---|---|
| 0 | 0.052973 | 0.947027 |
| 1 | 0.048792 | 0.951208 |
| 2 | 0.022337 | 0.977663 |
| 3 | 0.116595 | 0.883405 |
| 4 | 0.021743 | 0.978257 |

### The probabilities on the test set

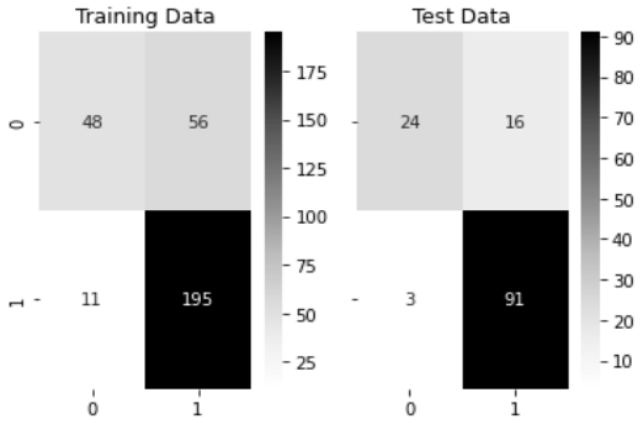
| | 0 | 1 |
|---|---|---|
| 0 | 0.012070 | 0.987930 |
| 1 | 0.494417 | 0.505583 |
| 2 | 0.308734 | 0.691266 |
| 3 | 0.489126 | 0.510874 |
| 4 | 0.014080 | 0.985920 |

- Accuracy score of training data:92.5%
- Accuracy score of test data:80.5%

## 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 the models and write inference which model is best/optimized.

## Logistic Regression Model

## Confusion matrix on the training and test data




## Inference

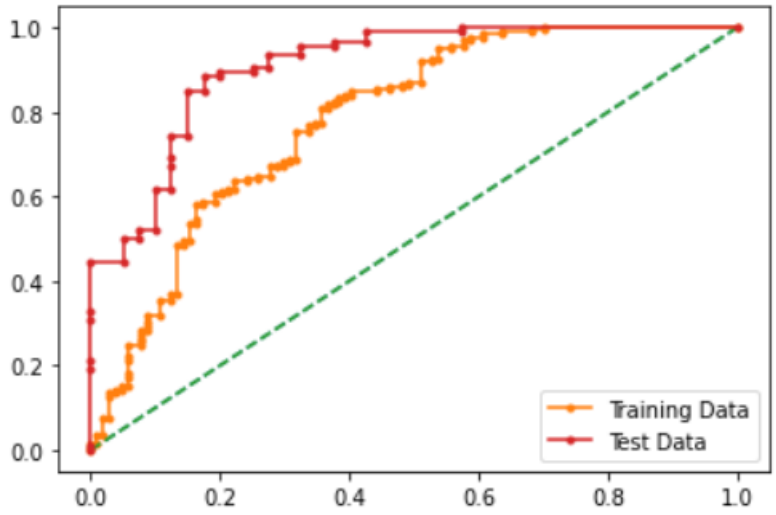
Training data:
True Negative: 48 False Positive: 56
False Negative: 11 True Positive :195
Test data:
True Negative: 24 False Positive: 16
False Negative: 3 True Positive: 91

### Classification Report of training and test data

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.81 | 0.46 | 0.59 | 104 |
| 1 | 0.78 | 0.95 | 0.85 | 206 |
| accuracy | | | 0.78 | 310 |
| macro avg | 0.80 | 0.70 | 0.72 | 310 |
| weighted avg | 0.79 | 0.78 | 0.76 | 310 |

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.89 | 0.60 | 0.72 | 40 |
| 1 | 0.85 | 0.97 | 0.91 | 94 |
| accuracy | | | 0.86 | 134 |
| macro avg | 0.87 | 0.78 | 0.81 | 134 |
| weighted avg | 0.86 | 0.86 | 0.85 | 134 |

## AUC and ROC for the training and test data




```
AUC for the Training Data: 0.786
AUC for the Test Data: 0.909
```

## Inference

Train Data:

