Credit Card Approval Classification and Prediction using R

Ravi Singhal

12/12/21

Data Mining, Machine Learning and Statistical Analysis

Table of Contents

Introduction

In general, customer/user fills out an application and then lender will pull customer's credit report and/or credit score. They'll use this credit profile and other factors, like income or debt-to-income ratio, to determine if customer meet their underwriting standards.

Underwriting standards are benchmarks a lender sets regarding who can qualify for their credit card and at what terms. If customer meet these standards, they will be offered. If not, they will learn why they were rejected.

This requires lenders to manually check all the parameters for each customer daily. This task can be done by the machine once we make it that capable. We will use Data Mining, Machine Learning and Statistical analysis to build a model which will determine if the customer is eligible for the credit based on the standards set by the lenders.

This report consists of techniques we have used to build a model based on the sample dataset to us. We have built the model performed classification and performed the prediction for the sample sets. We have also created decision tree for the lender.

DataSet

Abstract of Data

This data concerns credit card application. This data has been extracted from UCI Machine Learning Repository. The crx.names file contains the details of the attribute's information. The dataset has 690 records and 16 attributes. There are 67 missing values, and the dataset is mix of numerical, continuous, categorical. All attribute names and values have been changed to meaningless symbols to protect confidentiality of the data.

Attribute Information

The variable description are as follows:

1. Sex: Male or Female
2. Age: age in years
3. Debt: debt amount
4. Married: married or unmarried
5. Bank Customer: yes or no
6. Education Level: Qualified, Underqualified, overqualified .....
7. Ethnicity: American Indian, Latino or Hispanic, Asian – Japanese ....
8. Years Employed: number of years
9. Prior Default: yes or no
10. Employed: yes or no
11. Credit Score: Credit score ranging from 0 to 67
12. Driver's License: yes or no
13. Citizen: yes or no
14. Zipcode: zip codes of different areas
15. Income: ranging from 0 to 100000

Attribute Information

The target variable is the binary value showing whether the load will be Approved or not. Also, the data is well distributed:

Approved: + and –

+: 307 (44.5%)

-: 383 (55.5%)

Reading data

We will start with loading the library and then the dataset which we have downloaded from UCI.

Loading Library

```
library(dplyr)
library(tidyverse)
library(tidyr)
library(superml)
library(caTools)
library(MASS)
library(pscl)
library(car)
library(InformationValue)
library(caret)
library(rpart)
library(rpart.plot)
library(C50)
library(ggplot2)
library(GGally)
```

Reading Data from txt file

```
data = read.table("C:\\Users\\ravis\\OneDrive\\Desktop\\R\\credit.data", sep = ",", na.strings = "?")
names(data) <- c('Sex', 'Age', 'Debt', 'Married', 'BankCust', 'EducLevel', 'Ethnicity', 'YearsEmployed',
'PriorDefault', 'Employed', 'CreditScore', 'DLicense', 'Citizen', 'ZipCode', 'Income', 'Approved')
```

>head(data)

```
  Sex   Age  Debt Married BankCust EducLevel Ethnicity YearsEmployed
1   b 30.83 0.000       u        g         w         v          1.25
2   a 58.67 4.460       u        g         q         h          3.04
3   a 24.50 0.500       u        g         q         h          1.50
4   b 27.83 1.540       u        g         w         v          3.75
5   b 20.17 5.625       u        g         w         v          1.71
6   b 32.08 4.000       u        g         m         v          2.50


  PriorDefault Employed CreditScore DLicense Citizen ZipCode Income Approved
1            t        t           1        f       g     202      0        +
2            t        t           6        f       g      43    560        +
3            t        f           0        f       g     280    824        +
4            t        t           5        t       g     100      3        +
5            t        f           0        f       s     120      0        +
6            t        f           0        t       g     360      0        +
```

The dimensions of data shows that there are 690 observations and 16 attributes.

>dim(data)

690 16

>str(data)

```
'data.frame':   690 obs. of  16 variables:
 $ Sex           : chr  "b" "a" "a" "b" ...
 $ Age           : num  30.8 58.7 24.5 27.8 20.2 ...
 $ Debt          : num  0 4.46 0.5 1.54 5.62 ...
 $ Married       : chr  "u" "u" "u" "u" ...
 $ BankCust      : chr  "g" "g" "g" "g" ...
 $ EducLevel     : chr  "w" "q" "q" "w" ...
 $ Ethnicity     : chr  "v" "h" "h" "v" ...
 $ YearsEmployed: num  1.25 3.04 1.5 3.75 1.71 ...
 $ PriorDefault : chr  "t" "t" "t" "t" ...
 $ Employed      : chr  "t" "t" "f" "t" ...
 $ CreditScore   : int  1 6 0 5 0 0 0 0 0 0 ...
 $ DLicense      : chr  "f" "f" "f" "t" ...
 $ Citizen       : chr  "g" "g" "g" "g" ...
 $ ZipCode       : int  202 43 280 100 120 360 164 80 180 52 ...
 $ Income        : int  0 560 824 3 0 0 31285 1349 314 1442 ...
 $ Approved      : chr  "+" "+" "+" "+" ...
```

Data Cleaning

There are various techniques used for cleaning the data. We have used below for our data:

1. Data Transformation
2. Missing Value Treatment
3. Feature Selection
4. Noisy data Treatment

Data Transformation:

Manual:

We are now going to manually convert the character data into numerical data. Also, the target variable needs to be converted to factor.

We have used table command to check the count of the values for each category and further using if else statements assigned them 0 and 1.

```
table(data$Approved)

 -   +

383 307
```

```
data$Sex = ifelse (data$Sex == "a",0,1)

data$Married = ifelse (data$Married != "u", 1,0)

data$BankCust = ifelse (data$BankCust != "g", 1, 0)

data$PriorDefault = ifelse (data$PriorDefault == "f", 0, 1)

data$Employed = ifelse (data$Employed == "f", 0, 1)

data$DLicense = ifelse (data$DLicense == "f", 0, 1)

data$Citizen = ifelse (data$Citizen != "g",1, 0)

data$Approved = ifelse (data$Approved == "-",0,1)
```

***Automatically*:**

We will now convert Ethnicity and EducLevel into numeric automatically using superml library.

```
label <- LabelEncoder$new ()

data$Ethnicity<- label$fit_transform(data$Ethnicity)

table(data$Ethnicity)

 0  1  2  3  4  5  6  7  8

408 138  59  57   8   8   2   6   4

-------------------------------------------------

>data$EducLevel<- label$fit_transform(data$EducLevel)

> table(data$EducLevel)

 0  1  2  3  4  5  6  7  8  9 10 11 12 13

 64  78  38   3  41  51 137  30  38  68  25  54  53  10


data$Approved = as.factor(data$Approved)
```

We have now done with data transformation, now we will see the structure of the data:

str(data)

```
'data.frame':   690 obs. of  16 variables:
 $ Sex           : num  1 0 0 1 1 1 1 0 1 1 ...
 $ Age           : num  30.8 58.7 24.5 27.8 20.2 ...
 $ Debt          : num  0 4.46 0.5 1.54 5.62 ...
 $ Married       : num  0 0 0 0 0 0 0 0 1 1 ...
 $ BankCust      : num  0 0 0 0 0 0 0 0 1 1 ...
 $ EducLevel     : num  0 1 1 0 0 2 3 4 5 0 ...
 $ Ethnicity     : num  0 1 1 0 0 0 1 0 1 0 ...
 $ YearsEmployed: num  1.25 3.04 1.5 3.75 1.71 ...
 $ PriorDefault : num  1 1 1 1 1 1 1 1 1 1 ...
 $ Employed      : num  1 1 0 1 0 0 0 0 0 0 ...
 $ CreditScore   : num  1 6 0 5 0 0 0 0 0 0 ...
 $ DLicense      : num  0 0 0 1 0 1 1 0 0 1 ...
 $ Citizen       : num  0 0 0 0 1 0 0 0 0 0 ...
 $ ZipCode       : num  202 43 280 100 120 360 164 80 180 52 ...
 $ Income        : num  0 560 824 3 0 ...
 $ Approved      : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
```

Missing Value Treatment:

Upon analyzing the data, we found that missing values in the data were represented by question mark (?). Hence, we have converted all these to NA while reading the dataset.

