Project Report

on

Bank Loan Default Case

Loan Default Analysis

Problem Statement:

The loan default dataset has 8 variables and 850 records, each record being loan default status for each customer. Each Applicant was rated as "Defaulted" or "Not-Defaulted". New applicants for loan application can also be evaluated on these 8 predictor variables and classified as a default or non-default based on predictor variables.

Tools: Python Jupyter Notebook, R Studio.

Techniques: Logistic Regression, Decision Tree, Random Forest, KNN Imputation, Naïve Bayes.

Data Analysis

For this project, the dataset we have consists of 850 observations and 8 features. 9th column is the dependent variable containing 150 empty cells which will be predicted in the project based on best suitable model. We started from Exploratory Data Analysis on given data.

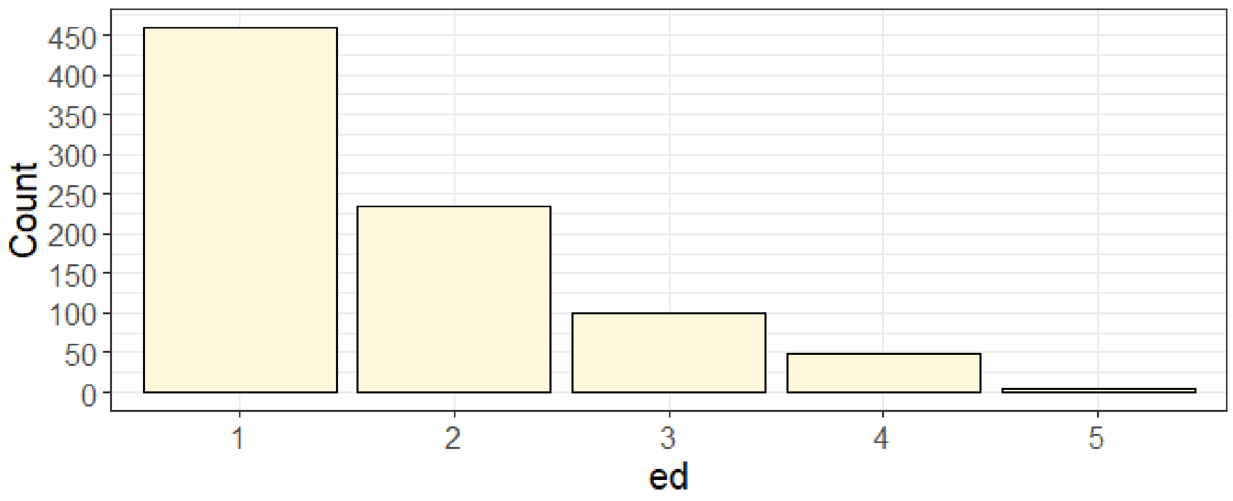
Missing Value Analysis:

First, we check if there is any empty cell in any column or not. In the Bank-loan data, there is no empty cell in any independent variable, therefore, there is no need to impute any value or remove any row.

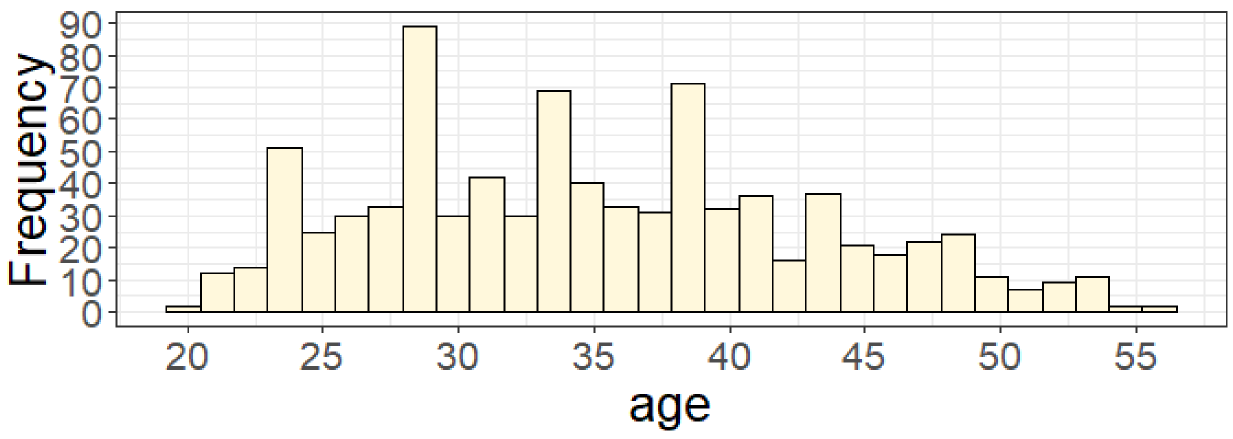
Graphs:

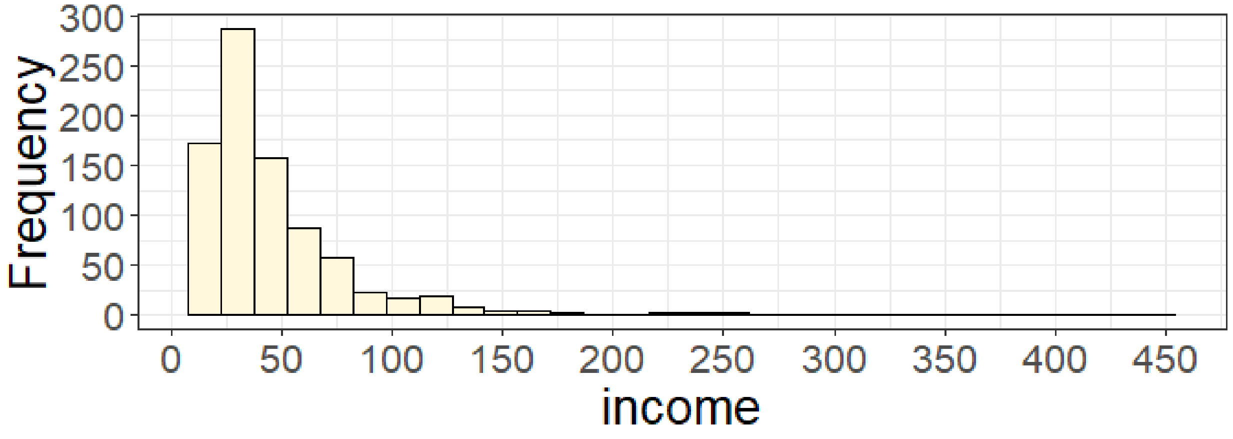
Plotting Graphs to know the nature of data