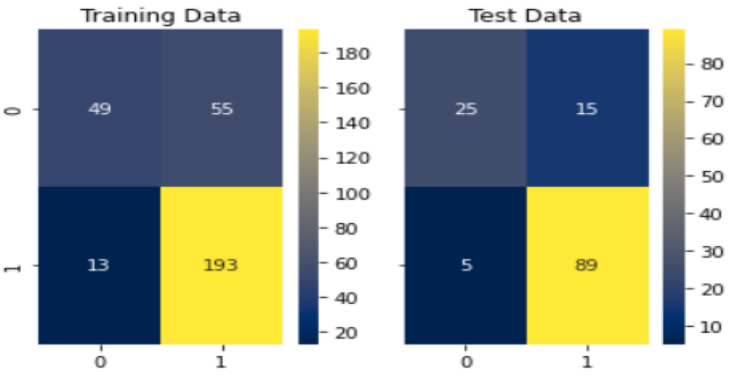
- AUC: 78.6%
- Accuracy: 78%
- precision : 85%
- recall : 95%
- f1 :85%

Test Data:

- AUC: 90.9%
- Accuracy: 86%
- precision: 85%
- recall : 97%
- f1 : 91%
- Training and Test set results are almost similar, this proves no overfitting or underfitting

## Linear Discriminant Analysis

## Confusion matrix on the training and test data




## Inference:

Training data:

- True Negative: 49 False Positive: 55
- False Negative: 13 True Positive: 193

Test data:

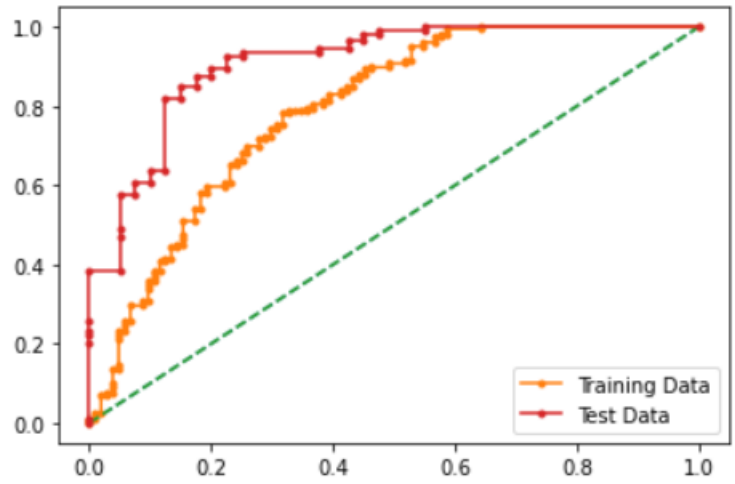
- True Negative: 25 False Positive: 15
- False Negative: 5 True Positive: 89

## Classification Report of training and test data

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.79 | 0.47 | 0.59 | 104 |
| 1 | 0.78 | 0.94 | 0.85 | 206 |
| accuracy | | | 0.78 | 310 |
| macro avg | 0.78 | 0.70 | 0.72 | 310 |
| weighted avg | 0.78 | 0.78 | 0.76 | 310 |

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.83 | 0.62 | 0.71 | 40 |
| 1 | 0.86 | 0.95 | 0.90 | 94 |
| accuracy | | | 0.85 | 134 |
| macro avg | 0.84 | 0.79 | 0.81 | 134 |
| weighted avg | 0.85 | 0.85 | 0.84 | 134 |

## AUC and ROC for the training and test data




```
AUC for the Training Data: 0.793
AUC for the Test Data: 0.911
```

## Inference

Train Data:

- AUC: 79.3%
- Accuracy: 83%
- precision : 78%
- recall : 94%
- f1 :85%

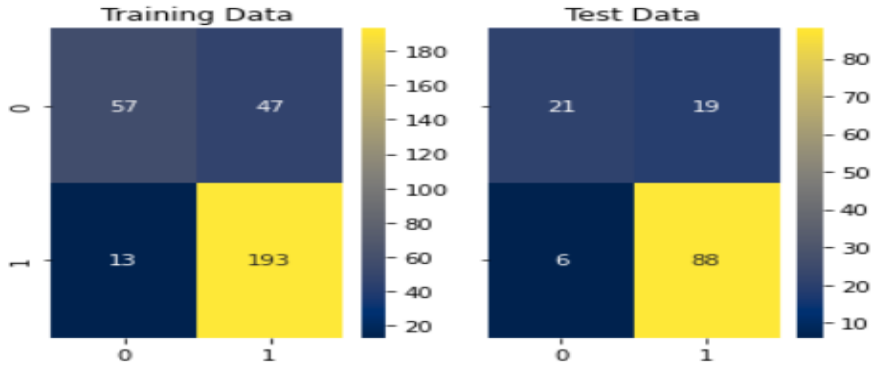
Test Data:

