MidtermProject

Caleb Qi

3/26/2022

Installing packages and loading the data

Taking a look at the variables. It seems like MobileNo_Avl_Flag is useless because there are only 1's in it. We will split employment type up with one hot encoding because it contains 3 different variables. Also, there are a few date variables, and the CNS description variable could be removed. We have some string data, which need to be converted into integers. For example the dates. We can also spot the binary variables vs id variables from the summary.

```
attach(train_data)
summary(train_data)
```

```
##     UniqueID      disbursed_amount   asset_cost           ltv
##  Min.   :417428   Min.   : 13320   Min.   :  37000   Min.   :13.50
##  1st Qu.:476558   1st Qu.: 47145   1st Qu.:  65720   1st Qu.:68.92
##  Median :535949   Median : 53803   Median :  70960   Median :76.85
##  Mean   :535783   Mean   : 54365   Mean   :  75854   Mean   :74.77
##  3rd Qu.:594874   3rd Qu.: 60350   3rd Qu.:  79183   3rd Qu.:83.69
##  Max.   :671084   Max.   :990572   Max.   :1628992   Max.   :95.00
##    branch_id       supplier_id    manufacturer_id  Current_pincode_ID
##  Min.   :  1.00   Min.   :10524   Min.   : 45.00   Min.   :   1
##  1st Qu.: 14.00   1st Qu.:16535   1st Qu.: 48.00   1st Qu.:1511
##  Median : 61.00   Median :20328   Median : 86.00   Median :2967
##  Mean   : 72.95   Mean   :19634   Mean   : 69.07   Mean   :3393
##  3rd Qu.:130.00   3rd Qu.:23000   3rd Qu.: 86.00   3rd Qu.:5667
##  Max.   :261.00   Max.   :24803   Max.   :156.00   Max.   :7345
##  Date.of.Birth      Employment.Type    DisbursalDate         State_ID
##  Length:174865      Length:174865      Length:174865      Min.   : 1.000
##  Class :character   Class :character   Class :character   1st Qu.: 4.000
##  Mode  :character   Mode  :character   Mode  :character   Median : 6.000
##                                                           Mean   : 7.266
##                                                           3rd Qu.:10.000
##                                                           Max.   :22.000
##  Employee_code_ID MobileNo_Avl_Flag  Aadhar_flag        PAN_flag
##  Min.   :   1     Min.   :1         Min.   :0.0000   Min.   :0.00000
##  1st Qu.: 713     1st Qu.:1         1st Qu.:1.0000   1st Qu.:0.00000
##  Median :1451     Median :1         Median :1.0000   Median :0.00000
##  Mean   :1550     Mean   :1         Mean   :0.8403   Mean   :0.07591
##  3rd Qu.:2365     3rd Qu.:1         3rd Qu.:1.0000   3rd Qu.:0.00000
##  Max.   :3795     Max.   :1         Max.   :1.0000   Max.   :1.00000
##   VoterID_flag     Driving_flag     Passport_flag      PERFORM_CNS.SCORE
##  Min.   :0.0000   Min.   :0.00000   Min.   :0.000000   Min.   :  0.0
##  1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.:0.000000   1st Qu.:  0.0
##  Median :0.0000   Median :0.00000   Median :0.000000   Median :  0.0
```

```
## Mean   :0.1449   Mean   :0.02337   Mean   :0.002139   Mean   :289.1
## 3rd Qu.:0.0000   3rd Qu.:0.00000   3rd Qu.:0.000000   3rd Qu.:678.0
## Max.   :1.0000   Max.   :1.00000   Max.   :1.000000   Max.   :890.0
## PERFORM_CNS.SCORE.DESCRIPTION PRI.NO.OF.ACCTS   PRI.ACTIVE.ACCTS
## Length:174865                 Min.   :  0.000   Min.   :  0.000
## Class :character              1st Qu.:  0.000   1st Qu.:  0.000
## Mode  :character              Median :  0.000   Median :  0.000
##                               Mean   :  2.436   Mean   :  1.036
##                               3rd Qu.:  3.000   3rd Qu.:  1.000
##                               Max.   :453.000   Max.   :144.000
## PRI.OVERDUE.ACCTS PRI.CURRENT.BALANCE PRI.SANCTIONED.AMOUNT
## Min.   : 0.000    Min.   :-6678296    Min.   :0.000e+00
## 1st Qu.: 0.000    1st Qu.:       0    1st Qu.:0.000e+00
## Median : 0.000    Median :       0    Median :0.000e+00
## Mean   : 0.156    Mean   :  164886    Mean   :2.184e+05
## 3rd Qu.: 0.000    3rd Qu.:   34904    3rd Qu.:6.200e+04
## Max.   :25.000    Max.   :96524920    Max.   :1.000e+09
## PRI.DISBURSED.AMOUNT SEC.NO.OF.ACCTS    SEC.ACTIVE.ACCTS   SEC.OVERDUE.ACCTS
## Min.   :0.000e+00    Min.   : 0.00000   Min.   : 0.00000   Min.   :0.000000
## 1st Qu.:0.000e+00    1st Qu.: 0.00000   1st Qu.: 0.00000   1st Qu.:0.000000
## Median :0.000e+00    Median : 0.00000   Median : 0.00000   Median :0.000000
## Mean   :2.179e+05    Mean   : 0.05925   Mean   : 0.02743   Mean   :0.007211
## 3rd Qu.:6.021e+04    3rd Qu.: 0.00000   3rd Qu.: 0.00000   3rd Qu.:0.000000
## Max.   :1.000e+09    Max.   :52.00000   Max.   :36.00000   Max.   :8.000000
## SEC.CURRENT.BALANCE SEC.SANCTIONED.AMOUNT SEC.DISBURSED.AMOUNT
## Min.   : -239782    Min.   :       0      Min.   :       0
## 1st Qu.:       0    1st Qu.:       0      1st Qu.:       0
## Median :       0    Median :       0      Median :       0
## Mean   :    5311    Mean   :    7242      Mean   :    7131
## 3rd Qu.:       0    3rd Qu.:       0      3rd Qu.:       0
## Max.   :29560540    Max.   :30000000      Max.   :30000000
## PRIMARY.INSTAL.AMT SEC.INSTAL.AMT    NEW.ACCTS.IN.LAST.SIX.MONTHS
## Min.   :       0   Min.   :      0   Min.   : 0.0000
## 1st Qu.:       0   1st Qu.:      0   1st Qu.: 0.0000
## Median :       0   Median :      0   Median : 0.0000
## Mean   :   13477   Mean   :    307   Mean   : 0.3796
## 3rd Qu.:    1989   3rd Qu.:      0   3rd Qu.: 0.0000
## Max.   :25642806   Max.   :4170901   Max.   :35.0000
## DELINQUENT.ACCTS.IN.LAST.SIX.MONTHS AVERAGE.ACCT.AGE   CREDIT.HISTORY.LENGTH
## Min.   : 0.00000                    Length:174865      Length:174865
## 1st Qu.: 0.00000                    Class :character   Class :character
## Median : 0.00000                    Mode  :character   Mode  :character
## Mean   : 0.09672
## 3rd Qu.: 0.00000
## Max.   :20.00000
## NO.OF_INQUIRIES   loan_default
## Min.   : 0.000   Min.   :0.000
## 1st Qu.: 0.000   1st Qu.:0.000
## Median : 0.000   Median :0.000
## Mean   : 0.206   Mean   :0.218
## 3rd Qu.: 0.000   3rd Qu.:0.000
## Max.   :28.000   Max.   :1.000
```

```r
table(loan_default)
```

```
## loan_default
##      0      1
## 136748  38117
```

```r
table(Employment.Type)
```

```
## Employment.Type
##                    Salaried Self employed
##          5679         73441         95745
```

```r
table(MobileNo_Avl_Flag)
```

```
## MobileNo_Avl_Flag
##      1
## 174865
```

```r
detach(train_data)
```

In order to convert the date variables into numeric, I took a look at the structure of the variables. For credit history length and average account age, I ran a for loop to break down the string, and extract the numbers, and convert them into months.

```
##  [1] "UniqueID"                            "disbursed_amount"
##  [3] "asset_cost"                          "ltv"
##  [5] "branch_id"                           "supplier_id"
##  [7] "manufacturer_id"                     "Current_pincode_ID"
##  [9] "Date.of.Birth"                       "Employment.Type"
## [11] "DisbursalDate"                       "State_ID"
## [13] "Employee_code_ID"                    "MobileNo_Avl_Flag"
## [15] "Aadhar_flag"                         "PAN_flag"
## [17] "VoterID_flag"                        "Driving_flag"
## [19] "Passport_flag"                       "PERFORM_CNS.SCORE"
## [21] "PERFORM_CNS.SCORE.DESCRIPTION"       "PRI.NO.OF.ACCTS"
## [23] "PRI.ACTIVE.ACCTS"                    "PRI.OVERDUE.ACCTS"
## [25] "PRI.CURRENT.BALANCE"                 "PRI.SANCTIONED.AMOUNT"
## [27] "PRI.DISBURSED.AMOUNT"                "SEC.NO.OF.ACCTS"
## [29] "SEC.ACTIVE.ACCTS"                    "SEC.OVERDUE.ACCTS"
## [31] "SEC.CURRENT.BALANCE"                 "SEC.SANCTIONED.AMOUNT"
## [33] "SEC.DISBURSED.AMOUNT"                "PRIMARY.INSTAL.AMT"
## [35] "SEC.INSTAL.AMT"                      "NEW.ACCTS.IN.LAST.SIX.MONTHS"
## [37] "DELINQUENT.ACCTS.IN.LAST.SIX.MONTHS" "AVERAGE.ACCT.AGE"
## [39] "CREDIT.HISTORY.LENGTH"               "NO.OF_INQUIRIES"
## [41] "loan_default"                        "Employment.TypeSalaried"
## [43] "Employment.TypeSelf.employed"
```

Now I remove variables that are unlikely to be useful, judging them based on intuition.