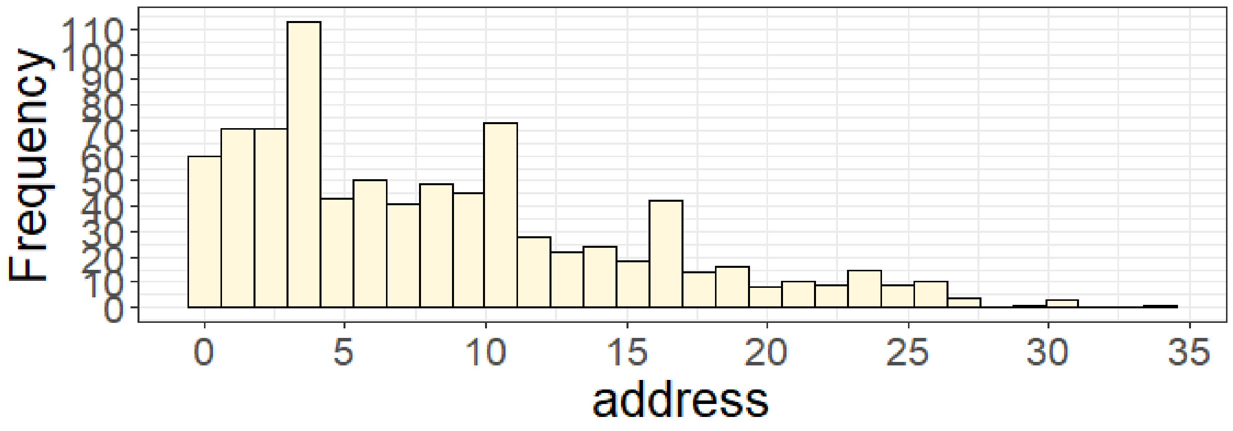
<u>Categorical Variable:</u>

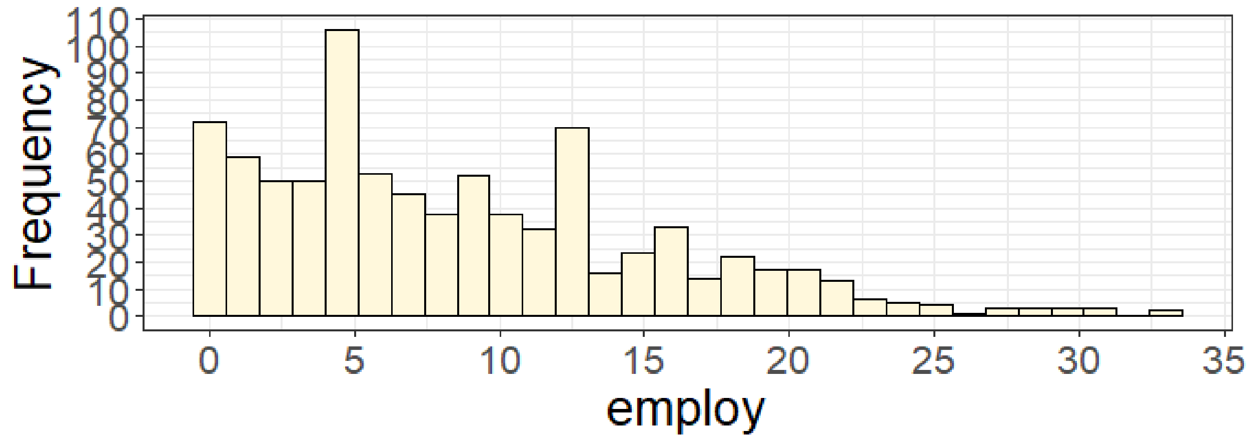


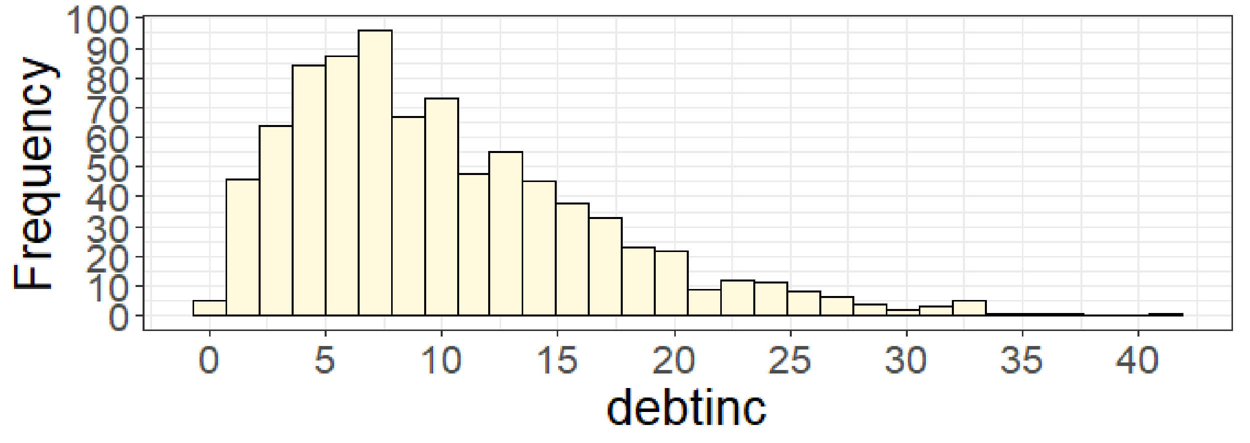
Continuous Variables:

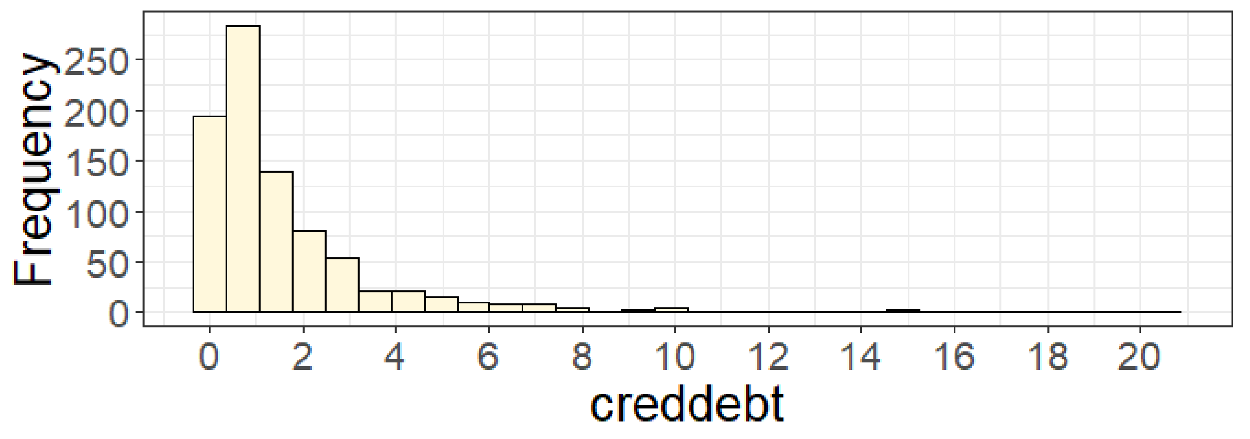


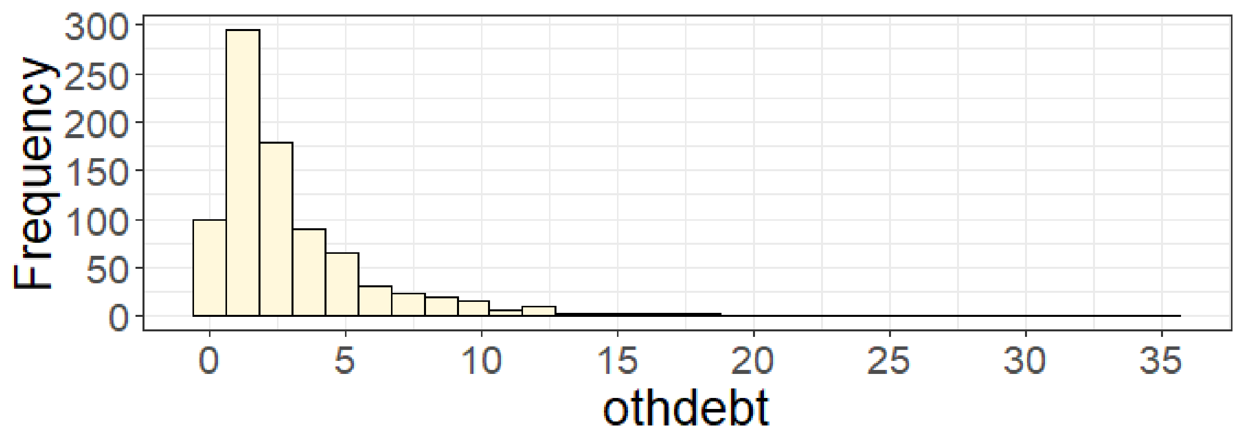





Frequency
0
10
20
30
40
50
60
70
80
90
100
110
0
5
10
15
20
25
30
35
employ


Frequency
0
10
20
30
40
50
60
70
80
90
100
0
5
10
15
20
25
30
35
40
debtinc


Frequency
0
50
100
150
200
250
0
2
4
6
8
10
12
14
16
18
20
creddebt

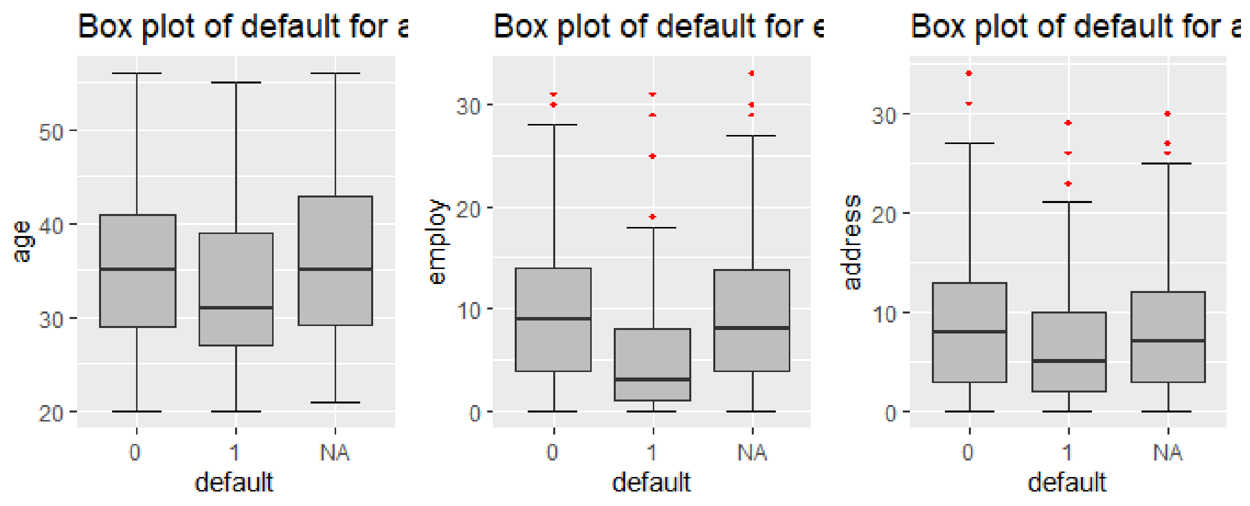


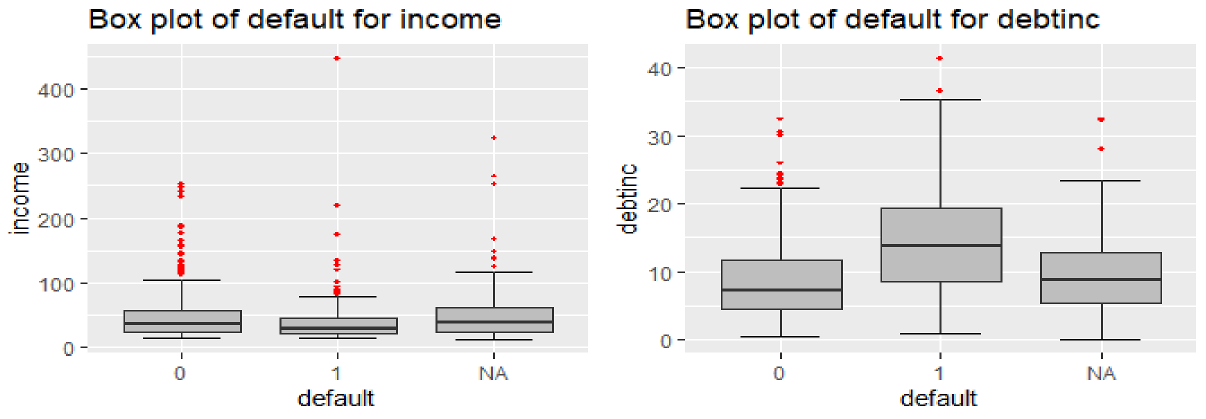
By Graphs, we can see that the data is Right skewed which means there is high possibility of outliers.