- AUC: 91.1%
- Acuracy: 83%
- precision :86%
- recall : 95%
- f1 : 90%

- Training and Test set results are almost similar, this proves no overfitting or underfitting

KNN Model

Confusion matrix on the training and test data




## Inference:

Training data:

- True Negative : 57 False Positive : 47
- False Negative : 13 True Positive : 193

Test data:

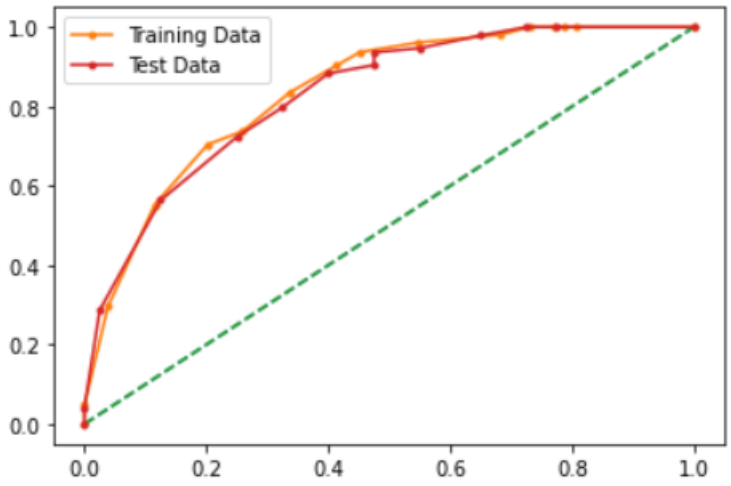
- True Negative : 21False Positive : 19
- False Negative : 6 True Positive : 88

## Classification Report of training and test data

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.81 | 0.55 | 0.66 | 104 |
| 1 | 0.80 | 0.94 | 0.87 | 206 |
| accuracy | | | 0.81 | 310 |
| macro avg | 0.81 | 0.74 | 0.76 | 310 |
| weighted avg | 0.81 | 0.81 | 0.79 | 310 |

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.78 | 0.53 | 0.63 | 40 |
| 1 | 0.82 | 0.94 | 0.88 | 94 |
| accuracy | | | 0.81 | 134 |
| macro avg | 0.80 | 0.73 | 0.75 | 134 |
| weighted avg | 0.81 | 0.81 | 0.80 | 134 |

## AUC and ROC for the training and test data




```
AUC for the Training Data: 0.837
AUC for the Test Data: 0.830
```

## Inference

Train Data:

- AUC: 91%
- Accuracy: 85%
- precision : 88%
- recall : 92%
- f1 :89%

Test Data:

- AUC: 89.3%
- Accuracy: 83%
- precision :85%
- recall : 90%
- f1 : 88%

- Training and Test set results are almost similar,This proves no overfitting or underfitting

## Bagging with random forest

### Confusion matrix on the training and test data


### Inference

Training data:

- True Negative : 39 False Positive : 65
- False Negative : 6 True Positive : 200

Test data:

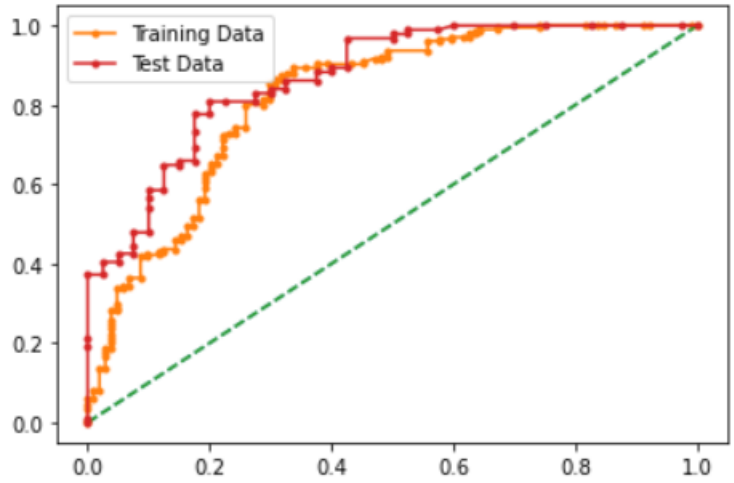
- True Negative : 20 False Positive : 20
- False Negative : 2 True Positive : 92