We can now easily calculate the NA values in the data frame. There are total 67 missing values.

```
> cbind (lapply (lapply (data, is.na), sum))
```

	[,1]
Sex	12
Age	12
Debt	0
Married	6
BankCust	6
EducLevel	9
Ethnicity	9
YearsEmployed	0
PriorDefault	0
Employed	0
CreditScore	0
DLicense	0
Citizen	0
ZipCode	13
Income	0
Approved	0

Upon analyzing the data, we found that the best way to fill the missing data is by taking median. There are attributes like Zipcode, Sex, Married, Education Level, Ethnicity & Bank Customer having values which will not give appropriate result by taking mean.

We will use for loop to update all the column's missing value with the median.

```
> for (i in 1: ncol(data)) {
  data [, i][is.na (data[, i])] <- median(data[,i], na.rm = TRUE)
}
```

After running the loop, we will confirm if there exist any NA values.

```
> cbind (lapply (lapply (data, is.na), sum))
```

	[,1]
Sex	0
Age	0
Debt	0
Married	0
BankCust	0
EducLevel	0
Ethnicity	0
YearsEmployed	0
PriorDefault	0
Employed	0
CreditScore	0
DLicense	0
Citizen	0
ZipCode	0
Income	0
Approved	0

We were able to successfully, fill all the values with the median value. Since we have done Data Transformation and Missing value treatment this may lead to some outliers which we need to fix. But before that we will remove 2 attributes which have no usage in our model.

Feature Selection:

We are discarding 2 attributes which does not help us further. In order to achieve this, we are using dplyr library.

```
data1 = data %>% dplyr::select (-ZipCode, -BankCust)
```

```
str(data1)
```

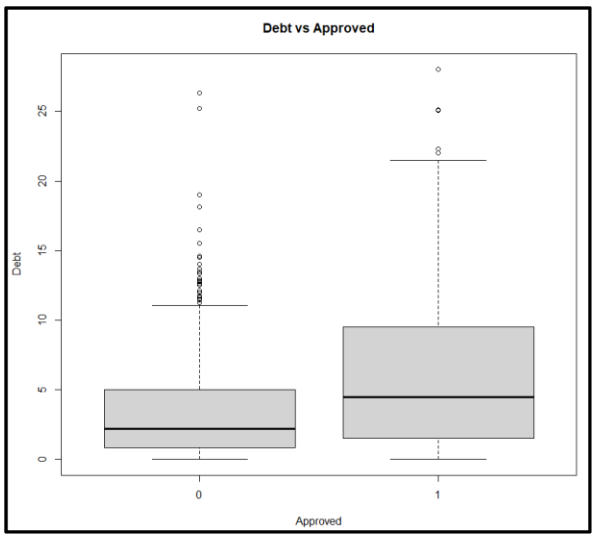
```
'data.frame':   690 obs. of  14 variables:
 $ Sex           : num  1 0 0 1 1 1 1 0 1 1 ...
 $ Age           : num  30.8 58.7 24.5 27.8 20.2 ...
 $ Debt          : num  0 4.46 0.5 1.54 5.62 ...
 $ Married       : num  0 0 0 0 0 0 0 0 1 1 ...
 $ EducLevel     : num  0 1 1 0 0 2 3 4 5 0 ...
 $ Ethnicity     : num  0 1 1 0 0 0 1 0 1 0 ...
 $ YearsEmployed: num  1.25 3.04 1.5 3.75 1.71 ...
 $ PriorDefault : num  1 1 1 1 1 1 1 1 1 1 ...
 $ Employed      : num  1 1 0 1 0 0 0 0 0 0 ...
 $ CreditScore   : num  1 6 0 5 0 0 0 0 0 0 ...
 $ DLicense      : num  0 0 0 1 0 1 1 0 0 1 ...
 $ Citizen       : num  0 0 0 0 1 0 0 0 0 0 ...
 $ Income        : num  0 560 824 3 0 ...
 $ Approved      : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
```

Noisy Data Treatment:

Outlier is a value that lies in a data set on its extremes, which is either very small or large and thus can affect the overall observation made from the data sets. Outliers are also termed as extremes because they lie on the end of a data sets. Outliers are usually treated as abnormal values that can affect the overall observation due to its very high or low extreme values and hence, we will be discarding from the data sets.

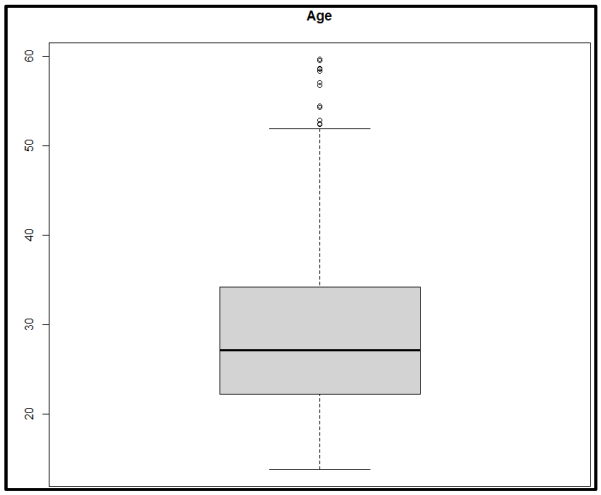
Box plot diagram also termed as Whisker's plot is a graphical method typically depicted by quartiles and inter quartiles that helps in defining the upper limit and lower limit beyond which any data lying will be considered as outliers. The very purpose of this diagram is to identify outliers and discard it from the data sets before making any further observation so that the conclusion made from the study gives more accurate results not influenced by any extremes or abnormal values

```
boxplot (Debt~Approved, main = "Debt vs Approved")
```




```
> summary(Debt)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.000   1.000   2.750   4.759   7.207  28.000
> b = data1$Debt [! data1$Debt %in% boxplot.stats(data1$Debt)$out]
> summary(b)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.000   0.835   2.500   3.973   6.020  14.790
> data1$Debt = ifelse (data1$Debt > max(b), NA, data1$Debt)
```

```
boxplot(data1$Age, main = "Age")
```

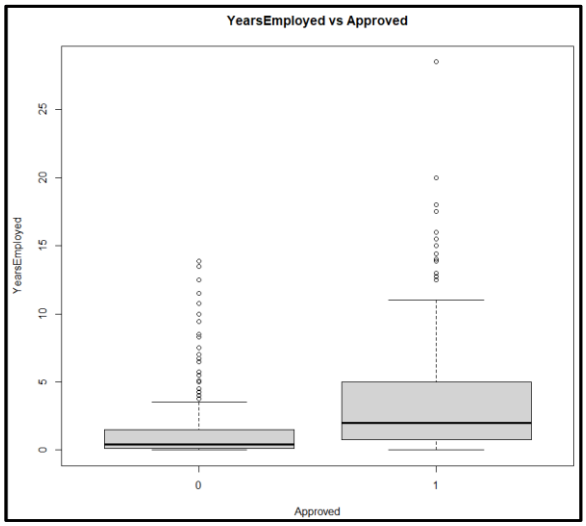


a = data1$Age [!data1$Age %in% boxplot.stats(data1$Age)$out]

data1$Age = ifelse (data1$Age > max(a), NA, data1$Age)

---

boxplot (YearsEmployed~Approved, main = "YearsEmployed vs Approved")

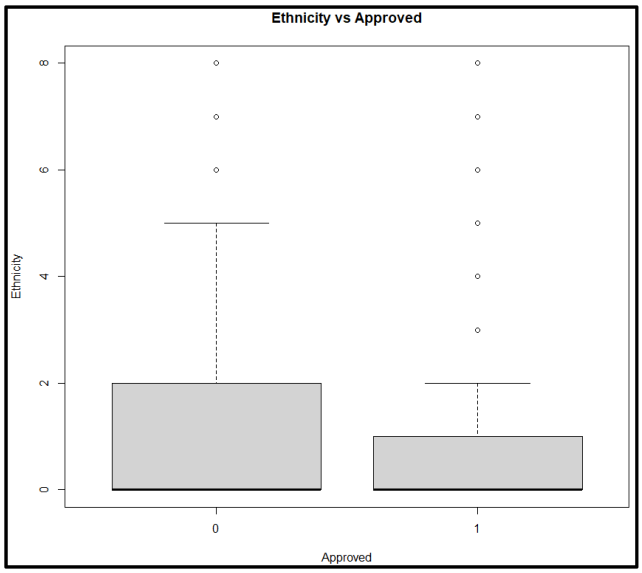


y = data1$YearsEmployed [!data1$YearsEmployed %in% boxplot.stats(data1$YearsEmployed)$out]

data1$YearsEmployed = ifelse (data1$YearsEmployed > max(y), NA, data1$YearsEmployed)

