Loan Granting

Mitul Shah

8/9/2017

Loading the data

```
## Loading the datasets
borrower_table <- read.csv("loan/borrower_table.csv")
loan_table <- read.csv("loan/loan_table.csv")

## Merging the datasets
data = merge(borrower_table, loan_table, by = "loan_id", all.x = TRUE)
```

Checking Data Quality

The columns fully repaid previous loans and currently repaying other loans provide the same information. Both these columns cannot have the same value as we have access to a specific bank loan data. Let's check this.

```
## Number of rows with fully repaid previous loans = 1 and currently repaying other loans = 1
nrow(subset(data, data$fully_repaid_previous_loans == 1 & data$currently_repaying_other_loans == 1))
```

```
## [1] 14974
```

```
## Number of rows with fully repaid previous loans = 0 and currently repaying other loans = 0
nrow(subset(data, data$fully_repaid_previous_loans == 0 & data$currently_repaying_other_loans == 0))
```

```
## [1] 2807
```

We need to remove these rows from the data.

```
## Subsetting rows with 0s in column 3 and 4 simultaneously
newdata1 <- data[which(data$fully_repaid_previous_loans == 0 & data$currently_repaying_other_loans == 0)

## Subsetting rows with 1s in column 3 and 4 simultaneously
newdata2 <- data[which(data$fully_repaid_previous_loans == 1 & data$currently_repaying_other_loans == 1)

## Removing the above rows
newdata <- data[-c(as.numeric(rownames(newdata1)),as.numeric(rownames(newdata2))),]

## Removing the column 4 as it indicates the same information as column 3
newdata <- newdata[, -4]

## Replace missing values
newdata[3][is.na(newdata[3])] <- 2
newdata[5][is.na(newdata[5])] <- 0

## Looking at the structure
str(newdata)
```

```
## 'data.frame':    83319 obs. of  15 variables:
##  $ loan_id                                       : int  30 34 37 39 40 48 111 120 121 140 ...
##  $ is_first_loan                                 : int  0 1 0 0 1 1 0 0 1 1 ...
##  $ fully_repaid_previous_loans                   : num  1 2 1 1 2 2 0 1 2 2 ...
##  $ total_credit_card_limit                       : int  4900 7800 3800 8200 5000 3200 3800 4800 540
##  $ avg_percentage_credit_card_limit_used_last_year: num  0.6 1.04 0.59 0.69 0.84 0.5 0.78 0.29 0.26
##  $ saving_amount                                 : int  1378 1053 3212 654 2362 805 929 3175 281 53
##  $ checking_amount                               : int  1414 1243 9336 5272 1308 1757 3114 2811 497
##  $ is_employed                                   : int  0 0 1 1 1 1 1 1 1 ...
##  $ yearly_salary                                 : int  0 0 36000 35700 9000 2400 42600 29800 34300
##  $ age                                           : int  43 71 27 44 45 57 61 25 62 57 ...
##  $ dependent_number                              : int  8 8 1 0 8 8 1 0 8 4 ...
##  $ loan_purpose                                  : Factor w/ 5 levels "business","emergency_funds",
##  $ date                                          : Factor w/ 260 levels "2012-01-02","2012-01-03",.
##  $ loan_granted                                  : int  0 0 1 0 0 1 1 1 0 0 ...
##  $ loan_repaid                                   : int  NA NA 1 NA NA 0 0 1 NA NA ...
```

```
## Convert some variables to factors
newdata$is_first_loan <- as.factor(newdata$is_first_loan)
newdata$fully_repaid_previous_loans <- as.factor(newdata$fully_repaid_previous_loans)
newdata$is_employed <- as.factor(newdata$is_employed)
newdata$loan_repaid <- as.factor(newdata$loan_repaid)
newdata$loan_granted <- as.factor(newdata$loan_granted)

## Load the library lubridate
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

## The following object is masked from 'package:base':
##
##     date
```

```
## Convert date to date variable
newdata$date <- as.Date(newdata$date, format = "%Y-%m-%d")
```

```
## Warning in strptime(x, format, tz = "GMT"): unknown timezone 'zone/tz/
## 2018c.1.0/zoneinfo/America/New_York'
```

```
## Extract day of the week and
newdata$week <- week(newdata$date)
newdata$weekday <- as.factor(weekdays(newdata$date))
```

Building a model better than the bank's model

Here, we have access to the data saying whether the borrower repaid the loan or not. So, in order to build a model better than the bank's model, we can predict which loan borrowers will repay the loan. If she will repay the loan, than we should grant the loan to her, otherwise not.