I removed all the IDs except state_ID and UniqueID because It makes sense for some states to have higher loan default rates than others, because of variables like income, number of cars, gas prices, road infrastructure, etc.

I also removed Employment Type after the one hot encoding performed above.

I also removed Employee ID Code because I don't think it will be useful in predicting loan default, unless it gives us information on the type of employment, and thus income.

Lastly, I removed CNS Score Description because it is a description of the cns score and does not contain any useful information

```
trash = c(5, 6, 7, 8, 10, 13, 14, 21)
train = train_data[-c(trash)]
test = test_data[-c(trash)]
names(train)
```

```
##  [1] "UniqueID"                            "disbursed_amount"
##  [3] "asset_cost"                          "ltv"
##  [5] "Date.of.Birth"                       "DisbursalDate"
##  [7] "State_ID"                            "Aadhar_flag"
##  [9] "PAN_flag"                            "VoterID_flag"
## [11] "Driving_flag"                        "Passport_flag"
## [13] "PERFORM_CNS.SCORE"                   "PRI.NO.OF.ACCTS"
## [15] "PRI.ACTIVE.ACCTS"                    "PRI.OVERDUE.ACCTS"
## [17] "PRI.CURRENT.BALANCE"                 "PRI.SANCTIONED.AMOUNT"
## [19] "PRI.DISBURSED.AMOUNT"                "SEC.NO.OF.ACCTS"
## [21] "SEC.ACTIVE.ACCTS"                    "SEC.OVERDUE.ACCTS"
## [23] "SEC.CURRENT.BALANCE"                 "SEC.SANCTIONED.AMOUNT"
## [25] "SEC.DISBURSED.AMOUNT"                "PRIMARY.INSTAL.AMT"
## [27] "SEC.INSTAL.AMT"                      "NEW.ACCTS.IN.LAST.SIX.MONTHS"
## [29] "DELINQUENT.ACCTS.IN.LAST.SIX.MONTHS" "AVERAGE.ACCT.AGE"
## [31] "CREDIT.HISTORY.LENGTH"               "NO.OF_INQUIRIES"
## [33] "loan_default"                        "Employment.TypeSalaried"
## [35] "Employment.TypeSelf.employed"
```

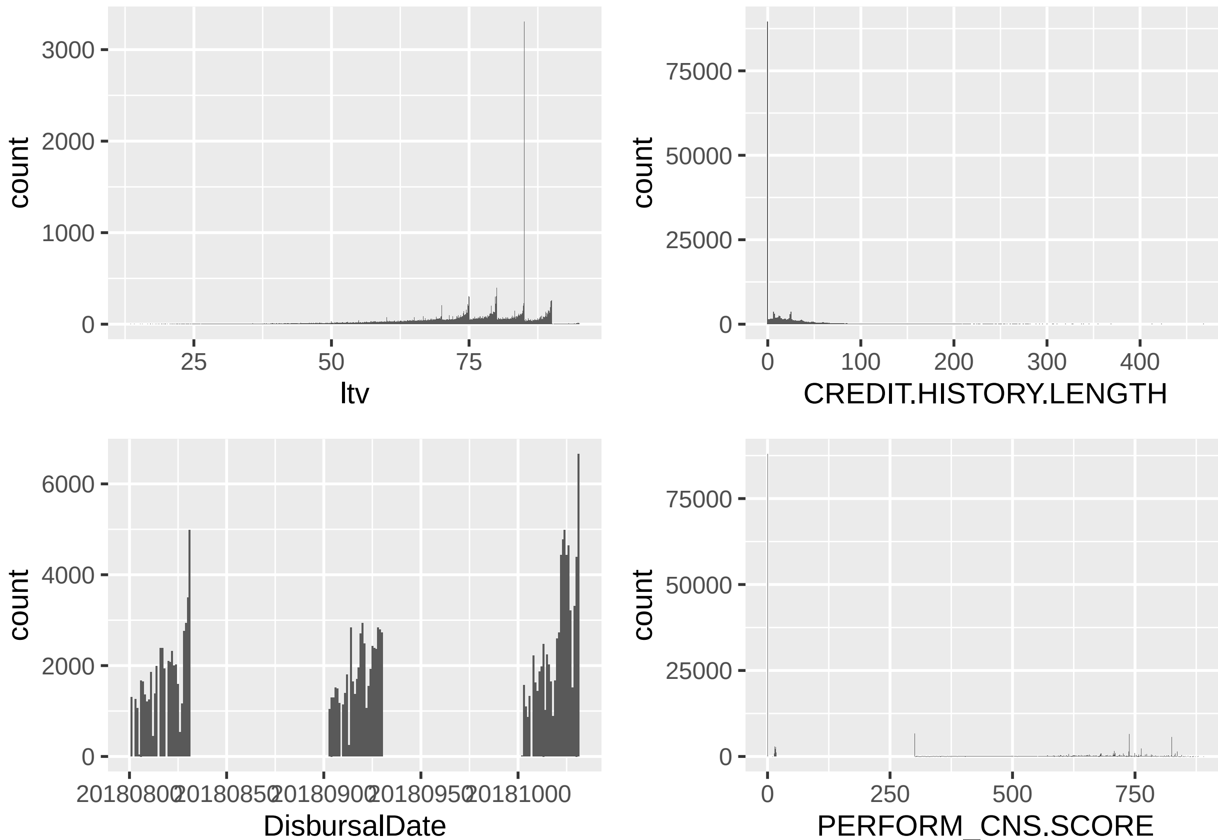
I explored the data a little bit, and here are some interesting plots I have found that give some intuition to the structure of the dataset. The LTV plot seems to indicate that there are 'sections' of loan to value, with a peak at the end of each section. Also, the credit history length plot seems to be roughly normally distributed, with a large amount of 0's.

In fact, for most variables, the most common amount seems to be 0. Intuitively this feels like it will be an issue, because if most of the variables consist of 0s, and most of the default types also consist of 0s, it will mean our model will not be able to learn much from most of the data. There needs to be some resampling done. I think instead of removing all the 0s I could do some bootsrapping.