---

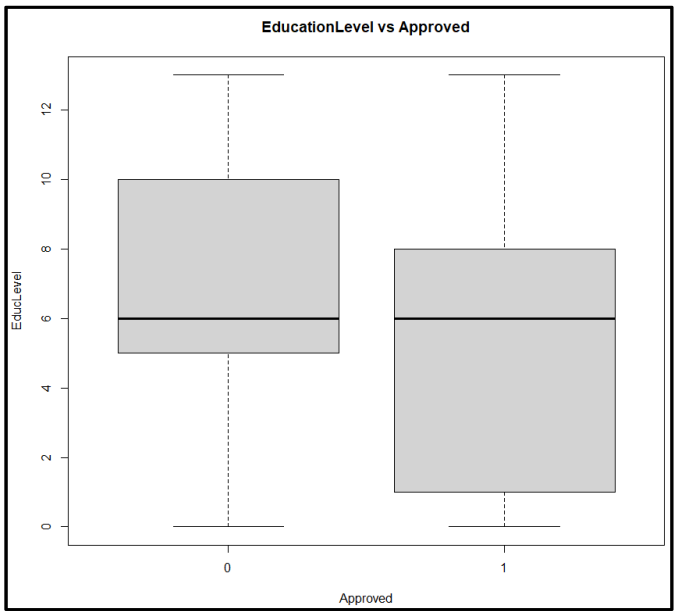
boxplot(data1$Ethnicity~data1$Approved, main = "Ethnicity vs Approved")



e = data1$Ethnicity[!data1$Ethnicity %in% boxplot.stats(data1$Ethnicity)$out]

data1$Ethnicity = ifelse (data1$Ethnicity > max(e), NA, data1$Ethnicity)

> boxplot (EducLevel~Approved, main = "EducationLevel vs Approved")



The EducationLevel boxplot looks perfect without any outliers.

We have detected the outliers, now we will go and drop the NA values which are the extreme values. These droppings of values will reduce the dimension of data to 540 observations.

data1 = data1 %>% drop_na()

>dim(data1)

540 14

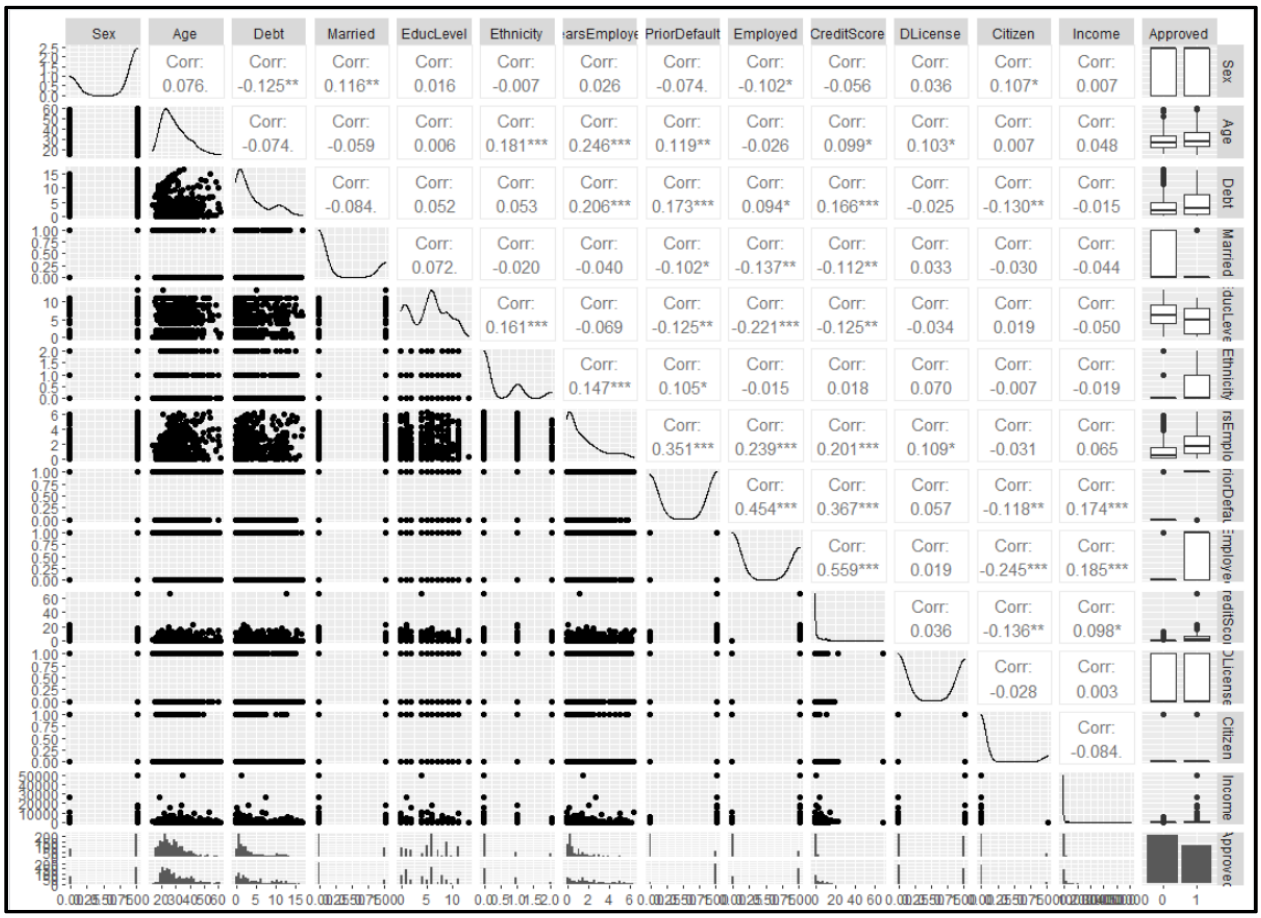
Data Visualization

There are various ways we can visualize the data. We will have a look at the pairplot, histogram and some barplots.

Let's have a look at the pair plot using ggplot library and see how these attributes are correlated to each other.

A **pairs plot** is a matrix of scatterplots that lets you understand the pairwise relationship between different variables in a dataset.

ggpairs(data1)

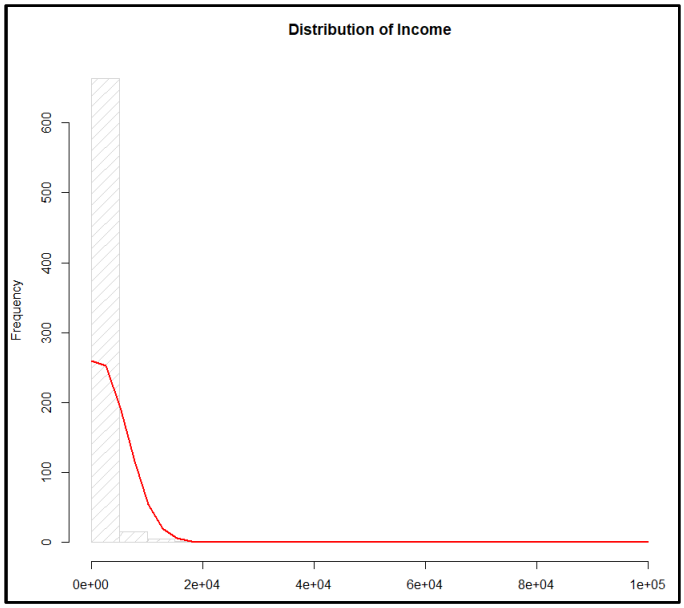


We will see Income attribute since, boxplot show the value above 0 as noise, but we do not want to delete those values. Hence, we are plotting histogram for Income.