### Classification Report of training and test data

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.87 | 0.38 | 0.52 | 104 |
| 1 | 0.75 | 0.97 | 0.85 | 206 |
| accuracy | | | 0.77 | 310 |
| macro avg | 0.81 | 0.67 | 0.69 | 310 |
| weighted avg | 0.79 | 0.77 | 0.74 | 310 |

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.91 | 0.50 | 0.65 | 40 |
| 1 | 0.82 | 0.98 | 0.89 | 94 |
| accuracy | | | 0.84 | 134 |
| macro avg | 0.87 | 0.74 | 0.77 | 134 |
| weighted avg | 0.85 | 0.84 | 0.82 | 134 |

## AUC and ROC for the training and test data




```
AUC for the Training Data: 0.822
AUC for the Test Data: 0.870
```

## Inference

Train Data:

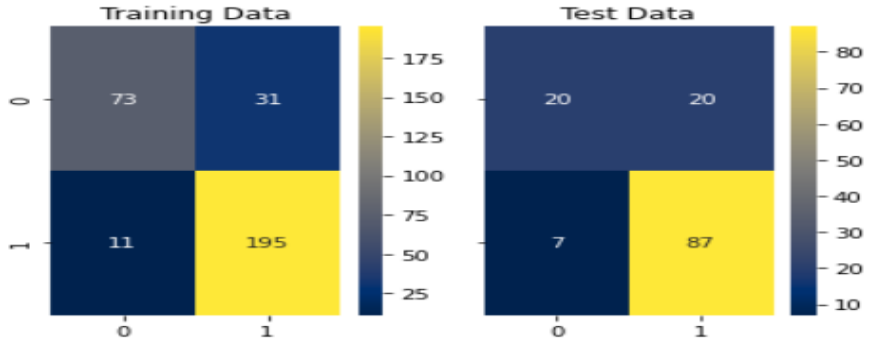
- AUC: 82.2%
- Accuracy: 77%
- precision : 75%
- recall : 97%
- f1 :85%

Test Data:

- AUC: 87.0%
- Accuracy: 84%
- precision :82%
- recall : 98%
- f1 : 89%
- Training and Test set results are almost similar, this proves no overfitting or underfitting.

## AdaBoostClassifier

Confusion matrix on the training and test data




## Inference

Training data:

- True Negative : 73 False Positive : 31
- False Negative : 11 True Positive : 195

Test data:

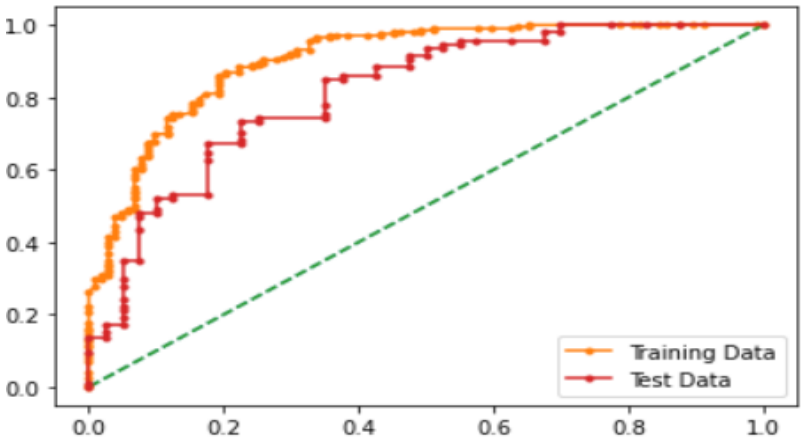
- True Negative : 20 False Positive : 20
- False Negative : 7 True Positive : 87

## Classification Report of training and test data

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.87 | 0.70 | 0.78 | 104 |
| 1 | 0.86 | 0.95 | 0.90 | 206 |
| accuracy | | | 0.86 | 310 |
| macro avg | 0.87 | 0.82 | 0.84 | 310 |
| weighted avg | 0.86 | 0.86 | 0.86 | 310 |