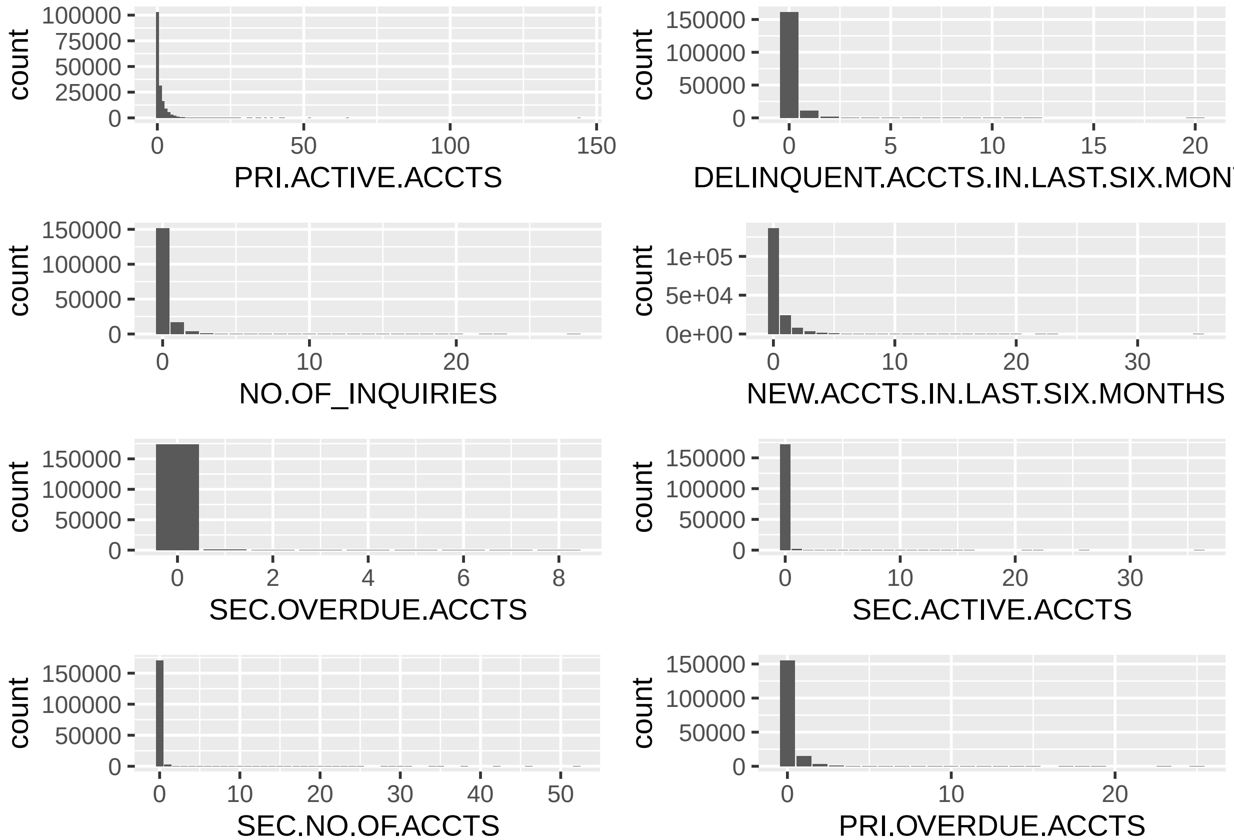
```
# patterns where the data seems split into sections
plot1 = ggplot(train, aes(x = ltv)) + geom_bar()
plot2 = ggplot(train, aes(x = CREDIT.HISTORY.LENGTH)) + geom_bar()
plot3 = ggplot(train, aes(x = DisbursalDate)) + geom_bar()
plot4 = ggplot(train, aes(x = PERFORM_CNS.SCORE)) + geom_bar()

# patterns where there is a decreasing trend of wider binds
plot5 = ggplot(train, aes(x = PRI.ACTIVE.ACCTS)) + geom_bar()
plot6 = ggplot(train, aes(x = DELINQUENT.ACCTS.IN.LAST.SIX.MONTHS)) + geom_bar()
plot7 = ggplot(train, aes(x = NO.OF_INQUIRIES)) + geom_bar()
plot8 = ggplot(train, aes(x = NEW.ACCTS.IN.LAST.SIX.MONTHS)) + geom_bar()
plot9 = ggplot(train, aes(x = SEC.OVERDUE.ACCTS)) + geom_bar()
plot10 = ggplot(train, aes(x = SEC.ACTIVE.ACCTS)) + geom_bar()
plot11 = ggplot(train, aes(x = SEC.NO.OF.ACCTS)) + geom_bar()
plot12 = ggplot(train, aes(x = PRI.OVERDUE.ACCTS)) + geom_bar()

ggarrange(plot1, plot2, plot3, plot4 ,ncol = 2, nrow = 2)
```




```
ggarrange(plot5, plot6, plot7, plot8, plot9, plot10, plot11, plot12 ,ncol = 2, nrow = 4)
```



Next I took a look at the correlation of our cleaned variables. THere were a few with very high correlation

Since PRI sanctioned amount and PRI disbursed amount are correlated, we remove Sanctioned amount because disbursal amount is more impactful because it is actually paid to the borrower as opposed to just a description of the allowed borrowed amount

PRI NO of accounts, and PRI active accounts are correlated, but we remove active accounts because total accounts are intuitively more indicative than currently active number of accounts in predicting loan defaults.

Since SEC Current balance, SEC sanctioned amount, and SEC disbursed amounts are very highly correlated, we remove sanctioned amount and current balance because it makes sense that the amount disbursed is most impactful, similar to what we did for the PRI variables.

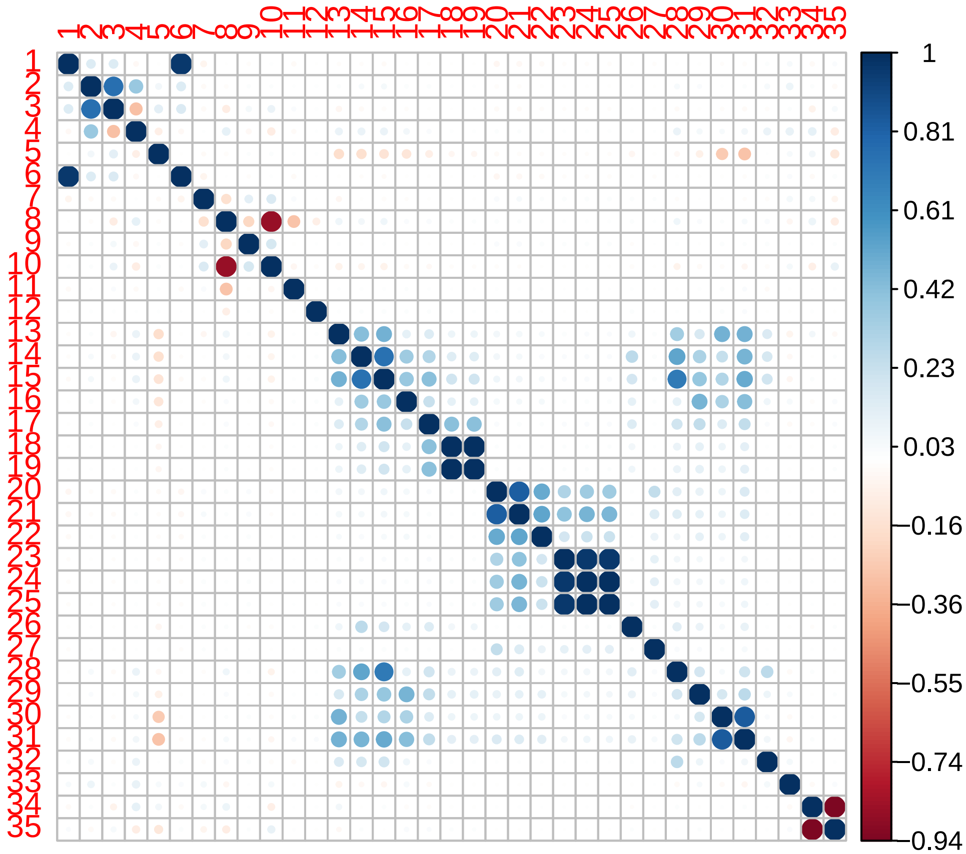
Since credit history length and average account age are highly correlated, we also drop one of credit history length and average account age because they are correlated we choose to drop average account age because credit history length seems more important

```
apply(train, 2, function(x) any(is.na(x)))
```

```
##                            UniqueID                     disbursed_amount
##                               FALSE                                FALSE
##                          asset_cost                                  ltv
##                               FALSE                                FALSE
##                       Date.of.Birth                        DisbursalDate
##                               FALSE                                FALSE
##                            State_ID                          Aadhar_flag
##                               FALSE                                FALSE
##                            PAN_flag                         VoterID_flag
##                               FALSE                                FALSE
##                        Driving_flag                        Passport_flag
```