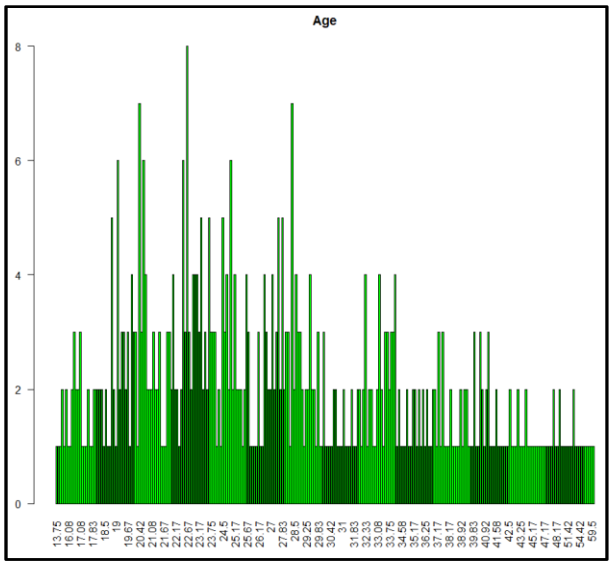
h <- hist (Income, breaks = 25, density = 10, xlab = "Income", main = "Distribution of Income")

xfit <- seq(min(Income), max(Income), length = 40)

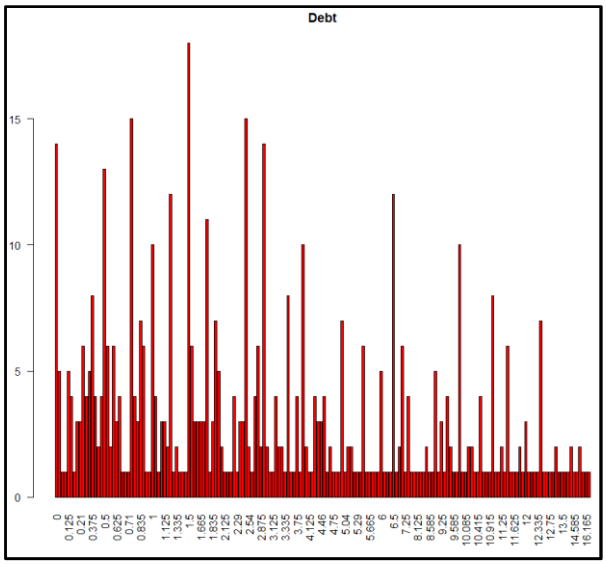
```
yfit <- dnorm (xfit, mean = mean(Income), sd = sd(Income))
yfit <- yfit * diff (h$mids[1:2]) * length(Income)
lines (xfit, yfit, col = "red", lwd = 2)
```




```
counts_a <- table(data1$Age)
barplot(counts_a, las = 2, col = "green", main = "Age")
counts_d <- table(data1$Debt)
barplot(counts_d, las = 2, col = "red", main = "Debt")
```

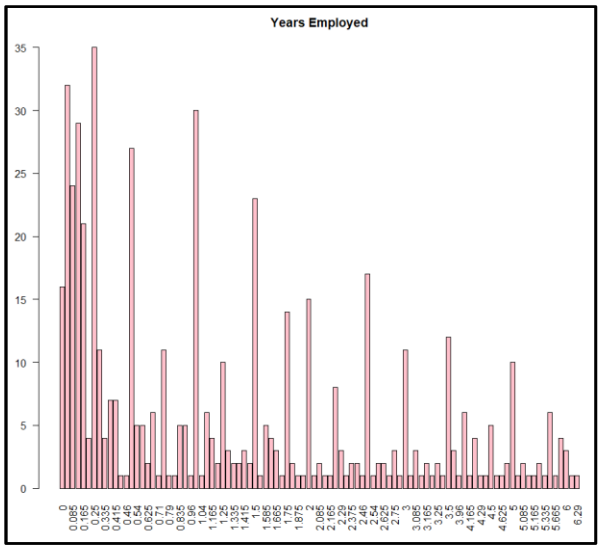


Age

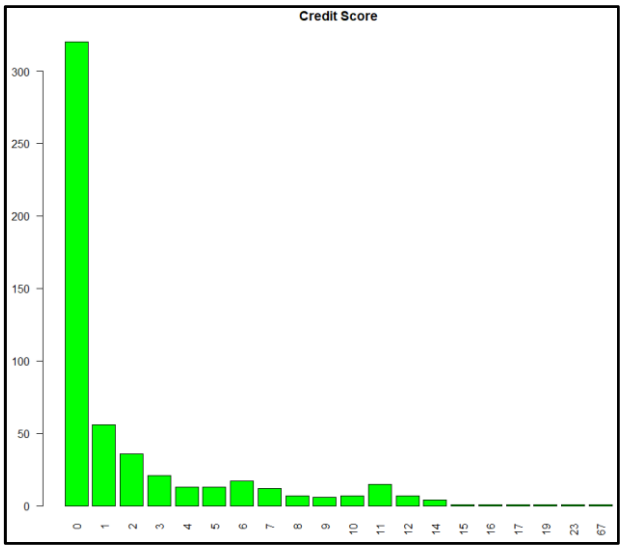


Debt

```
counts_y <- table(data1$YearsEmployed)
barplot(counts_y, las = 2, col = "pink", main = "Years Employed")
counts_c <- table(data1$CreditScore)
barplot(counts_c, las = 2, col = "green", main = "Credit Score")
```

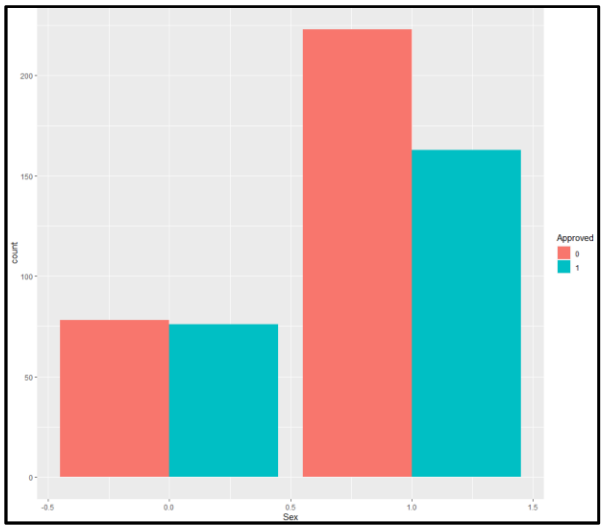


Years Employed

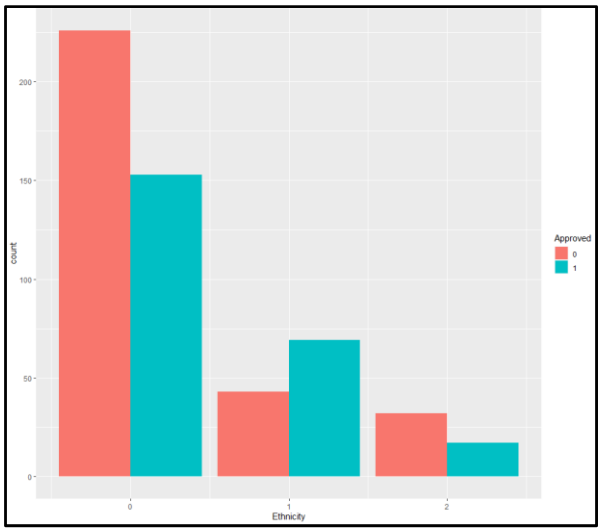


Credit Score

```
ggplot(data = data1) +
  geom_bar(mapping = aes(x = Sex, fill = Approved), position = "dodge")
ggplot(data = data1) +
  geom_bar(mapping = aes(x = Ethnicity, fill = Approved), position = "dodge")
```



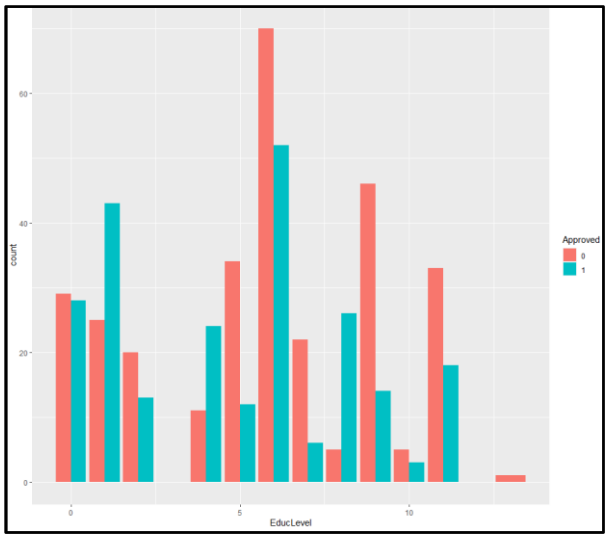
Sex



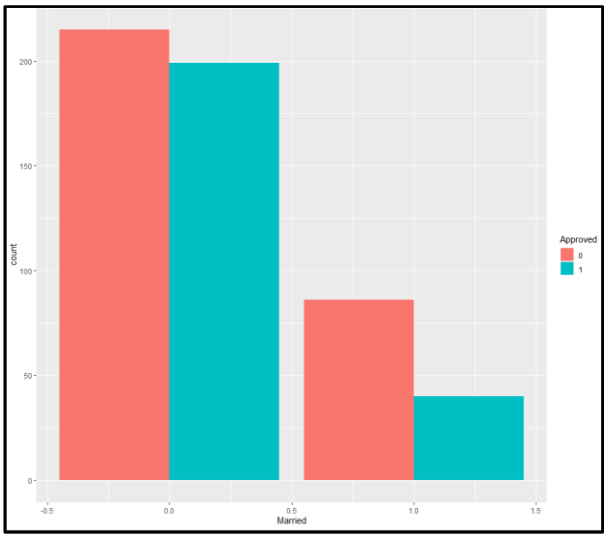
Ethnicity

```
ggplot(data = data1) +
  geom_bar(mapping = aes(x = EducLevel, fill = Approved), position = "dodge")
```

```
ggplot(data = data1) +
  geom_bar(mapping = aes(x = Married, fill = Approved), position = "dodge")
```