But first, we need to subset the data for which the loan was granted by the bank since only for those observations, we would know whether the borrower repaid the loan or not.

```
## Load dplyr
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:lubridate':
##
##     intersect, setdiff, union

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
## Subsetting observations for which the bank granted the laan
bank_loan_granted_data <- filter(newdata, loan_granted == 1)
```

Now, we will build the model to predict who will repay the loan using Random Forest.

```
## Taking 66% data as training data
train_sample <- sample(nrow(bank_loan_granted_data), size = nrow(bank_loan_granted_data)*0.66)

## Training data
train_data <- bank_loan_granted_data[train_sample,]
train_data <- train_data[, -c(1, 13, 14)]

## Test data
test_data <- bank_loan_granted_data[-train_sample,]
test_data <- test_data[, -c(1, 13, 14)]

## Load the library randomForest
library(randomForest)
```

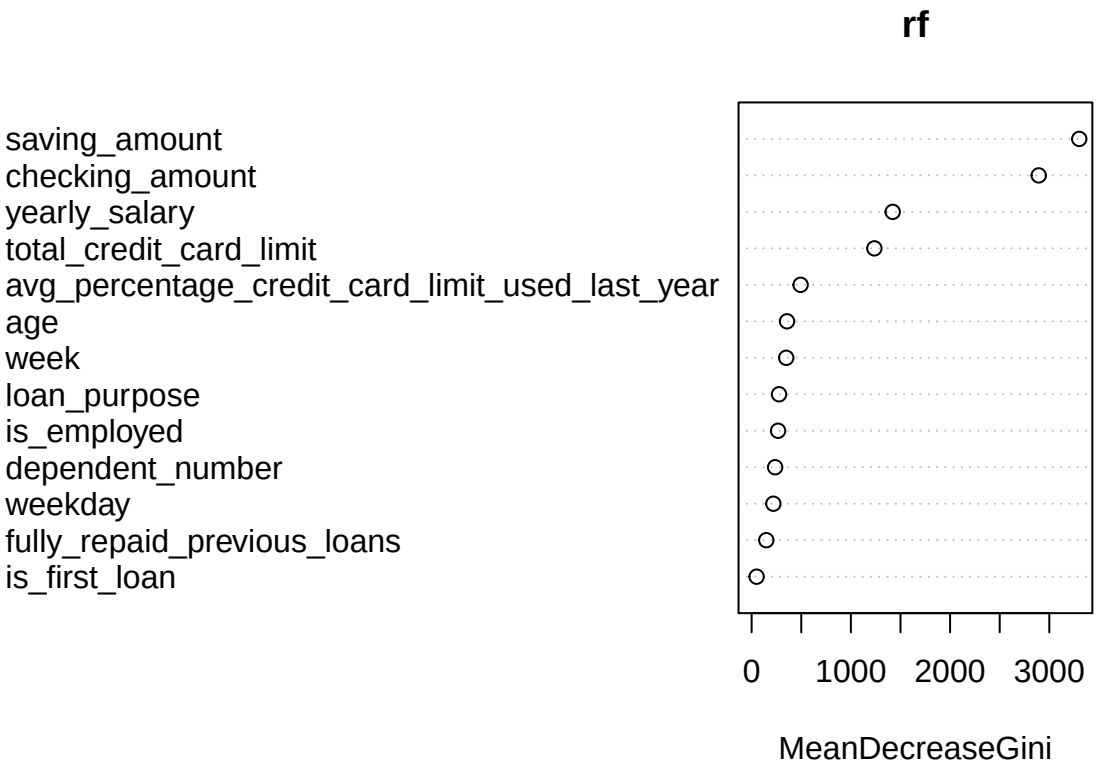
```
## randomForest 4.6-12

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:dplyr':
##
##     combine
```

```
## Random Forest Model
rf <- randomForest(y = train_data$loan_repaid, x = train_data[, -12],
ytest = test_data$loan_repaid, xtest = test_data[, -12], ntree = 100, mtry = 3, keep.forest = TRUE)
rf
```

```
##
## Call:
##  randomForest(x = train_data[, -12], y = train_data$loan_repaid,      xtest = test_data[, -12], ytes
##                Type of random forest: classification
##                      Number of trees: 100
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 8.49%
## Confusion matrix:
##      0     1 class.error
## 0 7093  1038  0.12765957
## 1 1225 17292  0.06615542
##                 Test set error rate: 8.24%
## Confusion matrix:
##      0    1 class.error
## 0 3695  551  0.12976919
## 1  580 8902  0.06116853
```