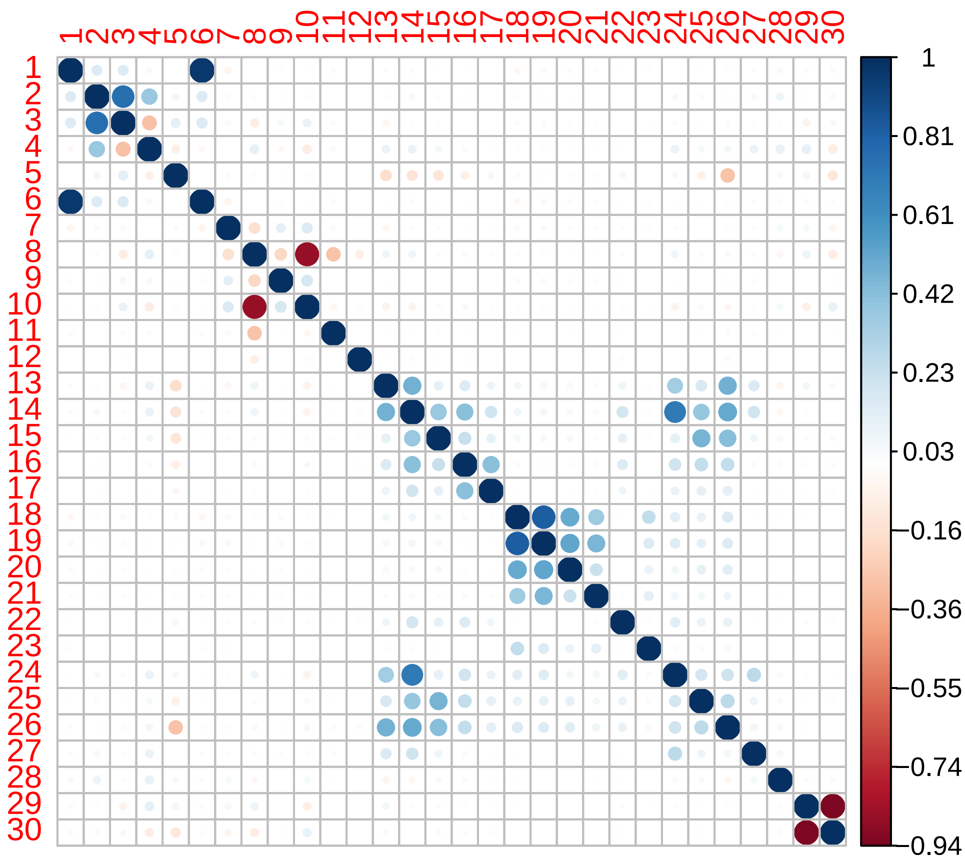
```
##                                FALSE                               FALSE
##                    PERFORM_CNS.SCORE                    PRI.NO.OF.ACCTS
##                                FALSE                               FALSE
##                     PRI.ACTIVE.ACCTS                   PRI.OVERDUE.ACCTS
##                                FALSE                               FALSE
##                  PRI.CURRENT.BALANCE               PRI.SANCTIONED.AMOUNT
##                                FALSE                               FALSE
##                 PRI.DISBURSED.AMOUNT                     SEC.NO.OF.ACCTS
##                                FALSE                               FALSE
##                     SEC.ACTIVE.ACCTS                   SEC.OVERDUE.ACCTS
##                                FALSE                               FALSE
##                  SEC.CURRENT.BALANCE               SEC.SANCTIONED.AMOUNT
##                                FALSE                               FALSE
##                 SEC.DISBURSED.AMOUNT                  PRIMARY.INSTAL.AMT
##                                FALSE                               FALSE
##                       SEC.INSTAL.AMT        NEW.ACCTS.IN.LAST.SIX.MONTHS
##                                FALSE                               FALSE
## DELINQUENT.ACCTS.IN.LAST.SIX.MONTHS                     AVERAGE.ACCT.AGE
##                                FALSE                               FALSE
##                CREDIT.HISTORY.LENGTH                     NO.OF_INQUIRIES
##                                FALSE                               FALSE
##                         loan_default            Employment.TypeSalaried
##                                FALSE                               FALSE
##        Employment.TypeSelf.employed
##                                FALSE
```

```
cormat = cor(train)
colnames(cormat) = c(1:dim(cormat)[1])
rownames(cormat) = c(1:dim(cormat)[2])
corrplot(cormat, is.corr = FALSE, method = 'circle')
```




```
# PRI no accts, PRI sanctioned amount, SEC current balance, SEC sanctioned amount, AVG Acct Age
trash = c(14, 18, 23, 24, 30)
train = train[-trash]
test = test[-trash]

cormat = cor(train)
colnames(cormat) = c(1:dim(cormat)[1])
rownames(cormat) = c(1:dim(cormat)[2])
corrplot(cormat, is.corr = FALSE, method = 'circle')
```




```
names(train)
```

```
##  [1] "UniqueID"                            "disbursed_amount"
##  [3] "asset_cost"                          "ltv"
##  [5] "Date.of.Birth"                       "DisbursalDate"
##  [7] "State_ID"                            "Aadhar_flag"
##  [9] "PAN_flag"                            "VoterID_flag"
## [11] "Driving_flag"                        "Passport_flag"
## [13] "PERFORM_CNS.SCORE"                   "PRI.ACTIVE.ACCTS"
## [15] "PRI.OVERDUE.ACCTS"                   "PRI.CURRENT.BALANCE"
## [17] "PRI.DISBURSED.AMOUNT"                "SEC.NO.OF.ACCTS"
## [19] "SEC.ACTIVE.ACCTS"                    "SEC.OVERDUE.ACCTS"
## [21] "SEC.DISBURSED.AMOUNT"                "PRIMARY.INSTAL.AMT"
## [23] "SEC.INSTAL.AMT"                      "NEW.ACCTS.IN.LAST.SIX.MONTHS"
## [25] "DELINQUENT.ACCTS.IN.LAST.SIX.MONTHS" "CREDIT.HISTORY.LENGTH"
## [27] "NO.OF_INQUIRIES"                     "loan_default"
## [29] "Employment.TypeSalaried"             "Employment.TypeSelf.employed"
```

I then performed resampling, because as previously mentioned, the default to non default ratio is too small, and most of the data for the variables are 0. In order to resample, I used the bootstrapping method, which randomly samples from the training minority, keeping the distribution of the data but magnifying the information in the data.

```
train_majority = train[train$loan_default == 0,]
train_minority = train[train$loan_default == 1,]

# Here are the 0s and 1s for train data BEFORE resampling
dim(train_majority)
```

```
## [1] 136748     30
```

```
dim(train_minority)
```

```
## [1] 38117    30
```

```
train_minority_upsampled = train_minority[sample(nrow(train_minority), dim(train_majority)[1], replace =
final_train = rbind(train_majority, train_minority_upsampled)

# Here are the 0s and 1s for train data AFTER resampling
dim(train_majority)
```

```
## [1] 136748     30
```

```
dim(train_minority_upsampled)
```

```
## [1] 136748     30
```

```
# The final training set now has following dimensions
dim(final_train)
```

```
## [1] 273496     30
```

Splitting the training data into training set and a validation set. I used 80% validation and 20% validation.

```
valcut = 0.8
set = c(sample(nrow(final_train), dim(train_majority)[1]*valcut, replace = FALSE))
train_set = final_train[set, ]
val_set = final_train[-set, ]
```

First I fit a logistic regression model with the remaining variables. I got an f1 score of 0.6. An interesting observation is that Voter ID was not an important predictor. I guess what this means is that voting tendencies of individuals are unlikely to help predict loan defaults. This might mean Republicans vs Democrats are not good predictors of vehicle loan defaults. It also seems like number of accounts (active, overdue, or new accounts in last six months) are not very useful in prediction. Lastly, employment type salaried is not very important, while self employed is imporant. This could be interpreted as the variable 'salaried or not' is less important than the variable 'self employed' or not. Maybe the people who call themselves 'self employed' are the ones who have a thin bank account.

Then I tried to refit a new logistic regression after removing the variables that were not significant. The results were roughly the same.

```
logistic.model = glm(formula = train_set$loan_default ~ ., family = binomial, data = train_set)
logistic.probs = predict(logistic.model, newdata = val_set, type="response")
logistic.preds = rep(0, dim(val_set)[1])
logistic.preds[logistic.probs > 0.5] = 1
confusionMatrix(as.factor(logistic.preds), as.factor(val_set$loan_default), mode = "everything", positi
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 44847 29501
##          1 37338 52412
##
##                Accuracy : 0.5927
##                  95% CI : (0.5903, 0.5951)
##     No Information Rate : 0.5008
##     P-Value [Acc > NIR] : < 2.2e-16
##
```

```
##                   Kappa : 0.1855
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.6398
##             Specificity : 0.5457
##          Pos Pred Value : 0.5840
##          Neg Pred Value : 0.6032
##               Precision : 0.5840
##                  Recall : 0.6398
##                      F1 : 0.6106
##              Prevalence : 0.4992
##          Detection Rate : 0.3194
##    Detection Prevalence : 0.5469
##       Balanced Accuracy : 0.5928
##
##        'Positive' Class : 1
##
```

```
summary(logistic.model)
```

```
##
## Call:
## glm(formula = train_set$loan_default ~ ., family = binomial,
##     data = train_set)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.2543  -1.1546   0.5724   1.1205   2.8510
##
## Coefficients:
##                                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)                          4.864e+04  5.504e+03   8.837  < 2e-16 ***
## UniqueID                             4.149e-06  3.349e-07  12.389  < 2e-16 ***
## disbursed_amount                    -1.423e-05  2.743e-06  -5.187 2.13e-07 ***
## asset_cost                           1.480e-05  1.837e-06   8.058 7.76e-16 ***
## ltv                                  3.989e-02  2.168e-03  18.397  < 2e-16 ***
## Date.of.Birth                        8.231e-07  6.841e-08  12.032  < 2e-16 ***
## DisbursalDate                       -2.411e-03  2.727e-04  -8.841  < 2e-16 ***
## State_ID                             2.592e-02  1.412e-03  18.363  < 2e-16 ***
## Aadhar_flag                         -2.322e-01  4.942e-02  -4.698 2.63e-06 ***
## PAN_flag                            -8.259e-02  2.434e-02  -3.393 0.000692 ***
## VoterID_flag                         3.090e-02  4.860e-02   0.636 0.524970
## Driving_flag                        -2.216e-01  5.829e-02  -3.801 0.000144 ***
## Passport_flag                       -6.606e-01  1.468e-01  -4.500 6.78e-06 ***
## PERFORM_CNS.SCORE                   -2.375e-04  2.339e-05 -10.154  < 2e-16 ***
## PRI.ACTIVE.ACCTS                    -7.932e-02  7.070e-03 -11.220  < 2e-16 ***
## PRI.OVERDUE.ACCTS                    2.944e-01  1.546e-02  19.037  < 2e-16 ***
## PRI.CURRENT.BALANCE                 -9.654e-08  1.170e-08  -8.250  < 2e-16 ***
## PRI.DISBURSED.AMOUNT                 2.915e-09  2.256e-09   1.292 0.196225
## SEC.NO.OF.ACCTS                     -2.341e-02  1.999e-02  -1.171 0.241612
## SEC.ACTIVE.ACCTS                     3.648e-02  4.342e-02   0.840 0.400813
## SEC.OVERDUE.ACCTS                    5.406e-02  7.039e-02   0.768 0.442468
## SEC.DISBURSED.AMOUNT                -2.202e-07  6.615e-08  -3.329 0.000870 ***
## PRIMARY.INSTAL.AMT                   8.984e-08  4.211e-08   2.133 0.032889 *
```