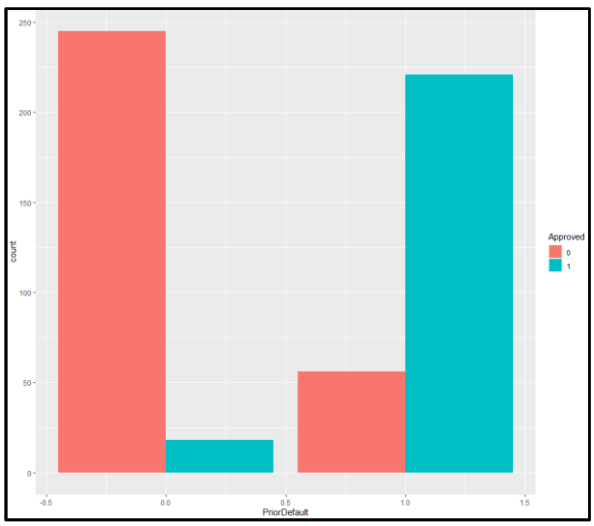
EducLevel



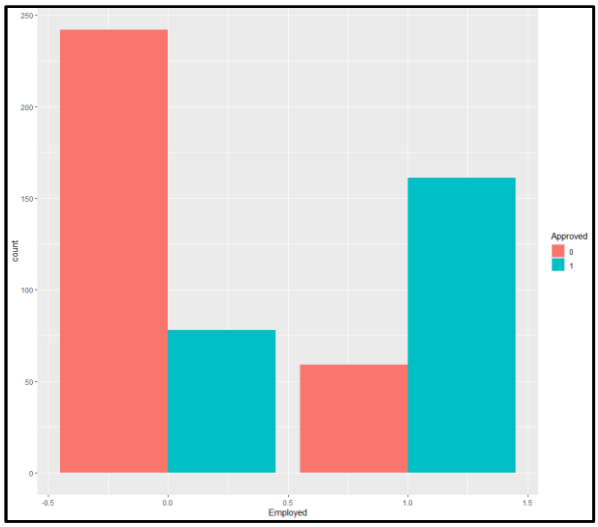
Married

```
ggplot(data = data1) +
  geom_bar(mapping = aes(x = PriorDefault, fill = Approved), position = "dodge")
```

```
ggplot(data = data1) +
  geom_bar(mapping = aes(x = Employed, fill = Approved), position = "dodge")
```

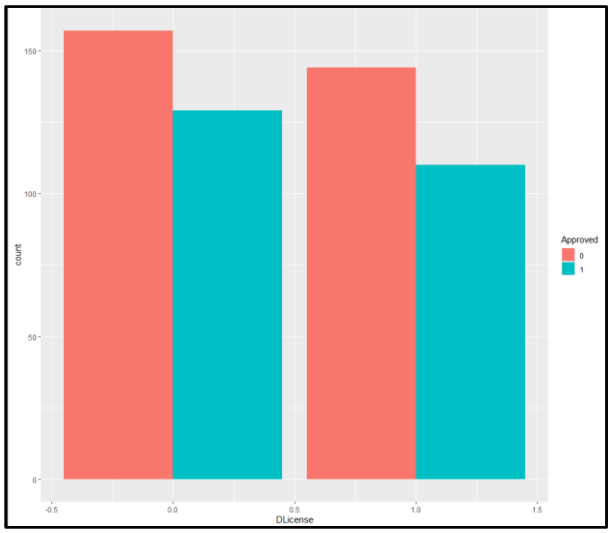


Prior Default

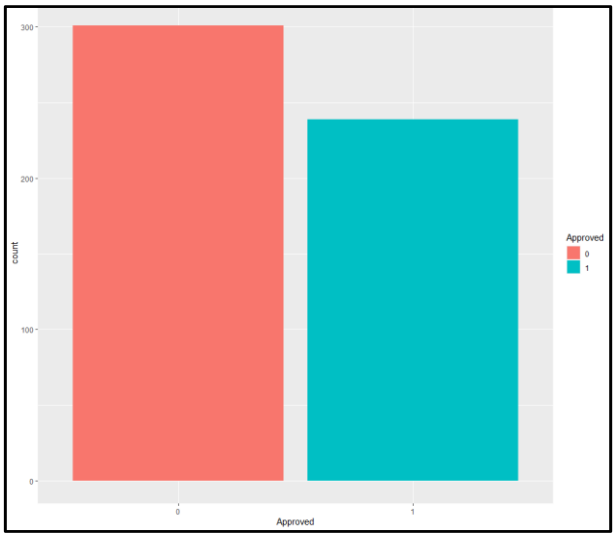


Employed

```
ggplot(data = data1) +
  geom_bar(mapping = aes(x = DLicense, fill = Approved), position = "dodge")
ggplot(data = data1) +
  geom_bar(mapping = aes(x = Approved, fill = Approved), position = "dodge")
```



DLicense



Approved

Model Building & Testing

We have done data cleaning and visualization and now we are going to build the model. But first we need to split the data into train and test so that we can further used test data to check how accurate our model is built.

We split the data into 80:20 ratio. Training datasets consist of 80% i.e., 432 observations, test data consist of 20% i.e., 108 observations.

> split = sample.split(data1$Approved, SplitRatio = 0.8)

> train_data <- subset(data1, split == T)

> test_data = subset(data1, split == F)

Logistic regression models are models that have a certain fixed number of parameters that depend on the number of input features, and they output categorical prediction, like for example Yes or No, 0 or 1.

We will now build a logistic remodel with all the attributes:

model = glm(formula = Approved~., family = "binomial", data = train_data)

summary(model)

```
Call:
glm(formula = Approved ~ ., family = "binomial", data = train_data)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.1293  -0.3755  -0.2167   0.4863   2.8474

Coefficients:
                Estimate Std. Error z value Pr(>|z|)
(Intercept)   -2.9747767  0.7604746  -3.912 9.16e-05 ***
Sex           -0.3416019  0.3446319  -0.991   0.3216
Age            0.0215290  0.0192166   1.120   0.2626
Debt          -0.0128461  0.0395381  -0.325   0.7453
Married       -0.4676815  0.3773120  -1.240   0.2152
EducLevel     -0.0446395  0.0447597  -0.997   0.3186
Ethnicity     -0.2178400  0.2400097  -0.908   0.3641
YearsEmployed  0.2487026  0.1114647   2.231   0.0257 *
PriorDefault   3.4962977  0.3649138   9.581  < 2e-16 ***
Employed       0.4656647  0.4593269   1.014   0.3107
CreditScore    0.1806653  0.0895285   2.018   0.0436 *
DLicense      -0.7957757  0.3218887  -2.472   0.0134 *
Citizen        0.6551964  0.5414977   1.210   0.2263
Income         0.0004818  0.0001948   2.473   0.0134 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 593.08  on 431  degrees of freedom
Residual deviance: 279.58  on 418  degrees of freedom
AIC: 307.58

Number of Fisher Scoring iterations: 6
```

After looking at the model, we found that all the attributes were not contributing. Hence, we decided to use Akaike's ‘An Information Criterion (AIC).

The AIC will take each model and rank them from best to worst. The “best” model will be the one that neither under-fits nor over-fits.