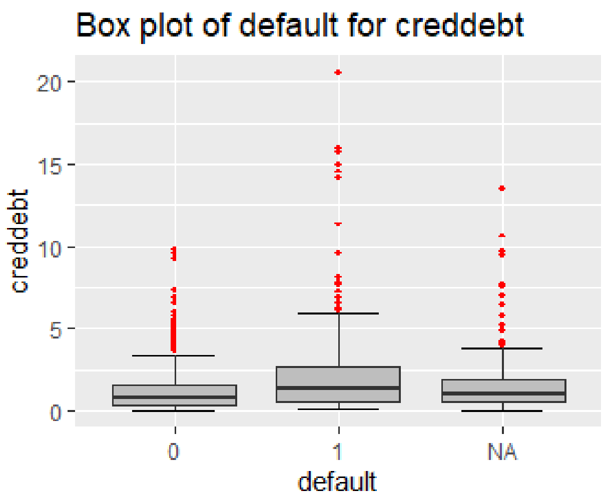
Outlier Analysis:

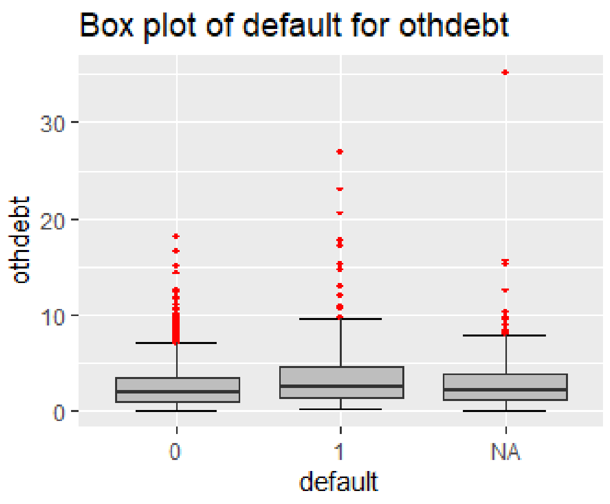
Using Box Plot method, we tested continuous numerical variables for outliers. After detecting outliers, there are two ways to remove outliers:

- Removing rows with outliers.
- Imputing values in place of outliers.







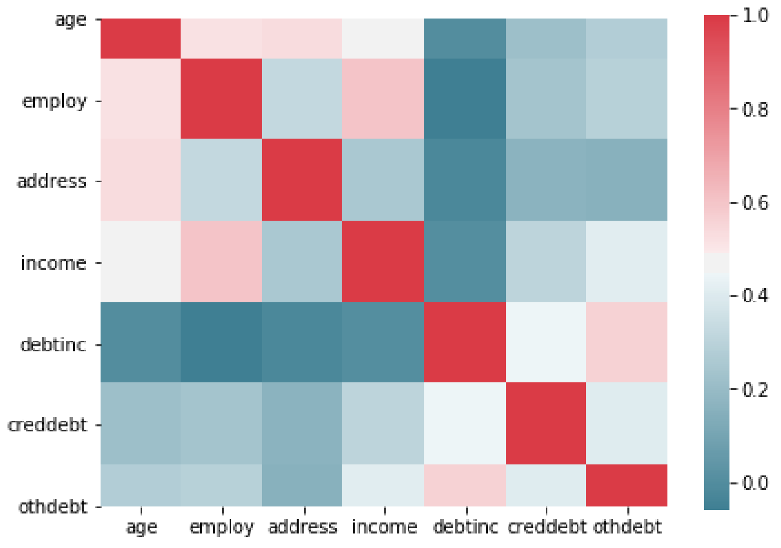


The data set is already very small. If we remove the rows which have outliers, it becomes very difficult to build model with good accuracy on such small data set. Therefore, we replaced outliers with Mean value of that column.

Feature Selection:

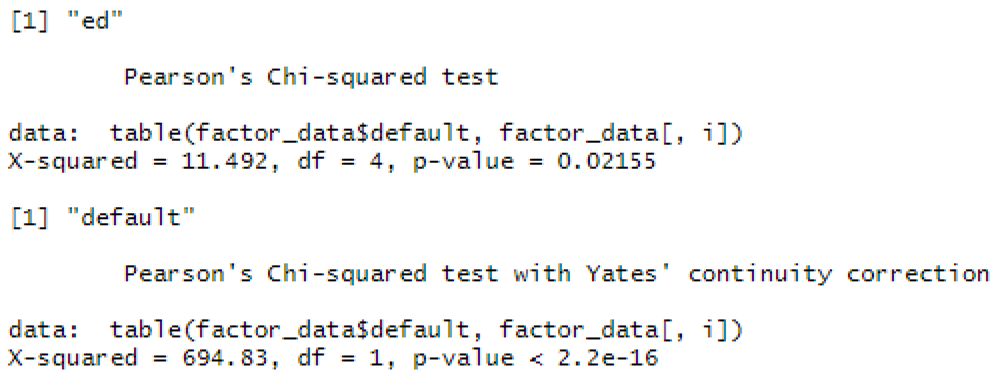
Some variables contains same information, so it's better to remove such variables. For continuous variable, correlation is used and for categorical variable, Chi – Squared test is used.

Heat map showing Correlation between continuous variables:



From Correlation Graph, we see that variables are not correlated with each other. Therefore, no variable is removed here.

Chi – Squared test:



```
[1] "ed"

        Pearson's Chi-squared test

data:  table(factor_data$default, factor_data[, i])
X-squared = 11.492, df = 4, p-value = 0.02155

[1] "default"

        Pearson's Chi-squared test with Yates' continuity correction

data:  table(factor_data$default, factor_data[, i])
X-squared = 694.83, df = 1, p-value < 2.2e-16
```

There is only one categorical variable and its p-value is less than 0.05, so it will not be removed.

Feature Scalling:

We have scaled the data using normalisation so that difference between means of variable should not be very large as it effects the model. We can do scalling using standarization also. Both methods gives same result for this problem statement.

Sampling:

The data set consists of 850 observations out of which 150 observations has target variable as null. Those observations which have 0 or 1 as target variable are taken as Historical data and 150 observations which have null target variable are taken as future observations.
Historical values are divided in two different sets: Train and Test to build model and check stability of model.
We have used Stratified sampling method to divide data in Test and Train data sets.
Train consists of 70% of Historical data while Test have remaining 30% data.