Impact of the most important variables on the prediction

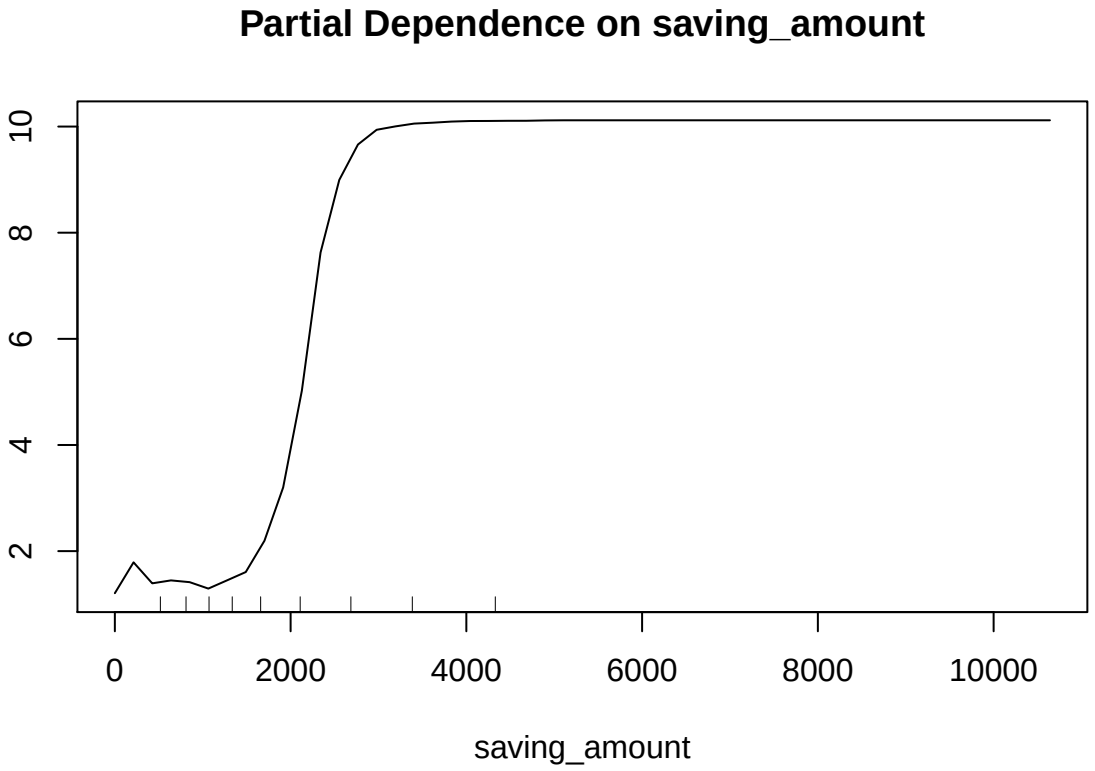
```
## Variable importance plot
varImpPlot(rf, type=2)
```