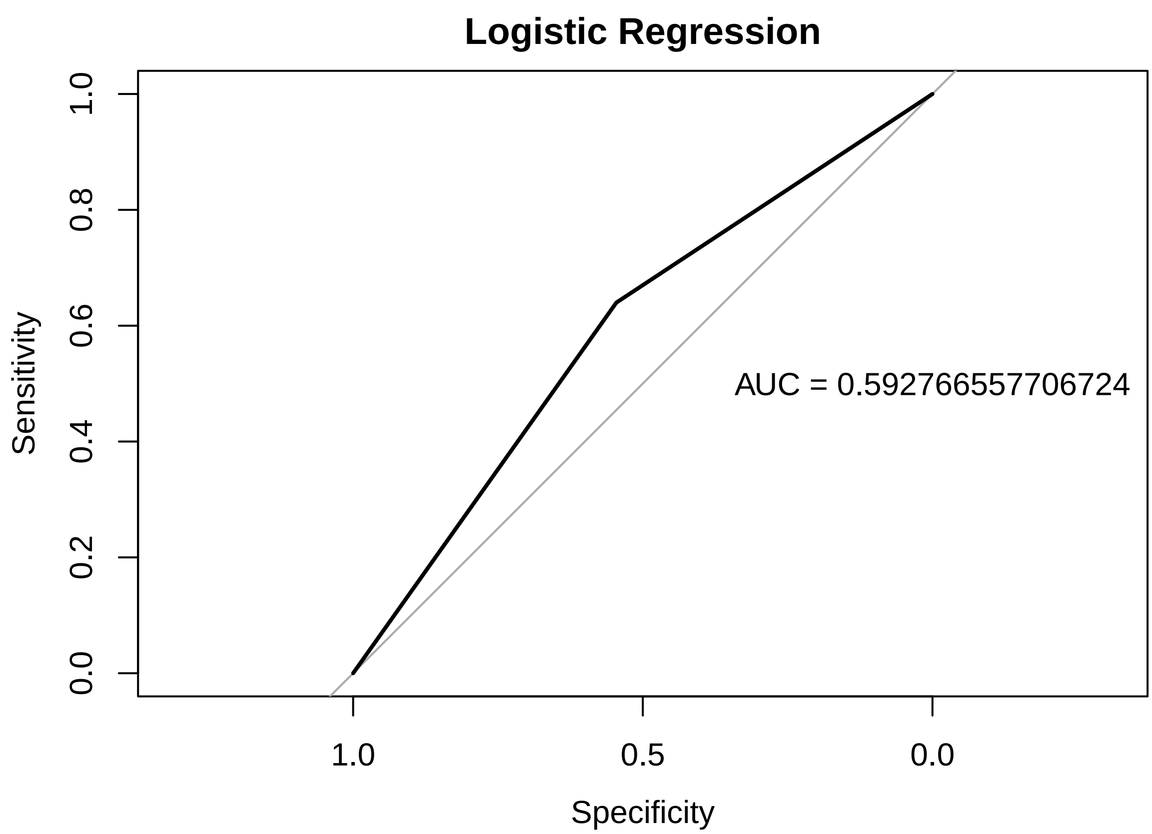
```
## SEC.INSTAL.AMT                        7.886e-07  6.030e-07   1.308 0.190934
## NEW.ACCTS.IN.LAST.SIX.MONTHS         -1.800e-02  1.092e-02  -1.648 0.099278 .
## DELINQUENT.ACCTS.IN.LAST.SIX.MONTHS  2.864e-01  1.974e-02  14.510  < 2e-16 ***
## CREDIT.HISTORY.LENGTH                -2.997e-03  3.270e-04  -9.166  < 2e-16 ***
## NO.OF_INQUIRIES                       1.738e-01  9.542e-03  18.211  < 2e-16 ***
## Employment.TypeSalaried              -4.312e-03  3.650e-02  -0.118 0.905952
## Employment.TypeSelf.employed          1.735e-01  3.644e-02   4.761 1.93e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 151657  on 109397  degrees of freedom
## Residual deviance: 146092  on 109368  degrees of freedom
## AIC: 146152
##
## Number of Fisher Scoring iterations: 4
```

```r
logroc = roc(val_set$loan_default, logistic.preds)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```r
plot(logroc, main = 'Logistic Regression' )
text(0, 0.5, paste("AUC = ", auc(logroc)[1], sep = "") )
```




```r
# Finding the variables that were useless
trash_var = c(10, 17, 18, 19, 20, 24, 29)
train_set2 = train_set[-trash_var]
```

```r
final_test = test[-trash_var]
names(train_set2)
```

```
##  [1] "UniqueID"                            "disbursed_amount"
##  [3] "asset_cost"                          "ltv"
##  [5] "Date.of.Birth"                       "DisbursalDate"
##  [7] "State_ID"                            "Aadhar_flag"
##  [9] "PAN_flag"                            "Driving_flag"
## [11] "Passport_flag"                       "PERFORM_CNS.SCORE"
## [13] "PRI.ACTIVE.ACCTS"                    "PRI.OVERDUE.ACCTS"
## [15] "PRI.CURRENT.BALANCE"                 "SEC.DISBURSED.AMOUNT"
## [17] "PRIMARY.INSTAL.AMT"                  "SEC.INSTAL.AMT"
## [19] "DELINQUENT.ACCTS.IN.LAST.SIX.MONTHS" "CREDIT.HISTORY.LENGTH"
## [21] "NO.OF_INQUIRIES"                     "loan_default"
## [23] "Employment.TypeSelf.employed"
```

```r
# Refitting a new model after removing those variables
logistic.model2 = glm(formula = train_set2$loan_default ~ ., family = binomial, data = train_set2)
logistic.probs2 = predict(logistic.model2, newdata = val_set, type="response")
logistic.preds2 = rep(0, dim(val_set)[1])
logistic.preds2[logistic.probs2 > 0.5] = 1
confusionMatrix(as.factor(logistic.preds2), as.factor(val_set$loan_default), mode = "everything", posit
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 44829 29466
##          1 37356 52447
##
##                Accuracy : 0.5928
##                  95% CI : (0.5904, 0.5952)
##     No Information Rate : 0.5008
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.1857
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.6403
##             Specificity : 0.5455
##          Pos Pred Value : 0.5840
##          Neg Pred Value : 0.6034
##               Precision : 0.5840
##                  Recall : 0.6403
##                      F1 : 0.6109
##              Prevalence : 0.4992
##          Detection Rate : 0.3196
##    Detection Prevalence : 0.5473
##       Balanced Accuracy : 0.5929
##
##        'Positive' Class : 1
##
```