AIC = -2(log-likelihood) + 2K

```
Step:  AIC=295.02
Approved ~ YearsEmployed + PriorDefault + Employed + CreditScore +
    DLicense + Income

                Df Deviance    AIC
<none>               281.02 295.02
+ Ethnicity      1   279.19 295.19
+ Married        1   279.70 295.70
- Employed       1   283.79 295.79
+ Citizen        1   280.12 296.12
+ EducLevel      1   280.59 296.59
- YearsEmployed  1   284.75 296.75
+ Age            1   280.80 296.80
+ Sex            1   280.80 296.80
+ Debt           1   280.85 296.85
- DLicense       1   286.44 298.44
- CreditScore    1   286.72 298.72
- Income         1   291.42 303.42
- PriorDefault   1   390.06 402.06
```

We were able to reduce the AIC of model from 307 to 295

summary(model_1)

```
Call:
glm(formula = Approved ~ YearsEmployed + PriorDefault + Employed +
    CreditScore + DLicense + Income, family = "binomial", data = train_data)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.7218  -0.3623  -0.2341   0.4670   2.7260

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)  -2.9663592  0.3454970  -8.586   <2e-16 ***
YearsEmployed  0.1991590  0.1042842   1.910   0.0562 .
PriorDefault   3.1868409  0.3585816   8.887   <2e-16 ***
Employed       0.7380546  0.4411207   1.673   0.0943 .
CreditScore    0.1939689  0.0973584   1.992   0.0463 *
DLicense      -0.7248436  0.3165587  -2.290   0.0220 *
Income         0.0004368  0.0001809   2.415   0.0157 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 593.08  on 431  degrees of freedom
Residual deviance: 281.02  on 425  degrees of freedom
AIC: 295.02

Number of Fisher Scoring iterations: 6
```

We will use varImp, which helps us in calculation of variable importance for model on training data.

> caret::varImp(model_1)

	Overall
YearsEmployed	1.909772
PriorDefault	8.887351
Employed	1.673135
CreditScore	1.992317
DLicense	2.289761
Income	2.414659

Since we don't get R^2 value from logistic model, so we will get McFadden value.

> pscl::pR2(model_1)["McFadden"]

```
fitting null model for pseudo-r2
 McFadden
0.5261652
```

Variance inflation factor (VIF) is a measure of the amount of multicollinearity in a set of multiple regression variables. Mathematically, the VIF for a regression model variable is equal to the ratio of the overall model variance to the variance of a model that includes only that single independent variable.

We need to check if there is any multicollinearity in our model.

> car::vif(model_1)

YearsEmployed	PriorDefault	Employed	CreditScore	DLicense	Income
1.061850	1.085531	1.913654	1.978278	1.066354	1.012477

As per the standard if the VIF value is > 10, then there exists high multicollinearity. Since our values are in range of 1, it indicates we are in good shape and can proceed with the logistic regression.

We have verified our model which was built on training data and all the constraint seems good. Hence, we will begin with the prediction on the test data.

pred <- predict(model_1, test_data, type = "response")

optimal <- optimalCutoff(test_data$Approved, pred)

> optimal

0.3648302

After running the prediction on the test data, we tried to calculate the optimal cutoff value which came out to be 0.36. As per the standard we use 0.5 but to get the good result we will use the optimal cutoff obtained by us.

> pred_t = as.factor(ifelse(pred > 0.36, "1","0"))

We will draw the confusion matrix to see how the model performed on the test data.

> confusionMatrix(test_data$Approved, pred_t)

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 53  7
         1  5 43

               Accuracy : 0.8889
                 95% CI : (0.814, 0.9413)
    No Information Rate : 0.537
    P-Value [Acc > NIR] : 3.789e-15

                  Kappa : 0.7759

 Mcnemar's Test P-Value : 0.7728

            Sensitivity : 0.9138
            Specificity : 0.8600
         Pos Pred Value : 0.8833
         Neg Pred Value : 0.8958
             Prevalence : 0.5370
         Detection Rate : 0.4907
   Detection Prevalence : 0.5556
      Balanced Accuracy : 0.8869

       'Positive' Class : 0
```

The accuracy for the model turned out to be ~ **89%** which looks good.

Though, we have obtained the good accuracy, we will now check few parameters which are very essential. There are certain cases which need to be considered as per the bank perspective. We will first look at what those parameters are :

	Pred Approved	Pred Not Approved
Actual Approved	TP	FN
Actual Not Approved	FP	TN

Sensitivity (True Positive Rate) = TP / (TP + FN)

Specificity (False Positive Rate) = FP / (TN + FP)

We calculated Sensitivity and Specificity of the model. Upon checking, we found that both sensitivity and specificity are 91% an 86% respectively which indicates a good percentage. We checked the misclassification values and it turned out to be 11%.

We can say that our model will not let bank issue (90% cases) the credit card to the person who is not capable of clearing all his accounts.

sensitivity(test_data$Approved,pred_t)

0.9137

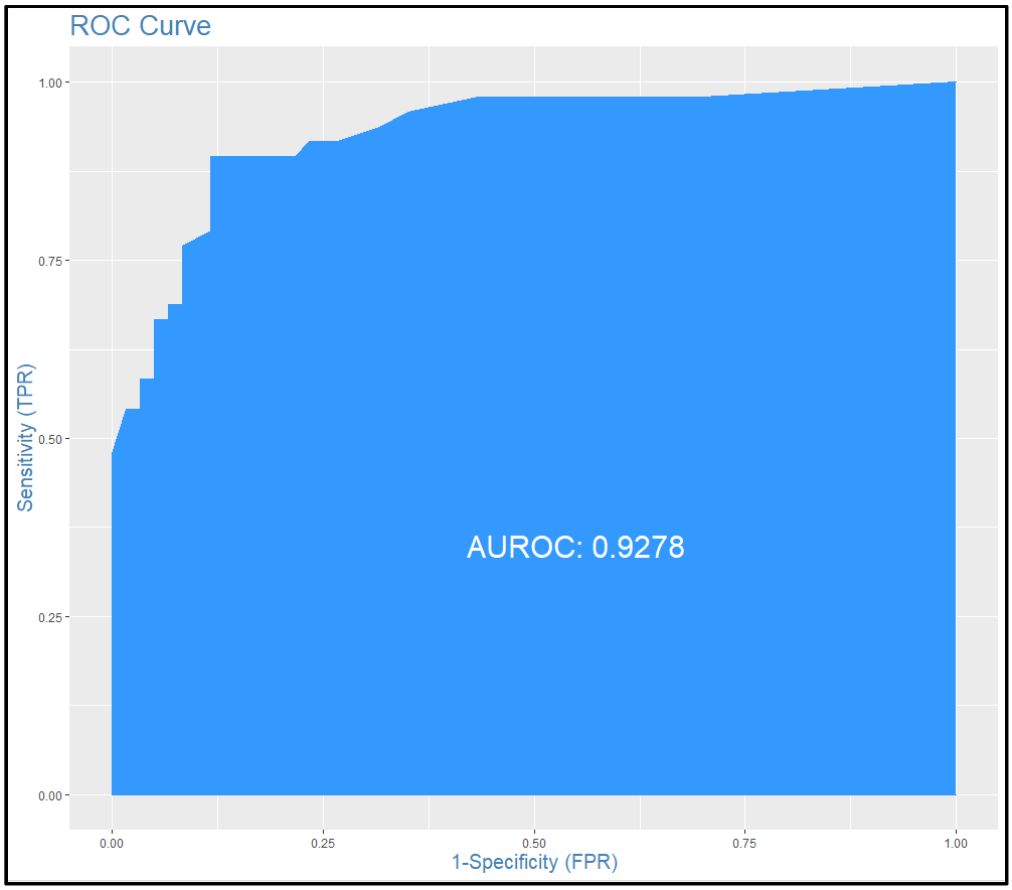
specificity(test_data$Approved, pred_t)

0.86

misClassError(test_data$Approved, pred, threshold=optimal)

0.1111

An ROC curve (receiver operating characteristic curve) is a graph showing the performance of a classification model at all classification thresholds. The curve plots Sensitivity vs Specificity. We got 92% under the Curve.



Classification & Prediction

Prediction and classification usually consist of building a decision tree or a ruleset. We will use rpart and C5 tool to perform classification and prediction. Further, we will analyze the ruleset and the decision tree generated by Rpart and C5.

rpart

rpart is a powerful machine learning library in R that is used for building classification and regression trees. This library implements recursive partitioning. Hence, we are going to create a classification tree for the dataset. We are setting the seed and then splitting the data into training and testing for model to test.

```
set.seed(123)

training.samples <- data1$Approved %>%

        createDataPartition(p = 0.8, list = FALSE)

train.data  <- data1[training.samples, ]

test.data <- data1[-training.samples, ]
```

Once, we have the data we are now going to train the model using Rpart which will select the attributes based on recursive partitioning technique and then we are going to plot the tree.