Models:

We are looking for people who will default.
For our Loan default prediction project, False Negatives Rate is the best metric to evaluate the model. Lower the number of false negatives, better the model is. So, we evaluated our models using the number of False negatives, area under ROC curve and accuracies.

Logistic Regression:

Code in R:

Observations:
Confusion Matrix and Statistics
Logistic Predictions

	0	1
0	137	18
1	23	31

Most basic terms, which are whole numbers:

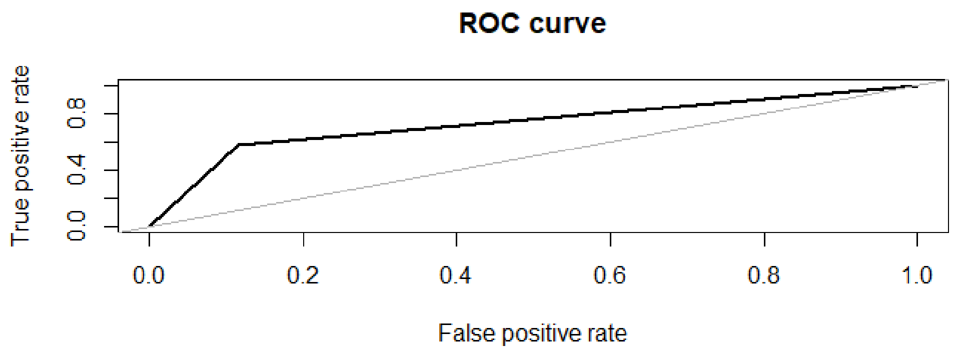
- **True positives (TP):** These are cases in which we predicted Yes, and they do default = 31

- **True negatives (TN):** We predicted No, and they do not default = 137
- **False positives (FP):** We predicted yes, but they do not actually do default. (Also known as a "Type I error.") = 18
- **False negatives (FN):** We predicted no, but they do default. (Also known as a "Type II error.") = 23

This is a list of rates that are often computed from a confusion matrix for a binary classifier:

- **Accuracy:** Overall, how often is the classifier correct?
  - (TP+TN)/total = (31 + 137)/209 = .803
- **Misclassification Rate:** Overall, how often is it wrong?
  - (FP+FN)/total = (21+21)/209 = 0.196
  - equivalent to 1 minus Accuracy
  - also known as "Error Rate"
- **True Positive Rate:** When it is yes, how often does it predict yes?
  - TP/actual yes = 31/54 = 0.574
  - also known as "Sensitivity" or "Recall"
- **False Positive Rate:** When it is no, how often does it predict yes?
  - FP/actual no = 18/155= 0.116
- **True Negative Rate:** When it is no, how often does it predict no?
  - TN/actual no = 137/155 = 0.883
  - equivalent to 1 minus False Positive Rate
  - also known as "Specificity"
- **False Negative Rate:** When it is Yes, how often does it predict no?
  - FN/actual yes = 23/54 = 0.425
- **Precision:** When it predicts yes, how often is it correct?
  - TP/predicted yes = 31/49 = 0.632
- **Prevalence:** How often does the yes condition occur in our sample?
  - actual yes/total = 54/209 = 0.258
- **Cohen's Kappa:** This is essentially a measure of how well the classifier performed as compared to how well it would have performed simply by chance. = 0.4722

ROC Curve: 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.



Area under the curve (AUC): 0.729

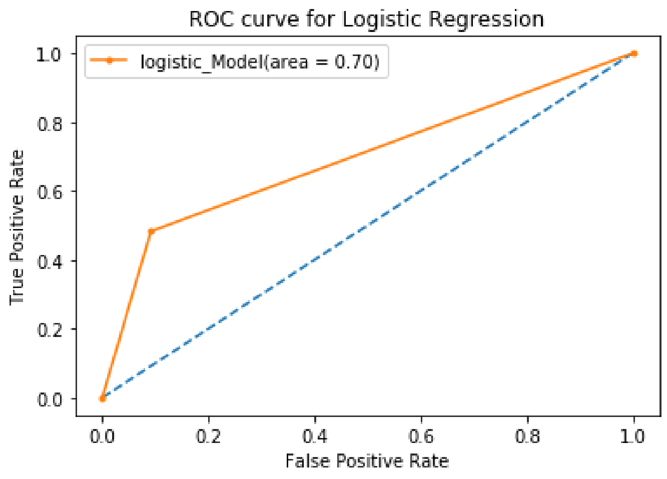
Code in Python:
Observations:
Confusion Matrix and Statistics

Logistic Predictions

```
      0   1
 0 148  15
 1   31  29
```

- **True positives (TP):** 29
- **True negatives (TN):** 148
- **False positives (FP):** 15
- **False negatives (FN):** 31
- **Accuracy:** (29 + 148)/223 = .7937
- **Misclassification Rate:** (FP+FN)/total = (31+15)/223 = 0.2063
- **True Positive Rate:** TP/actual yes = 29/60 = 0.4833
- **False Positive Rate:** FP/actual no = 15/163= 0.0920
- **True Negative Rate:** TN/actual no = 148/163 = 0.9079
- **False Negative Rate:** FN/actual yes = 31/60 = 0.5166
- **Precision:** TP/predicted yes = 29/44 = 0.6590

ROC Curve:



Result:

R	**Python**
Accuracy: 80.38%	Accuracy: 79.38%
FNR: 42.59%	FNR: 51.66%
ROC: 72.90%	ROC: 70.00%

Model have higher False Negative Rate and low accuracy, therefore, not selecting this Model for Loan Default prediction as we can achieve much better in other models.