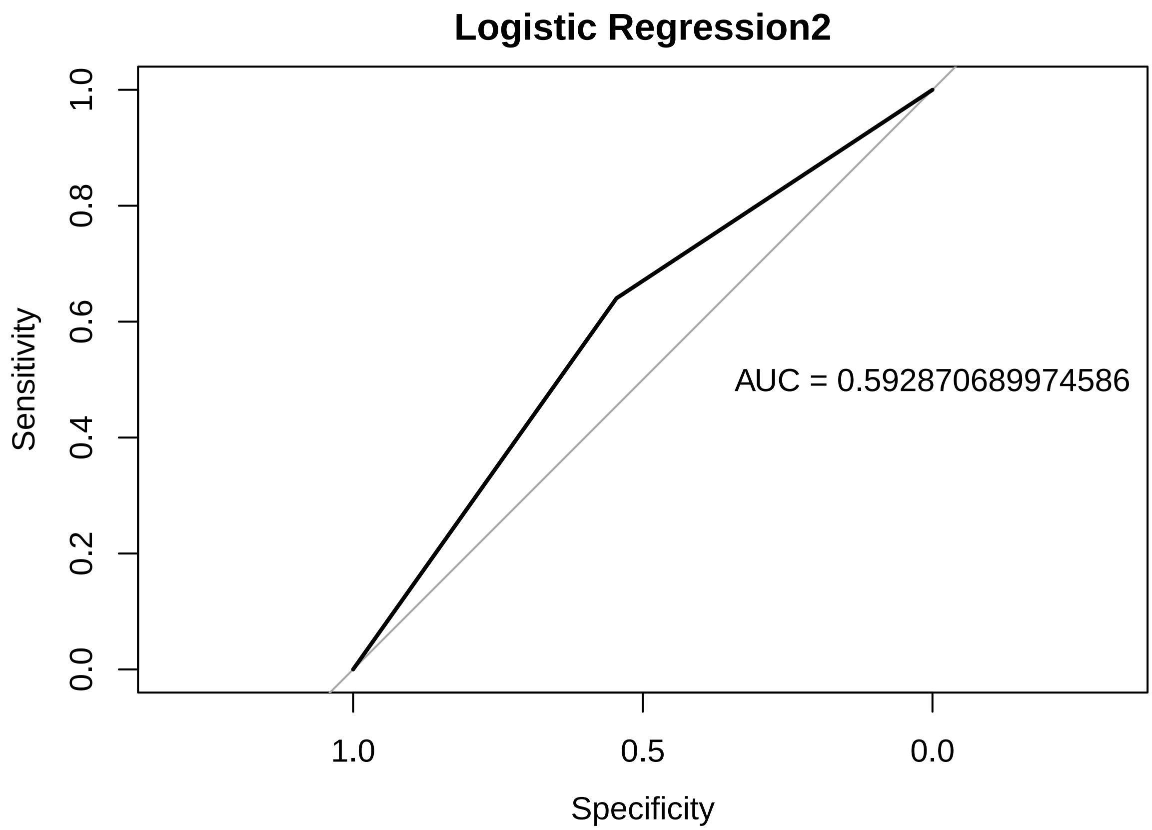
```r
summary(logistic.model2)
```

```
##
## Call:
## glm(formula = train_set2$loan_default ~ ., family = binomial,
##     data = train_set2)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.2577  -1.1548   0.5726   1.1203   2.8205
##
## Coefficients:
##                                        Estimate Std. Error z value Pr(>|z|)
## (Intercept)                           4.849e+04  5.503e+03   8.812  < 2e-16 ***
## UniqueID                              4.139e-06  3.348e-07  12.362  < 2e-16 ***
## disbursed_amount                     -1.422e-05  2.741e-06  -5.186 2.14e-07 ***
## asset_cost                            1.480e-05  1.836e-06   8.061 7.58e-16 ***
## ltv                                   3.985e-02  2.167e-03  18.391  < 2e-16 ***
## Date.of.Birth                         8.210e-07  6.742e-08  12.178  < 2e-16 ***
## DisbursalDate                        -2.404e-03  2.727e-04  -8.816  < 2e-16 ***
## State_ID                              2.590e-02  1.411e-03  18.358  < 2e-16 ***
## Aadhar_flag                          -2.620e-01  1.837e-02 -14.265  < 2e-16 ***
## PAN_flag                             -8.401e-02  2.428e-02  -3.461 0.000539 ***
## Driving_flag                         -2.469e-01  4.357e-02  -5.667 1.45e-08 ***
## Passport_flag                        -6.849e-01  1.418e-01  -4.828 1.38e-06 ***
## PERFORM_CNS.SCORE                    -2.394e-04  2.335e-05 -10.250  < 2e-16 ***
## PRI.ACTIVE.ACCTS                     -8.738e-02  5.149e-03 -16.970  < 2e-16 ***
## PRI.OVERDUE.ACCTS                     2.978e-01  1.527e-02  19.505  < 2e-16 ***
## PRI.CURRENT.BALANCE                  -9.145e-08  1.132e-08  -8.079 6.51e-16 ***
## SEC.DISBURSED.AMOUNT                 -2.111e-07  5.757e-08  -3.666 0.000246 ***
## PRIMARY.INSTAL.AMT                    8.980e-08  4.203e-08   2.137 0.032629 *
## SEC.INSTAL.AMT                        6.700e-07  5.854e-07   1.144 0.252475
## DELINQUENT.ACCTS.IN.LAST.SIX.MONTHS   2.897e-01  1.957e-02  14.807  < 2e-16 ***
## CREDIT.HISTORY.LENGTH                -2.872e-03  3.163e-04  -9.079  < 2e-16 ***
## NO.OF_INQUIRIES                       1.711e-01  9.381e-03  18.237  < 2e-16 ***
## Employment.TypeSelf.employed          1.776e-01  1.281e-02  13.862  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 151657  on 109397  degrees of freedom
## Residual deviance: 146100  on 109375  degrees of freedom
## AIC: 146146
##
## Number of Fisher Scoring iterations: 4
```

```r
logroc2 = roc(val_set$loan_default, logistic.preds2)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```r
plot(logroc2, main = 'Logistic Regression2' )
text(0, 0.5, paste("AUC = ", auc(logroc2)[1], sep = "") )
```



Here I chose to try a decision tree model. It predicted the results slightly better, with f1 of 61, which made me wonder if random forest would perform better because it combines decision trees in a more robust way. That being said, the AUC was lower, at 55. Judging from the plot, it seems like the 'wedge' point, the point where the angle of the curve changes, is shifted to a higher specificity.

I chose Decision Tree to try to get an intuitive sense of which variables were useful. This is because Decision trees classify by finding the best variable to split into pure branches, which is most similar to how humans think of classification. According to the summary, it seems like the only variable used in tree construction was ltv. I think more could be done though, and this is a good start. If I use a random forest I might be able to use more variables.

```
tree.model = tree(train_set2$loan_default~., train_set, split = c("deviance", "gini"), method = 'class')
tree.probs = predict(tree.model, val_set)
tree.preds = rep(0, dim(val_set)[1])
tree.preds[tree.probs > 0.5] = 1
confusionMatrix(as.factor(tree.preds), as.factor(val_set$loan_default), mode = "everything", positive="
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 22417 15139
##          1 59768 66774
##
##                Accuracy : 0.5435
##                  95% CI : (0.5411, 0.5459)
##     No Information Rate : 0.5008
##     P-Value [Acc > NIR] : < 2.2e-16
##
```

```
##                   Kappa : 0.0879
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.8152
##             Specificity : 0.2728
##          Pos Pred Value : 0.5277
##          Neg Pred Value : 0.5969
##               Precision : 0.5277
##                  Recall : 0.8152
##                      F1 : 0.6407
##              Prevalence : 0.4992
##          Detection Rate : 0.4069
##    Detection Prevalence : 0.7711
##       Balanced Accuracy : 0.5440
##
##        'Positive' Class : 1
##
```

```
summary(tree.model)
```

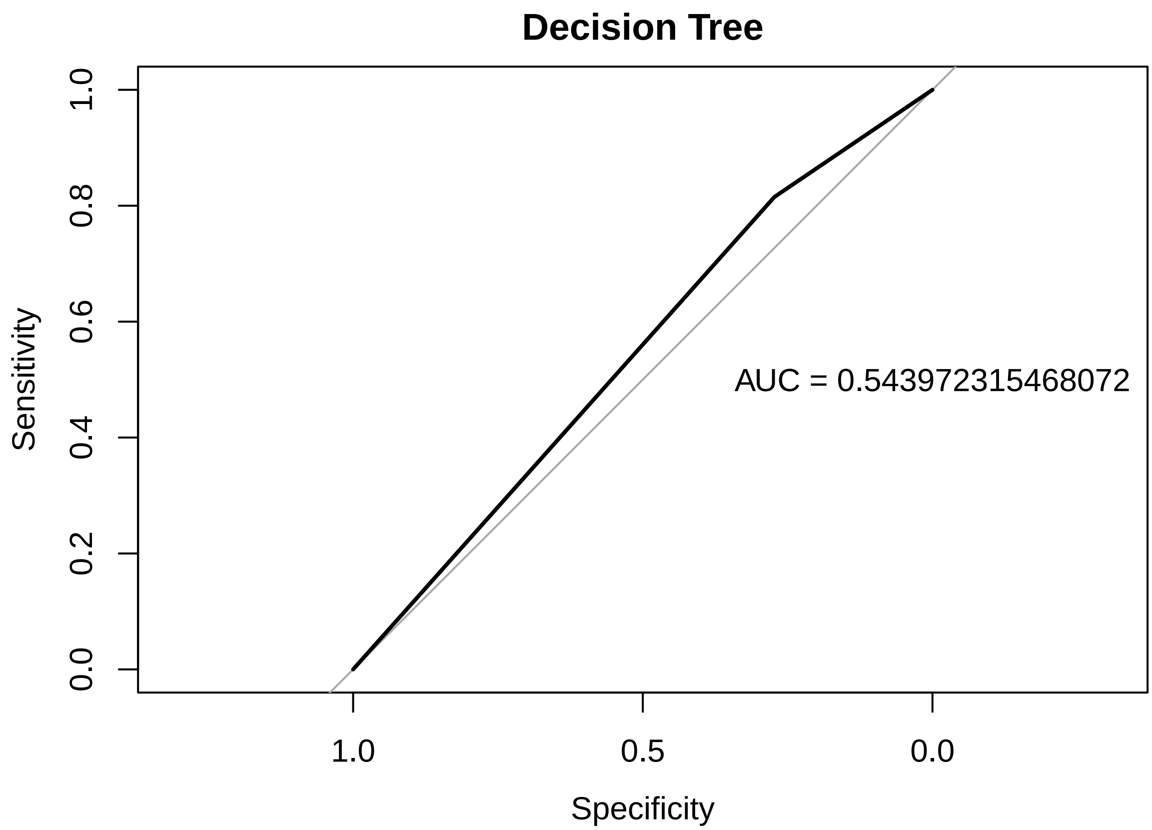
```
##
## Regression tree:
## tree(formula = train_set2$loan_default ~ ., data = train_set,
##     method = "class", split = c("deviance", "gini"))
## Variables actually used in tree construction:
## [1] "ltv"
## Number of terminal nodes:  2
## Residual mean deviance:  0.2469 = 27010 / 109400
## Distribution of residuals:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -0.5315 -0.5315  0.4685  0.0000  0.4685  0.6012
```

```
treeroc = roc(val_set$loan_default, tree.preds)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

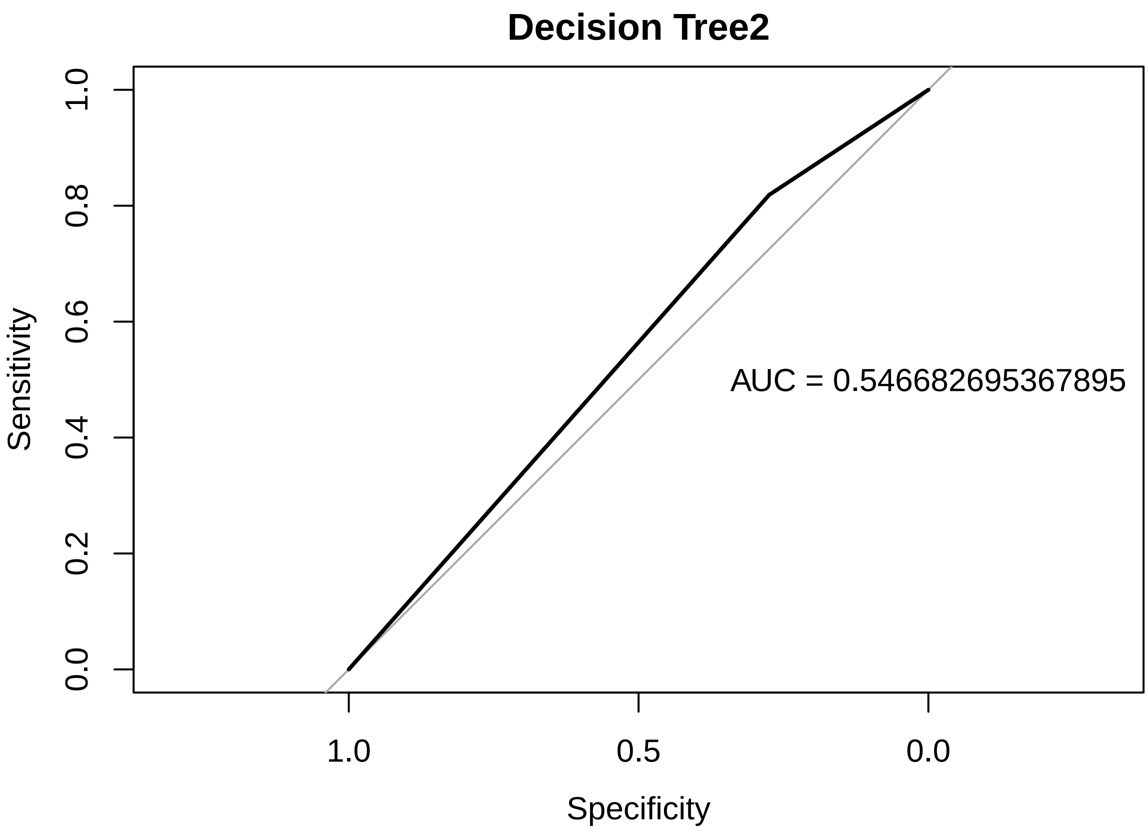
```
plot(treeroc, main = 'Decision Tree' )
text(0, 0.5, paste("AUC = ", auc(treeroc)[1], sep = "") )
```