```
md1 <- rpart(Approved ~., data = train.data, method = "class")

summary(md1)
```

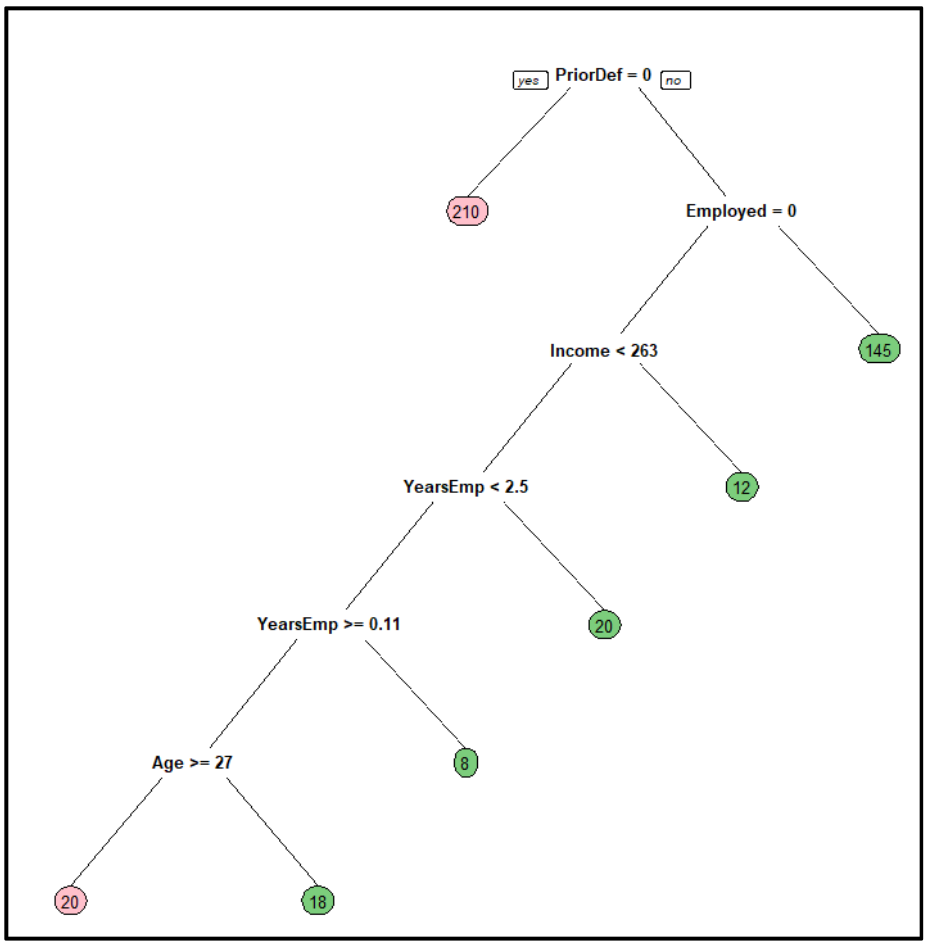
```
Call:
rpart(formula = Approved ~ ., data = train.data, method = "class")
  n= 433

          CP nsplit rel error    xerror       xstd
1 0.67187500      0 1.0000000 1.0000000 0.05384110
2 0.01388889      1 0.3281250 0.3281250 0.03821430
3 0.01041667      5 0.2656250 0.3593750 0.03966706
4 0.01000000      6 0.2552083 0.3541667 0.03943263

Variable importance
 PriorDefault   CreditScore      Employed YearsEmployed        Income          Debt           Age       Citizen
           32            18            18            14            10             6             1             1

Node number 1: 433 observations,    complexity param=0.671875
  predicted class=0  expected loss=0.443418  P(node) =1
    class counts:   241   192
   probabilities: 0.557 0.443
  left son=2 (210 obs) right son=3 (223 obs)
  Primary splits:
      PriorDefault  < 0.5    to the left,  improve=109.97720, (0 missing)
      CreditScore   < 2.5    to the left,  improve= 60.04257, (0 missing)
      Employed      < 0.5    to the left,  improve= 52.87992, (0 missing)
      Income        < 252    to the left,  improve= 30.86336, (0 missing)
      YearsEmployed < 1.27   to the left,  improve= 27.88778, (0 missing)
  Surrogate splits:
      Employed      < 0.5    to the left,  agree=0.732, adj=0.448, (0 split)
      CreditScore   < 0.5    to the left,  agree=0.732, adj=0.448, (0 split)
      YearsEmployed < 0.605  to the left,  agree=0.707, adj=0.395, (0 split)
      Income        < 156    to the left,  agree=0.633, adj=0.243, (0 split)
      Debt          < 1.395  to the left,  agree=0.594, adj=0.162, (0 split)
```

```
Node number 25: 20 observations
  predicted class=1  expected loss=0.35  P(node) =0.04618938
    class counts:     7    13
   probabilities: 0.350 0.650

Node number 48: 38 observations,    complexity param=0.01041667
  predicted class=0  expected loss=0.3421053  P(node) =0.08775982
    class counts:    25    13
   probabilities: 0.658 0.342
  left son=96 (20 obs) right son=97 (18 obs)
  Primary splits:
      Age           < 26.625 to the right, improve=3.1163740, (0 missing)
      EducLevel     < 8.5    to the right, improve=1.5909770, (0 missing)
      Debt          < 2.5625 to the left,  improve=0.9830409, (0 missing)
      DLicense      < 0.5    to the right, improve=0.5029904, (0 missing)
      YearsEmployed < 1.4375 to the right, improve=0.3389796, (0 missing)
  Surrogate splits:
      Debt          < 8.25   to the left,  agree=0.632, adj=0.222, (0 split)
      Ethnicity     < 1.5    to the right, agree=0.579, adj=0.111, (0 split)
      YearsEmployed < 1.6475 to the left,  agree=0.579, adj=0.111, (0 split)
      Citizen       < 0.5    to the right, agree=0.579, adj=0.111, (0 split)
      Married       < 0.5    to the left,  agree=0.553, adj=0.056, (0 split)

Node number 49: 8 observations
  predicted class=1  expected loss=0.25  P(node) =0.01847575
    class counts:     2     6
   probabilities: 0.250 0.750

Node number 96: 20 observations
  predicted class=0  expected loss=0.15  P(node) =0.04618938
    class counts:    17     3
   probabilities: 0.850 0.150

Node number 97: 18 observations
  predicted class=1  expected loss=0.4444444  P(node) =0.04157044
    class counts:     8    10
   probabilities: 0.444 0.556
```

We can see the model built by Rpart has one different attribute selection compared to what we obtained while building logistic model.

We will now plot the decision tree from the rules which has been generated while the model has been built.

```
only_count <- function(x, labs, digits, varlen) {
        paste(x$frame$n)}
boxcols <- c("pink", "palegreen3")[md1$frame$yval]
par(xpd=TRUE)
prp(md1, faclen = 0, cex = 0.8, node.fun=only_count, box.col = boxcols)
```



We will now predict the test data and see how different the accuracy of the model turns to be for the same set of data.

```
predicted.classes <- md1 %>%
+   predict(test.data, type = "class")
```

```
confusionMatrix(predicted.classes, test.data$Approved)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 53  4
         1  7 43

               Accuracy : 0.8972
                 95% CI : (0.8235, 0.9476)
    No Information Rate : 0.5607
    P-Value [Acc > NIR] : 3.231e-14

                  Kappa : 0.7927

 Mcnemar's Test P-Value : 0.5465

            Sensitivity : 0.8833
            Specificity : 0.9149
         Pos Pred Value : 0.9298
         Neg Pred Value : 0.8600
             Prevalence : 0.5607
         Detection Rate : 0.4953
   Detection Prevalence : 0.5327
      Balanced Accuracy : 0.8991

       'Positive' Class : 0
```

We found the accuracy for the model turned out to be 89.72% which is almost equal to what we got while we build the logistic regression model.

C5.0

The C50 package contains an interface to the C5.0 classification model. The main two modes for this model are:

- a basic tree-based model
- a rule-based model

C5.0 builds decision trees from a set of training data in the same way as ID3, using the concept of Information entropy. To fit a simple classification tree model, we will build basic tree-based model:

```
t_data_x = data1[1:450,1:13]

t_data_y = data1[1:450,14]

te_data_x = data1[451:540,1:13]

te_data_y = data1[451:540,14]

model_c <- C50::C5.0( t_data_x, t_data_y )

summary( model_c )
```

```
Call:
C5.0.default(x = t_data_x, y = t_data_y)


C5.0 [Release 2.07 GPL Edition]         Sat Dec 11 22:19:17 2021
-------------------------------

Class specified by attribute `outcome'