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.74 | 0.50 | 0.60 | 40 |
| 1 | 0.81 | 0.93 | 0.87 | 94 |
| accuracy | | | 0.80 | 134 |
| macro avg | 0.78 | 0.71 | 0.73 | 134 |
| weighted avg | 0.79 | 0.80 | 0.79 | 134 |

**AUC and ROC for the training and test data**




AUC for the Training Data: 0.906
AUC for the Test Data: 0.817

**Inference**

**Train Data:**

- AUC: 90.6%
- Accuracy: 86%
- precision : 86%
- recall : 95%
- f1 :90%

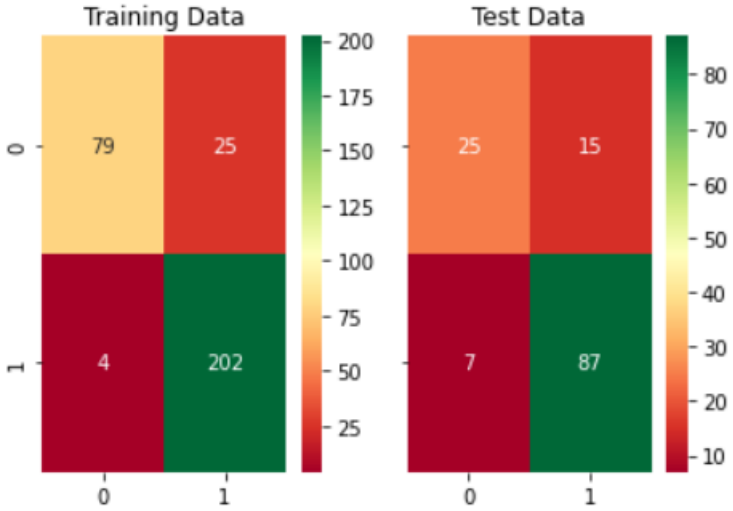
**Test Data:**

- AUC: 81.7%
- Accuracy: 80%
- precision :81%
- recall : 93%
- f1 : 87%

- Training and Test set results are almost similar, this proves no overfitting or underfitting

## Gradient Boosting

Confusion matrix on the training and test data




## Inference

Training data:

- True Negative : 79 False Positive :25
- False Negative : 4 True Positive : 202

Test data:

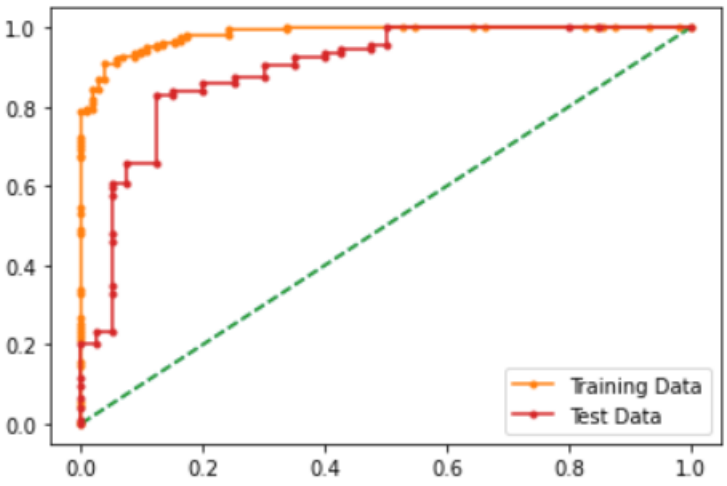
- True Negative : 25 False Positive : 15
- False Negative : 7 True Positive : 87

Classification Report of training and test data

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.95 | 0.76 | 0.84 | 104 |
| 1 | 0.89 | 0.98 | 0.93 | 206 |
| accuracy | | | 0.91 | 310 |
| macro avg | 0.92 | 0.87 | 0.89 | 310 |
| weighted avg | 0.91 | 0.91 | 0.90 | 310 |

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.78 | 0.62 | 0.69 | 40 |
| 1 | 0.85 | 0.93 | 0.89 | 94 |
| accuracy | | | 0.84 | 134 |
| macro avg | 0.82 | 0.78 | 0.79 | 134 |
| weighted avg | 0.83 | 0.84 | 0.83 | 134 |