```
tree.probs2 = predict(tree.model)
tree.preds2 = rep(0, dim(train_set)[1])
tree.preds2[tree.probs2 > 0.5] = 1
treeroc2 = roc(train_set$loan_default, tree.preds2)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
plot(treeroc2, main = 'Decision Tree2' )
text(0, 0.5, paste("AUC = ", auc(treeroc2)[1], sep = "") )
```



I also fit a random forest model, in hopes that the robustness provided by the random forest could better take advantage of resampled data. The f1 score results were varying around 81. Given how well the result for random forest is, I think that the results really depend on how much data we have, so I tried a bagging method, which reduces the variance of random forest. Especially since when I increased the number of trees from 25 to 40, the results improved a little. I could include more trees to maybe improve the f1 score higher by a decreasing amount, but It would take too long to run.

The inclusion of bagging further improved the results slightly (by less than 1). It seems that the initial ratio of default data to non default data was too small. After resampling, and bagging, and using random forest, our resulting model was able to better absorb all the information in the default data to learn the predictors better.

```
forest.model = randomForest(as.factor(train_set$loan_default) ~., data = train_set, ntree = 40)
forest.preds = predict(forest.model, val_set)
confusionMatrix(forest.preds, as.factor(val_set$loan_default), mode = "everything", positive="1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 61349 12347
##          1 20836 69566
##
##                Accuracy : 0.7978
##                  95% CI : (0.7958, 0.7997)
##     No Information Rate : 0.5008
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5956
```

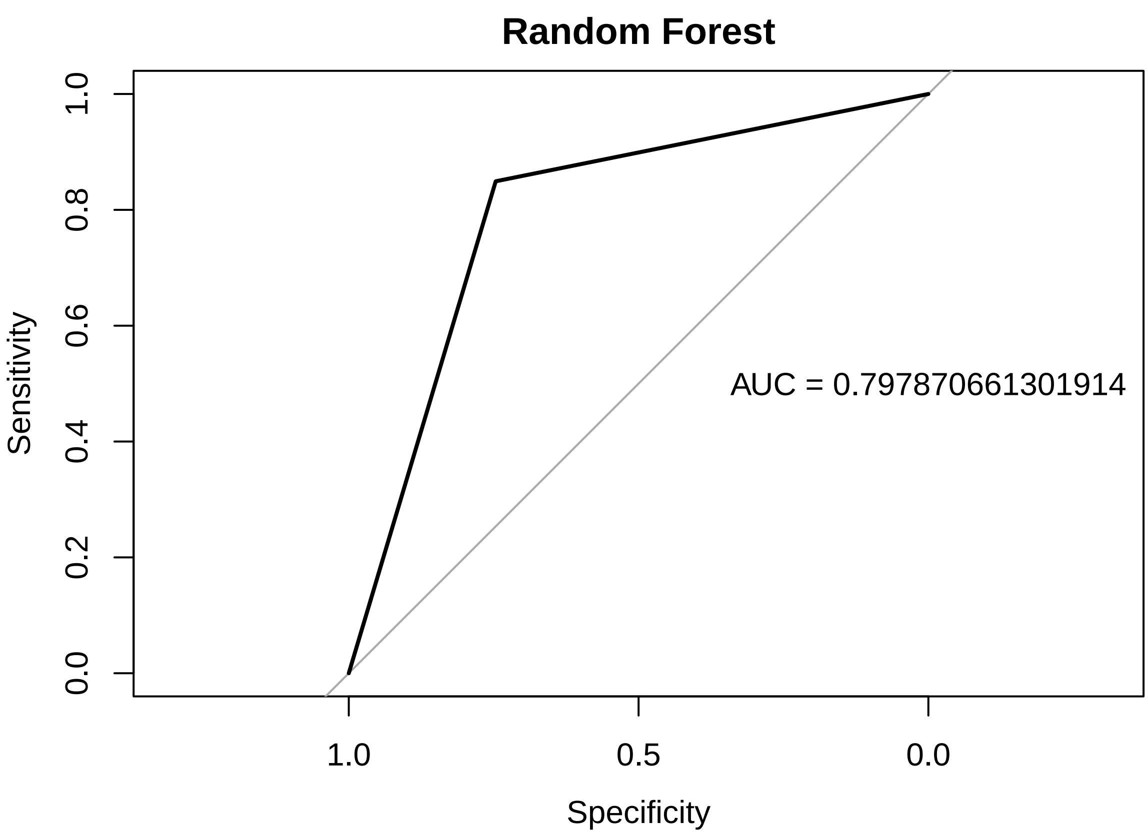
```
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.8493
##             Specificity : 0.7465
##          Pos Pred Value : 0.7695
##          Neg Pred Value : 0.8325
##               Precision : 0.7695
##                  Recall : 0.8493
##                      F1 : 0.8074
##              Prevalence : 0.4992
##          Detection Rate : 0.4239
##    Detection Prevalence : 0.5509
##       Balanced Accuracy : 0.7979
##
##        'Positive' Class : 1
##
```