<u>Decision Tree:</u>

Code in R

Observations:

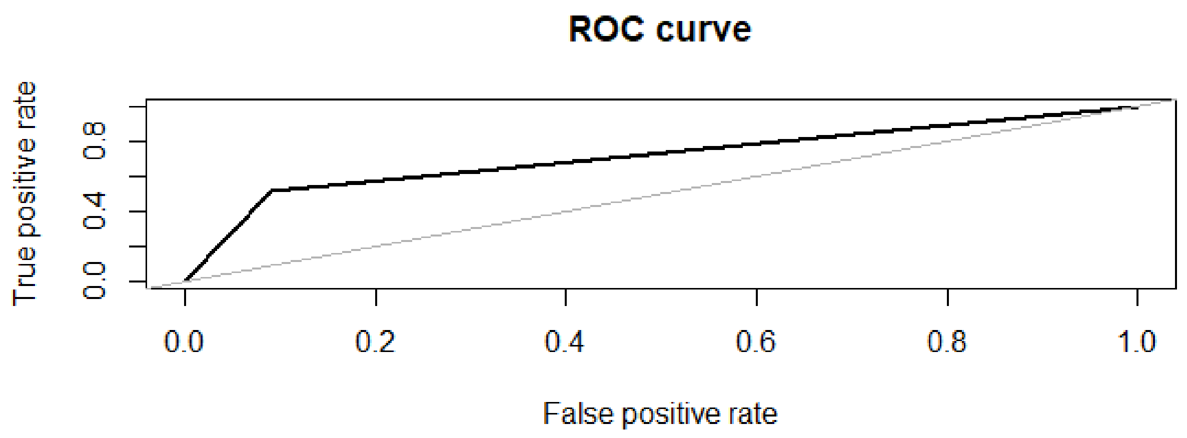
Confusion Matrix and Statistics

Decision tree Predictions

```
       0   1
 0 141  14
 1   26  28
```

- **True positives (TP):** 28
- **True negatives (TN):** 141
- **False positives (FP):** 14
- **False negatives (FN):** 26
- **Accuracy:** (28 + 141)/209 = .8086
- **Misclassification Rate:** (FP+FN)/total = (26+14)/209 = 0.1913
- **True Positive Rate:** TP/actual yes = 28/54 = 0.518
- **False Positive Rate:** FP/actual no = 14/155= 0.0903
- **True Negative Rate:** TN/actual no = 141/155 = 0.9096
- **False Negative Rate:** FN/actual yes = 26/54 = 0.4814
- **Precision:** TP/predicted yes = 28/42 = 0.6666
- **Cohen's Kappa:** 0.4616

ROC Curve:



Area under the curve (AUC): 0.714

Code in Python

Observations:

Confusion Matrix and Statistics

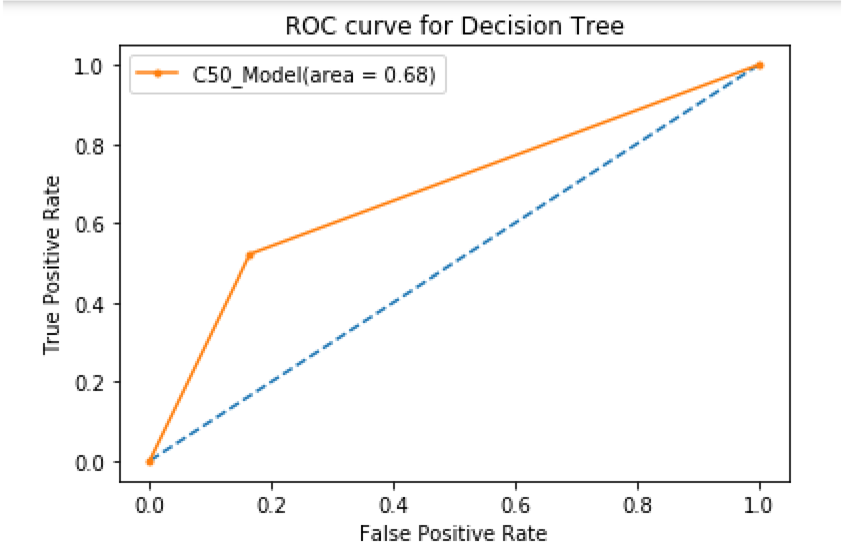
Decision tree Predictions

```
       0   1
 0 137  27
 1   22  24
```

- **True positives (TP):** 24
- **True negatives (TN):** 137

- **False positives (FP):** 27
- **False negatives (FN):** 22
- **Accuracy:** (24 + 137)/210 = .7666
- **Misclassification Rate:** (FP+FN)/total = (22+27)/210 = 0.2333
- **True Positive Rate:** TP/actual yes = 24/46 = 0.5217
- **False Positive Rate:** FP/actual no = 27/164= 0.1646
- **True Negative Rate:** TN/actual no = 137/164 = 0.8353
- **False Negative Rate:** FN/actual yes = 22/46 = 0.4782
- **Precision:** TP/predicted yes = 24/51 = 0.4705

ROC Curve:



Result:

R	**Python**
Accuracy: 80.86%	Accuracy: 76.66%
FNR: 48.14%	FNR: 47.82
ROC: 71.40%	ROC: 68.00%

False Negative Rate is High and less accuracy, therefore, not choosing this model.

Random Forest:

After Testing on so many numbers of trees, we have selected ntree = 625 as we get best result on this number.

Code in R

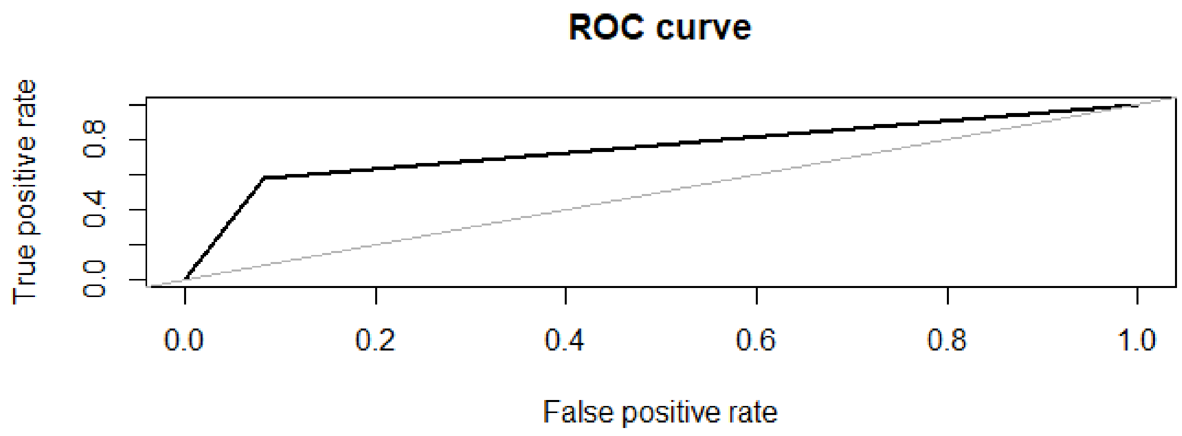
Observations:

Confusion Matrix and Statistics

```
RF Predictions
       0   1
 0 142  13
 1   23  31
```