## AUC and ROC for the training and test data




```
AUC for the Training Data: 0.984
AUC for the Test Data: 0.889
```

## Inference

### Train Data:

- AUC: 98.4%
- Accuracy: 93%
- precision : 89%
- recall : 98%
- f1 :93%

### Test Data:

- AUC: 88.9%
- Accuracy: 81%
- precision :85%
- recall : 93%
- f1 : 89%
- Training and Test set results are almost similar, this proves no overfitting orunderfitting
- This Gradient boosting shown better result for this dataset

## Final Model: Comparing all the models

| | LR Train | LR Test | LDA Train | LDA Test | KNN Train | KNN Test | BAGGING Train | BAGGING Test | ADA Train | ADA Test | Gradient Train | Gradient Test |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Accuracy | 0.78 | 0.86 | 0.78 | 0.85 | 0.81 | 0.81 | 0.77 | 0.84 | 0.86 | 0.81 | 0.91 | 0.91 |
| AUC | 0.79 | 0.91 | 0.79 | 0.91 | 0.84 | 0.83 | 0.82 | 0.87 | 0.91 | 0.82 | 0.98 | 0.89 |
| Recall | 0.95 | 0.97 | 0.94 | 0.95 | 0.94 | 0.94 | 0.97 | 0.98 | 0.95 | 0.93 | 0.98 | 0.93 |
| Precision | 0.78 | 0.85 | 0.78 | 0.86 | 0.80 | 0.82 | 0.75 | 0.82 | 0.86 | 0.81 | 0.89 | 0.85 |
| F1 Score | 0.85 | 0.91 | 0.85 | 0.90 | 0.87 | 0.88 | 0.85 | 0.89 | 0.90 | 0.87 | 0.93 | 0.89 |

## Inference

- Almost all the models performed well with accuracy between 78% to 86%. Gradient boosting improved the accuracy to 91% so it is better model for to predict that the employees preferred mode of transport
- Comparing all the model, Gradient boosting model is best model for this dataset with accuracy of 91% in both training and test set
- AUC of Train and test in Gradient boosting model is 98% and 89% respectively
- f1 score of Train and test in Gradient boosting model is 93% and 89% respectively
- Precision of Train and test in Gradient boosting model is 89% and 85% respectively
- Recall of Train and test in Gradient boosting model is 98% and 93% respectively
- Accuracy, AUC, Precision, Recall for test data are almost in line with training data in Gradient boosting model. This indicates no overfitting or underfitting inthe model
- Gradient boosting improved the accuracy to 91% so it is better model for to predict preferred mode of transport for the employees.

## Overall Best/Optimized model:

- Almost all the models performed well with accuracy between 78% to 86% with scaleddata. But Gradient boosting is best and optimized model with accuracy of 91% and also best AUC, Precision, F1 score, Recall .

## 1.8 Based on these predictions, what are the insights?

The main business objective of this project is to build a model to predict whether an employee will use public or private mode of transport.

Various model was built on scaled dataset in that it is found that Gradient Boosting model gave best/optimized accuracy with 91% to predict the employees preferred mode of transport.

Sample data test

With sample data we can test against each optimized model.

Information of the sample data:

- 'Age' :4
- 'Gender': 6
- 'Engineer': 10
- 'MBA': 8
- 'Work Exp':20
- 'Salary': 11
- 'Distance': 7
- 'License': 21

**From the sample employees, final model conclusion**

| Model | Prediction |
|---|---|
| **Logistic Regression** | Private Transport |
| **Linear Discriminant Analaysis** | Private Transport |
| **K-Nearest Neighbour** | Public Transport |
| **Bagging(with Random Forest)** | Public Transport |
| **Adaptive Boosting** | Public Transport |
| **Gradient Boosting** | Public Transport |

## Conclusion

- Based on this machine learning models, we can predict the preferred mode of transport with more sample data. Preferred mode of transport can be created with this model to predict which employee will choose which mode of transport.
- Employees who have more experience tends to have more age and earn more salary. Private mode of transport are affordable and used as mode of transport by employee who are earning more salary.
- Employees whose salary is more than 30L approx. are opting for private mode of transport. This could be since they are the group of audience who can afford a Private mode of transport.
- While distance plays a little role, people travelling more than 20 KMs of distance are choosing private mode of transport as it provides comfort.

THE END