```
forestroc = roc(val_set$loan_default, as.numeric(forest.preds ))
```

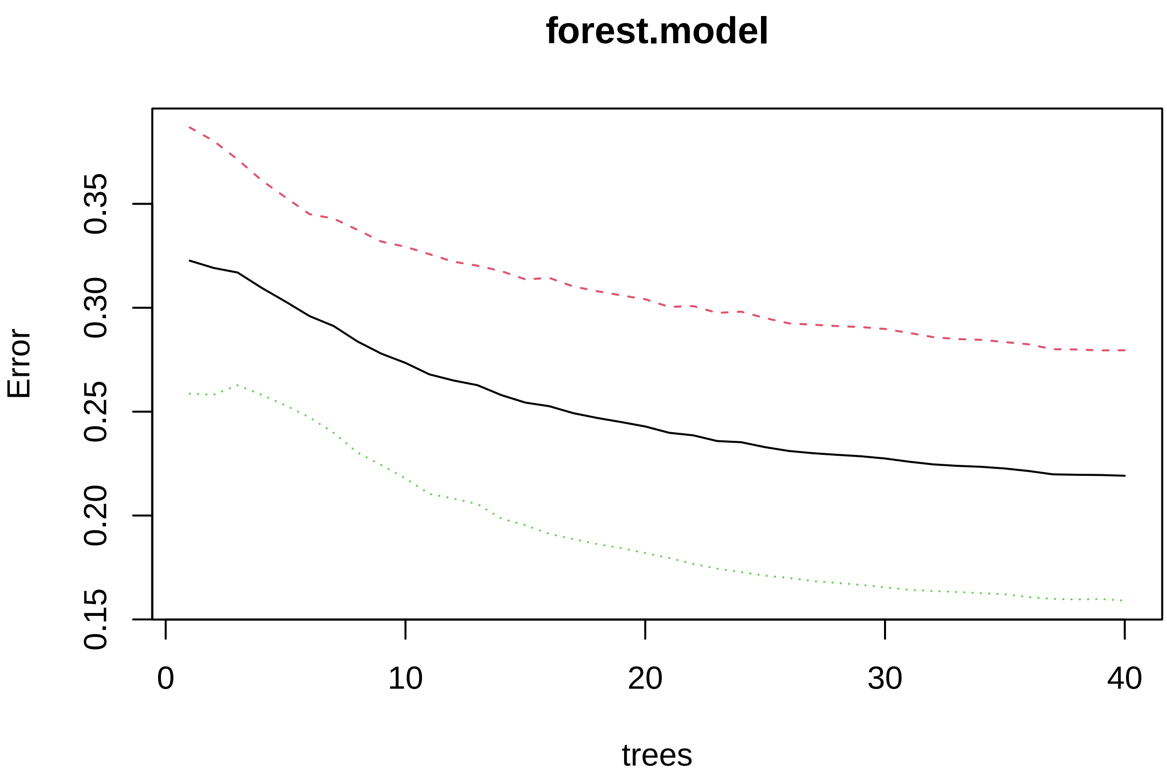
```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
plot(forestroc, main = 'Random Forest' )
text(0, 0.5, paste("AUC = ", auc(forestroc)[1], sep = "") )
```




```
plot(forest.model)
```




```
forest.model2 = randomForest(as.factor(train_set$loan_default) ~., data = train_set, mtry = 22, ntree =
forest.preds2 = predict(forest.model2, val_set)
confusionMatrix(forest.preds2, as.factor(val_set$loan_default), mode = "everything", positive="1")
```

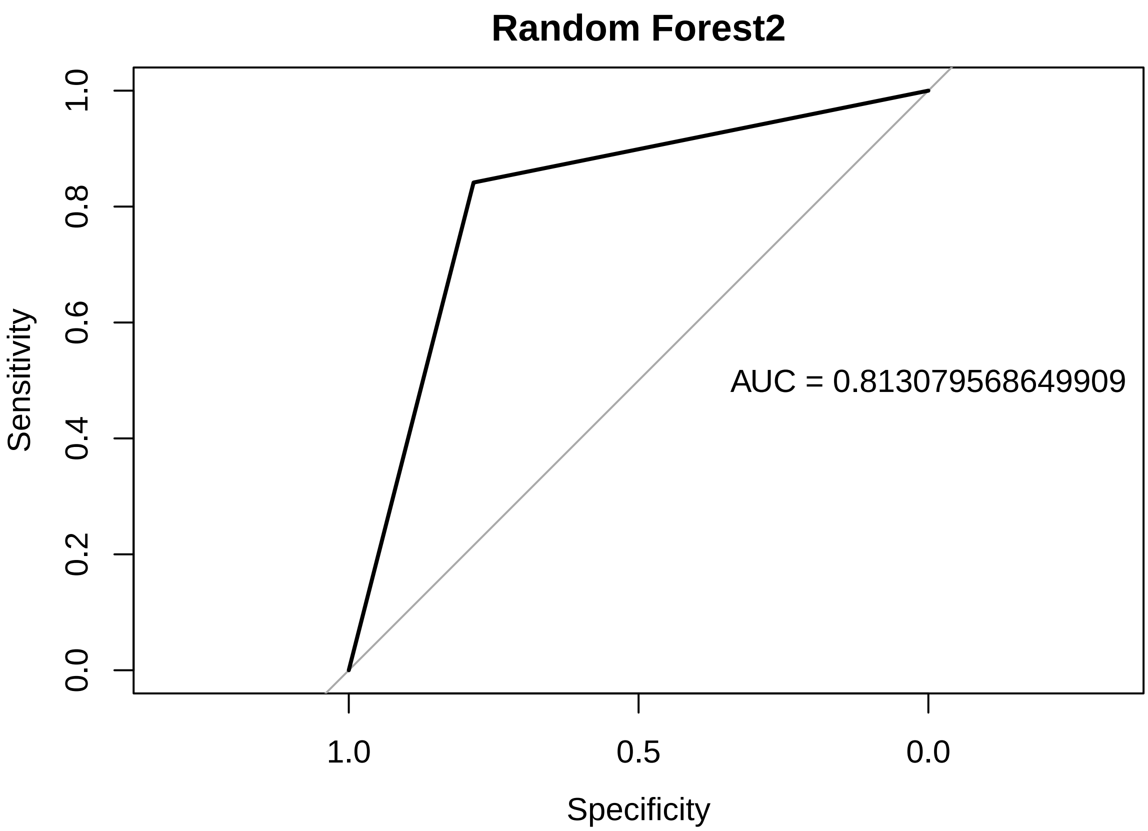
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 64487 12983
##          1 17698 68930
##
##                Accuracy : 0.813
##                  95% CI : (0.8111, 0.8149)
##     No Information Rate : 0.5008
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6261
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.8415
##             Specificity : 0.7847
##          Pos Pred Value : 0.7957
##          Neg Pred Value : 0.8324
##               Precision : 0.7957
##                  Recall : 0.8415
##                      F1 : 0.8180
##              Prevalence : 0.4992
##          Detection Rate : 0.4201
##    Detection Prevalence : 0.5279
```

```
##       Balanced Accuracy : 0.8131
##
##        'Positive' Class : 1
##
```

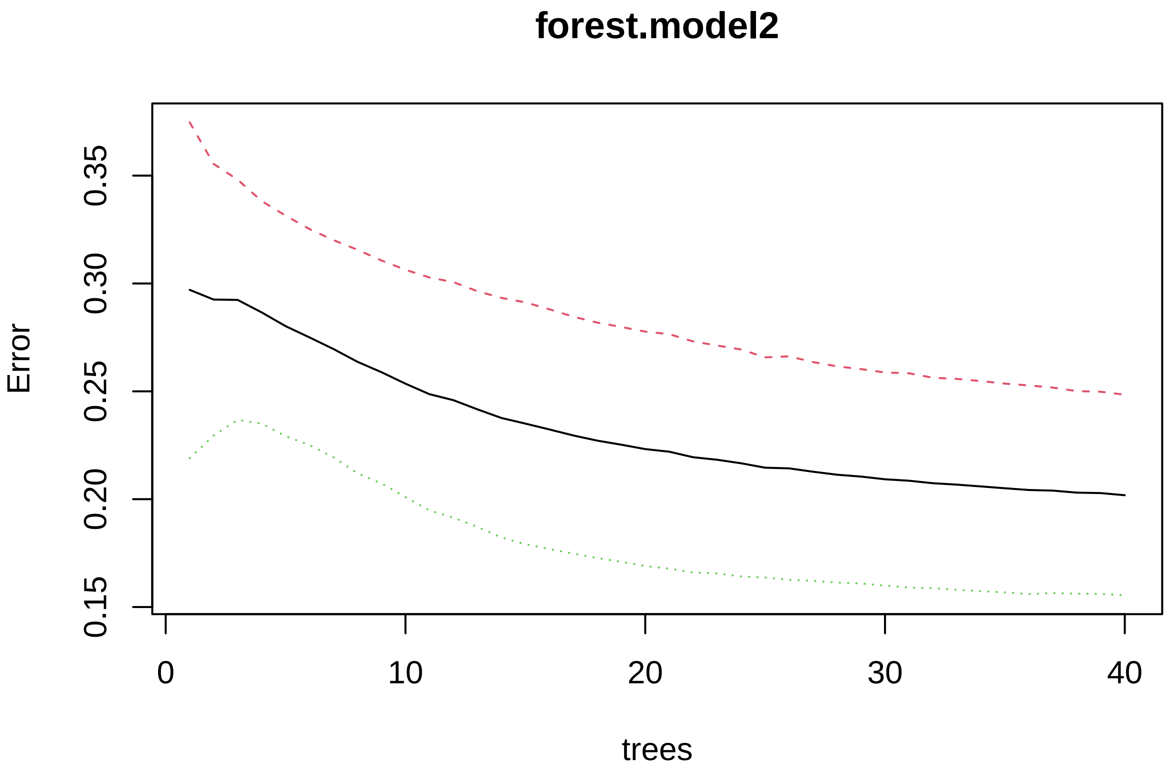
```
forestroc2 = roc(val_set$loan_default, as.numeric(forest.preds2))
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
plot(forestroc2, main = 'Random Forest2' )
text(0, 0.5, paste("AUC = ", auc(forestroc2)[1], sep = "") )
```




```
plot(forest.model2)
```



Finally, we refit a final model with the full data and predict with the test data to come up with results and save them into csv. The best model we obtained was one that uses bagging on random forest, which obtained an f1 score of 81 percent.

```
forest.model_final = randomForest(as.factor(final_train$loan_default) ~., data = final_train, mtry = 22
forest.preds_final = predict(forest.model_final, test)
write.csv(test_data,"/Users/Caleb/Documents/Classes/Boston/Spring 2022/815/Midterm Project/prediction.c
```