- **True positives (TP):** 31
- **True negatives (TN):** 142
- **False positives (FP):** 13
- **False negatives (FN):** 23
- **Accuracy:** (28 + 141)/209 = .8278
- **Misclassification Rate:** (FP+FN)/total = (13+23)/209 = 0.1722
- **True Positive Rate:** TP/actual yes = 31/54 = 0.5740
- **False Positive Rate:** FP/actual no = 13/155= 0.0838
- **True Negative Rate:** TN/actual no = 142/155 = 0.9161
- **False Negative Rate:** FN/actual yes = 23/54 = 0.4259
- **Precision:** TP/predicted yes = 31/44 = 0.7045
- **Cohen's Kappa:** 0.5217

ROC Curve:



Area under the curve (AUC): 0.745

Code in Python

Observations:

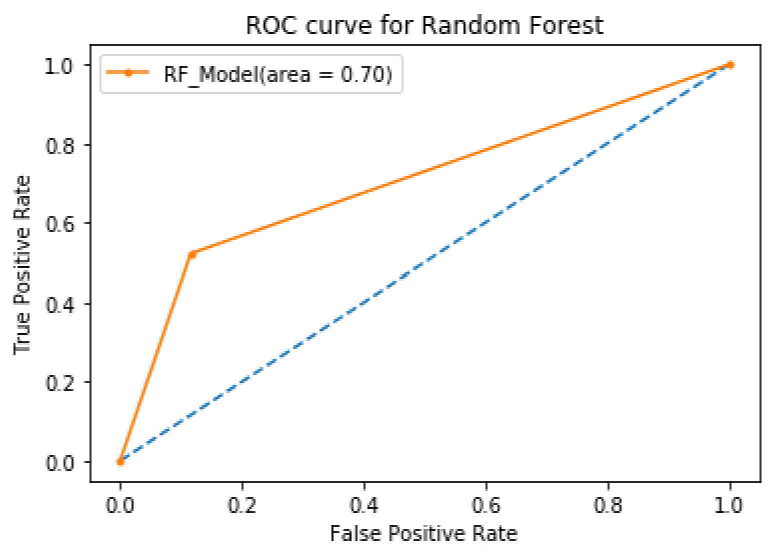
Confusion Matrix and Statistics

Random Forest Predictions

```
      0   1
 0 145  19
 1   22  24
```

- **True positives (TP):** 24
- **True negatives (TN):** 145
- **False positives (FP):** 19
- **False negatives (FN):** 22
- **Accuracy:** (24 + 145)/210 = .8047
- **Misclassification Rate:** (FP+FN)/total = (22+19)/210 = 0.195
- **True Positive Rate:** TP/actual yes = 24/46 = 0.5217
- **False Positive Rate:** FP/actual no = 19/164= 0.1158
- **True Negative Rate:** TN/actual no = 145/164 = 0.8841
- **False Negative Rate:** FN/actual yes = 22/46 = 0.4782
- **Precision:** TP/predicted yes = 24/51 = 0.5581

ROC Curve:



Result:

R
Accuracy: 82.78%
FNR: 42.59%
ROC: 74.50%

Python
Accuracy: 80.47%
FNR: 47.82
ROC: 70.29%

Model have Higher accuracy than other models in both programming languages and False negative Rate is also low comparing to other models. Area under ROC curve is also High, so selecting this model to predict future data.

<u>KNN Implementation:</u>

Code in R:

Observations:

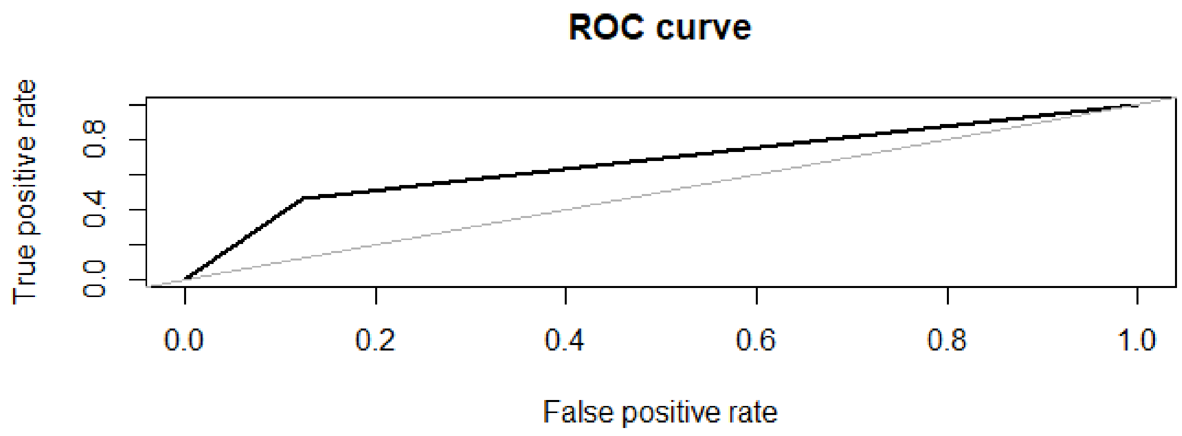
Confusion Matrix and Statistics

```
KNN Predictions
       0   1
0  140  29
1    15  25
```

- **True positives (TP):** 25
- **True negatives (TN):** 140
- **False positives (FP):** 15
- **False negatives (FN):** 29
- **Accuracy:** (25 + 140)/209 = .7895
- **Misclassification Rate:** (FP+FN)/total = (15+29)/209 = 0.2105
- **True Positive Rate:** TP/actual yes = 25/54 = 0.4629
- **False Positive Rate:** FP/actual no = 15/155= 0.096
- **True Negative Rate:** TN/actual no = 140/155 = 0.9032

- **False Negative Rate:** FN/actual yes = 29/54 = 0.5370
- **Precision:** TP/predicted yes = 25/40 = 0.625
- **Cohen's Kappa:** 0.4

ROC Curve:



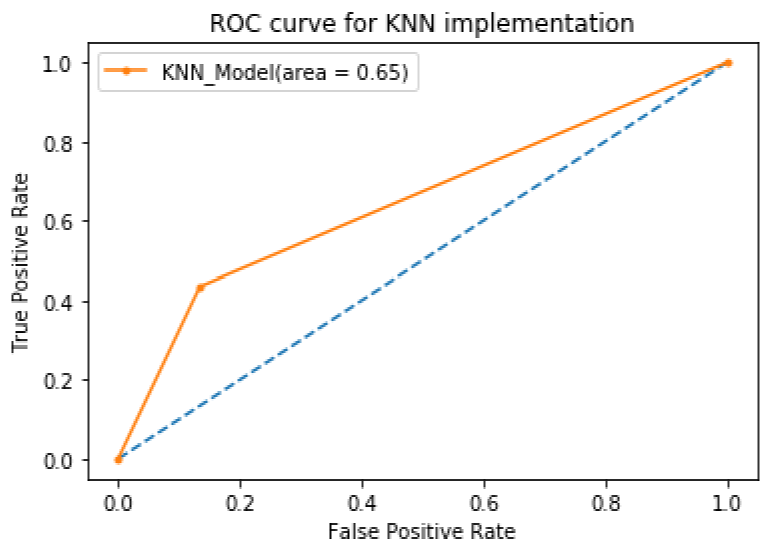
Area under the curve (AUC): 0.683

Code in Python:
Observations:
Confusion Matrix and Statistics
KNN Predictions

```
      0   1
0   142  26
1    22  20
```

- **True positives (TP):** 20
- **True negatives (TN):** 142
- **False positives (FP):** 22
- **False negatives (FN):** 26
- **Accuracy:** (20 + 142)/210 = .7714
- **Misclassification Rate:** (FP+FN)/total = (22+26)/210 = 0.2286
- **True Positive Rate:** TP/actual yes = 20/46 = 0.4347
- **False Positive Rate:** FP/actual no = 22/164= 0.1341
- **True Negative Rate:** TN/actual no = 142/164 = 0.8658
- **False Negative Rate:** FN/actual yes = 26/46 = 0.5652
- **Precision:** TP/predicted yes = 20/42 = 0.4761

ROC Curve:



Result:

R
Accuracy: 78.95%
FNR: 53.70%
ROC: 68.53%

Python
Accuracy: 77.14%
FNR: 56.52%
ROC: 65.03%

Very High False Negative Rate and Accuracy is also low in both languages.

<u>Naive Bayes:</u>

Code in R:

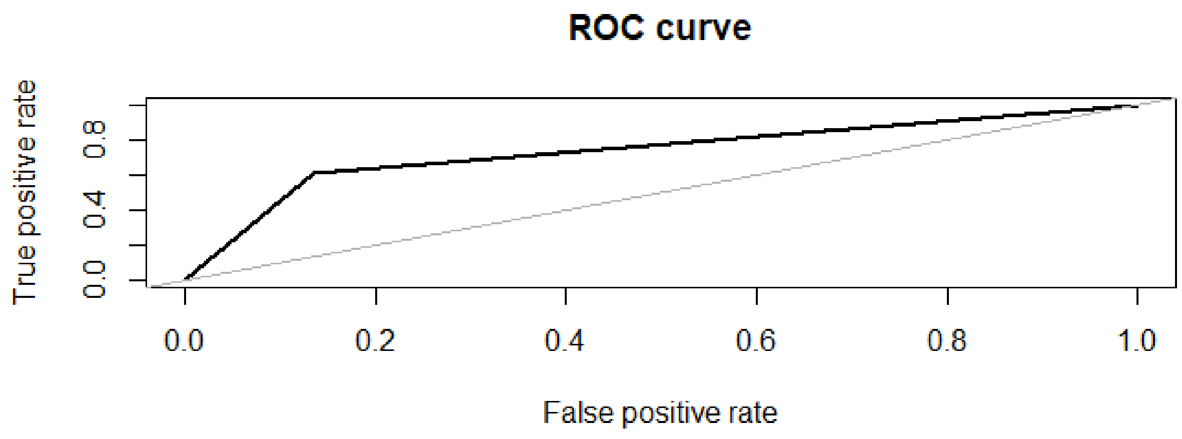
Observations:

Confusion Matrix and Statistics

```
          predicted
observed   0   1
     0    134  21
     1     21  33
```

- **True positives (TP):** 33
- **True negatives (TN):** 134
- **False positives (FP):** 21
- **False negatives (FN):** 21
- **Accuracy:** (21 + 134)/209 = .799
- **Misclassification Rate:** (FP+FN)/total = (21+21)/209 = 0.2009
- **True Positive Rate:** TP/actual yes = 33/54 = 0.6111
- **False Positive Rate:** FP/actual no = 21/155= 0.1354
- **True Negative Rate:** TN/actual no = 134/155 = 0.8645
- **False Negative Rate:** FN/actual yes = 21/54 = 0.3888
- **Precision:** TP/predicted yes = 33/54 = 0.6111
- **Cohen's Kappa:** 0.4756

ROC Curve:



Area under the curve (AUC): 0.738

Code in Python:

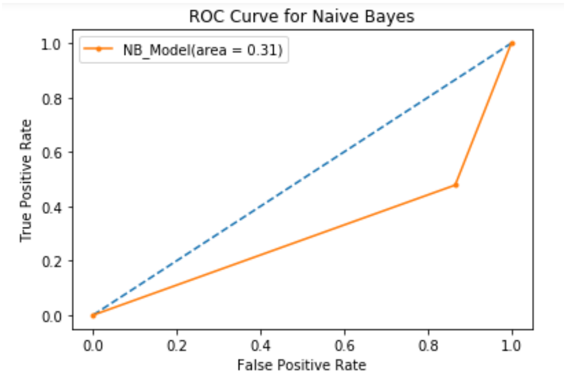
Observations:

Confusion Matrix and Statistics

```
          predicted
observed    0   1
      0     142  22
      1      22  24
```

- **True positives (TP):** 24
- **True negatives (TN):** 142
- **False positives (FP):** 22
- **False negatives (FN):** 22
- **Accuracy:** (24 + 142)/210 = .7904
- **Misclassification Rate:** (FP+FN)/total = (21+21)/209 = 0.2096
- **True Positive Rate:** TP/actual yes = 24/46 = 0.5217
- **False Positive Rate:** FP/actual no = 22/164= 0.1341
- **True Negative Rate:** TN/actual no = 142/164 = 0.8658
- **False Negative Rate:** FN/actual yes = 22/46 = 0.3888
- **Precision:** TP/predicted yes = 24/46 = 0.5217

ROC Curve:



Result:

R	Python
Accuracy: 79.90%	Accuracy: 79.04%
FNR: 38.88%	FNR: 47.82%
ROC: 73.80%	ROC: 30.62%

Lowest False negative Rate is achieved in this model in R language and Accuracy is also fine but Area under ROC curve in Python is very low and overall stability of Random Forest is better than Naïve Bayes, so choosing Random Forest over it.

Conclusion:

Choosing Random Forest Model for Loan Default predictions as it is more stable than other models in both programming languages.