Read 450 cases (14 attributes) from undefined.data

Decision tree:

PriorDefault <= 0: 0 (186/13)
PriorDefault > 0:
:...Employed > 0: 1 (165/14)
    Employed <= 0:
    :...Ethnicity > 1:
        :...Age <= 27.25: 1 (2)
        :   Age > 27.25: 0 (10/1)
        Ethnicity <= 1:
        :...Income > 258: 1 (17)
            Income <= 258:
            :...Ethnicity > 0: 1 (19/4)
                Ethnicity <= 0:
                :...Married > 0: 0 (12/3)
                    Married <= 0:
                    :...Income > 70: 1 (2)
                        Income <= 70:
                        :...Sex <= 0:
                            :...YearsEmployed <= 1.665: 1 (6)
                            :   YearsEmployed > 1.665: 0 (4/1)
                            Sex > 0:
                            :...YearsEmployed > 1.415: 1 (8/1)
                                YearsEmployed <= 1.415:
                                :...Age <= 45: 0 (17/4)
                                    Age > 45: 1 (2)
```

```
Evaluation on training data (450 cases):

            Decision Tree
          ----------------
          Size      Errors

            13   41( 9.1%)   <<


           (a)   (b)    <-classified as
          ----  ----
           207    19    (a): class 0
            22   202    (b): class 1


        Attribute usage:

        100.00% PriorDefault
         58.67% Employed
         22.00% Ethnicity
         19.33% Income
         11.33% Married
          8.22% Sex
          8.22% YearsEmployed
          6.89% Age
```

We found that accuracy on the training data is just 91.9%. Hence, we decided to use trails which enables boosting procedure. Boosting is an idea of filtering or weighting the data that is used to train our weak learners, so that each new learner gives more weight or is only trained with observations that have been poorly classified by the previous learners

model_c1 <- C50::C5.0(t_data_x, t_data_y, trials=10)

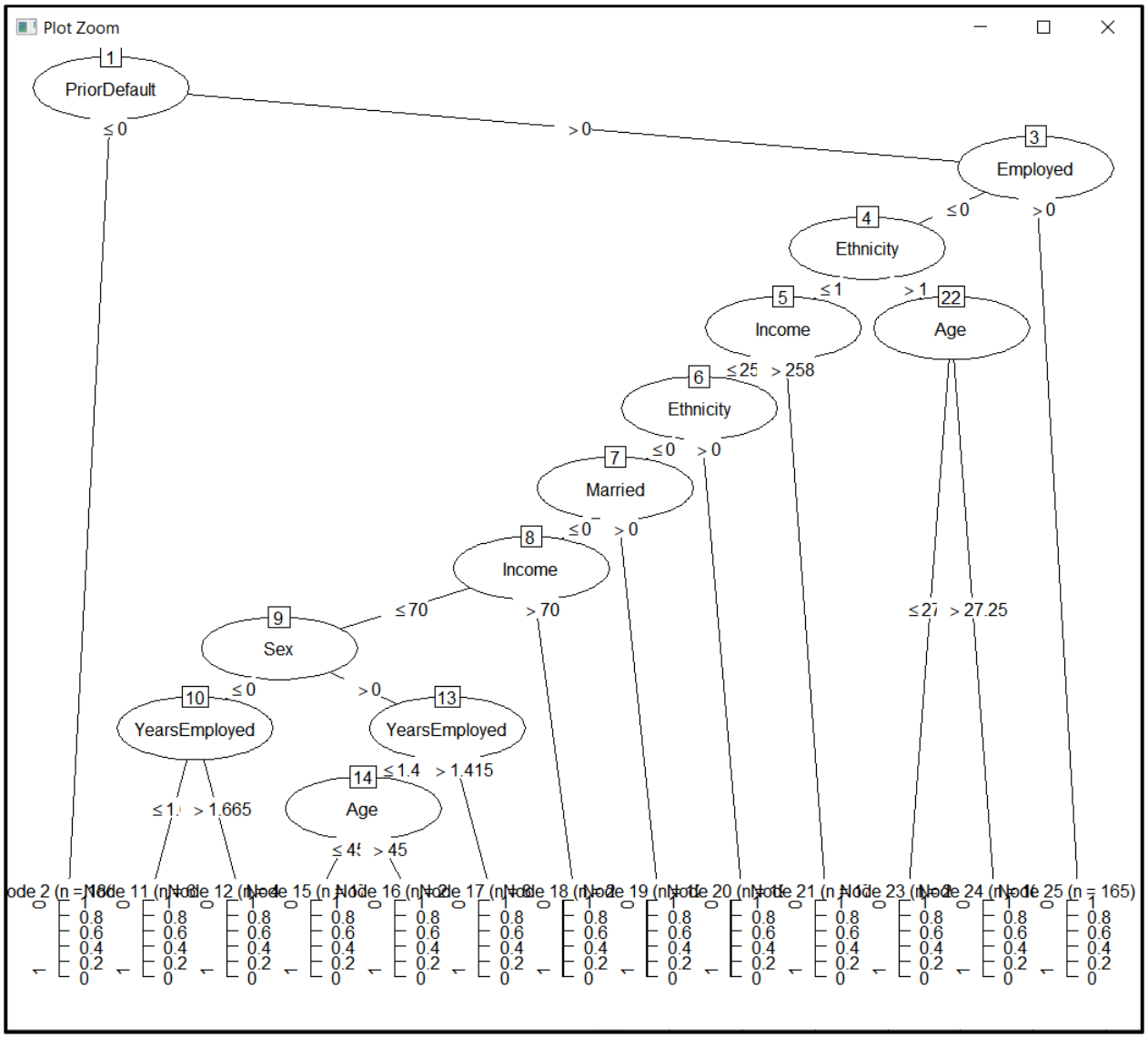
summary(model_c1)

```
-----  Trial 9:  -----

Decision tree:

PriorDefault <= 0: 0 (122.2/21.5)
PriorDefault > 0:
:...Income > 228: 1 (45.3/2)
    Income <= 228:
    :...Age <= 19.75: 0 (33.1/9.8)
        Age > 19.75:
        :...YearsEmployed <= 0.085: 1 (13.5)
            YearsEmployed > 0.085:
            :...CreditScore > 3: 1 (22.2/2.1)
                CreditScore <= 3:
                :...Ethnicity > 1: 0 (14.4/3.4)
                    Ethnicity <= 1:
                    :...Citizen > 0: 1 (29.4/6.8)
                        Citizen <= 0:
                        :...Debt <= 0.54: 1 (10.4)
                            Debt > 0.54:
                            :...Debt <= 1.29: 0 (25.6/5.7)
                                Debt > 1.29:
                                :...Income > 5: 1 (19.5/0.6)
                                    Income <= 5:
                                    :...Income > 0: 0 (7.6)
                                        Income <= 0:
                                        :...Ethnicity > 0: 1 (27.4/5.4)
                                            Ethnicity <= 0:
                                            :...EducLevel <= 0: 0 (7.5)
                                                EducLevel > 0:
                                                :...CreditScore > 1: 1 (6.4)
                                                    CreditScore <= 1:
                                                    :...DLicense <= 0: 0 (17.7/3.4)
                                                        DLicense > 0: 1 (35.8/12.6)
```

```
Evaluation on training data (450 cases):

Trial       Decision Tree
-----     ----------------
          Size      Errors

   0        13   41( 9.1%)
   1         9   89(19.8%)
   2        15   57(12.7%)
   3        17   59(13.1%)
   4        17   79(17.6%)
   5        14   65(14.4%)
   6        19   85(18.9%)
   7        11   84(18.7%)
   8        10   49(10.9%)
   9        16   48(10.7%)
boost            21( 4.7%)   <<


           (a)   (b)    <-classified as
          ----  ----
           218     8    (a): class 0
            13   211    (b): class 1


        Attribute usage:

        100.00% PriorDefault
        100.00% CreditScore
         88.22% Debt
         82.00% Income
         80.44% Age
         78.89% YearsEmployed
         76.44% Sex
         72.22% EducLevel
         67.56% Ethnicity
         58.67% Employed
         38.67% Citizen
         37.33% DLicense
         34.44% Married
```

We have now obtained 95.3% on the training set. We can now plot the decision tree.

plot(model_c1)



We will now predict based on the decision tree generated by C5

p <- predict(model_c1, te_data_x, type="class")

sum(p == te_data_y) / length(p)

> 0.8889

We found the accuracy came from the C5 to be ~89% which is like our logistic model and model created by rpart.

After working with different models, we can confirm that for the given data we can obtain 89% accuracy which is good.