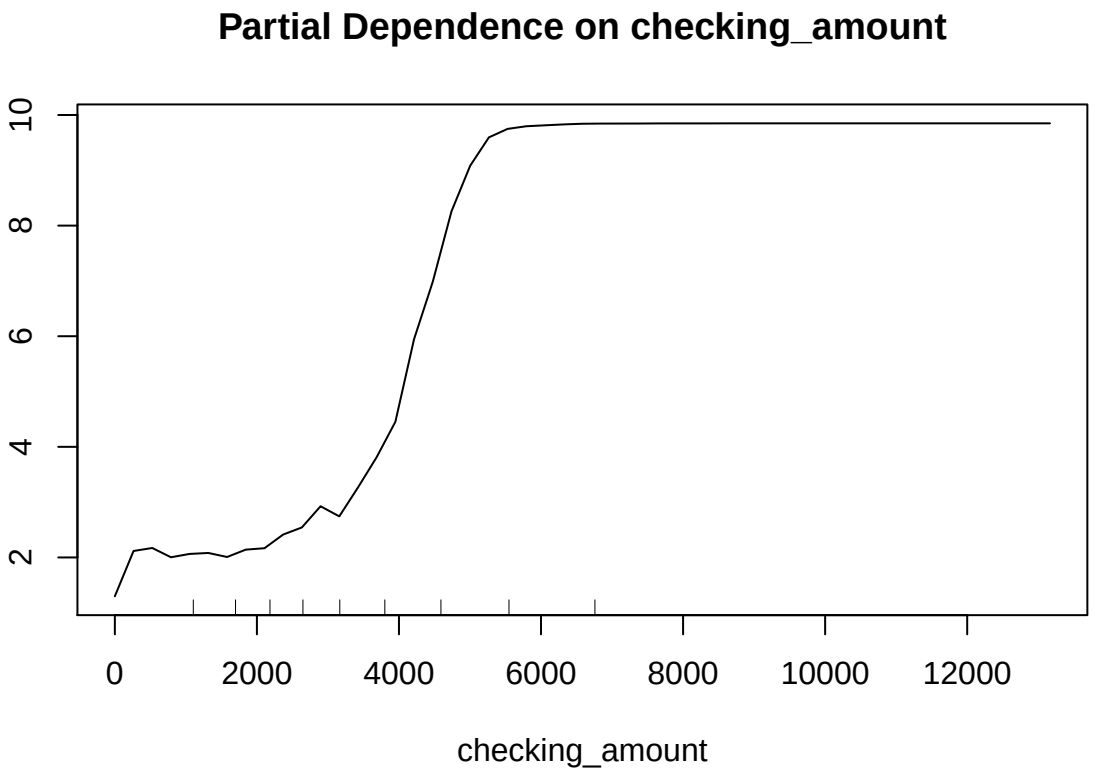
The most important variables are saving amount, checking amount, yearly salary and the total credit card limit.

Here are the partial dependence plots of these four variables:

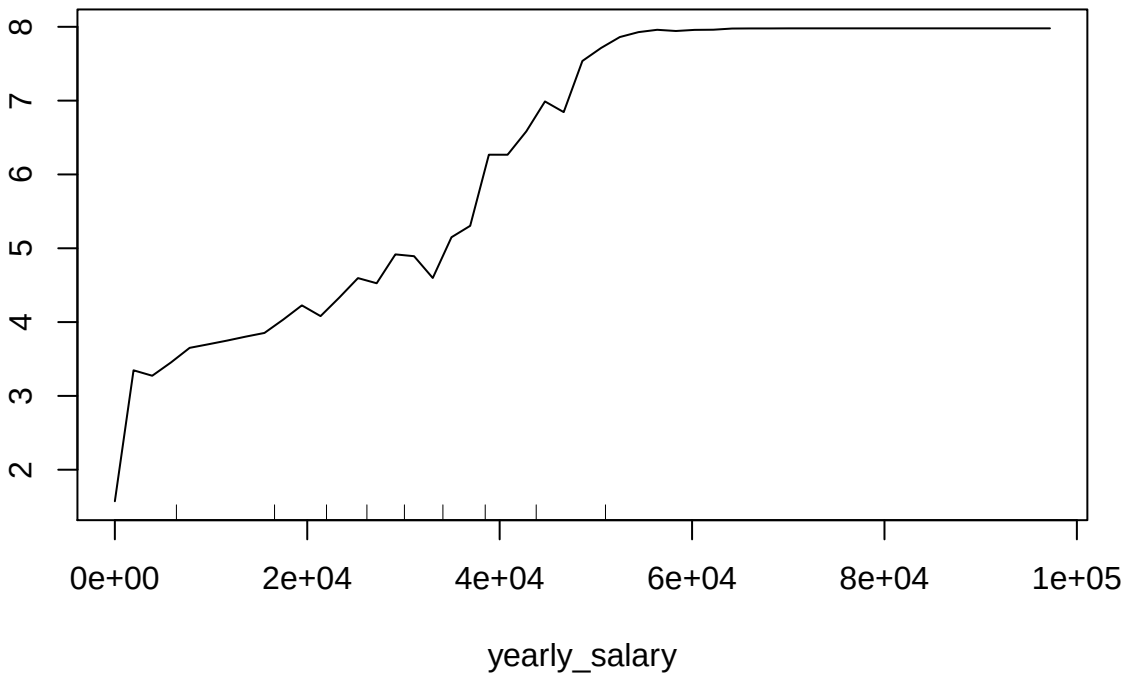
```
## Partial dependence plot of Saving Amount
partialPlot(rf, train_data, saving_amount , 1)
```



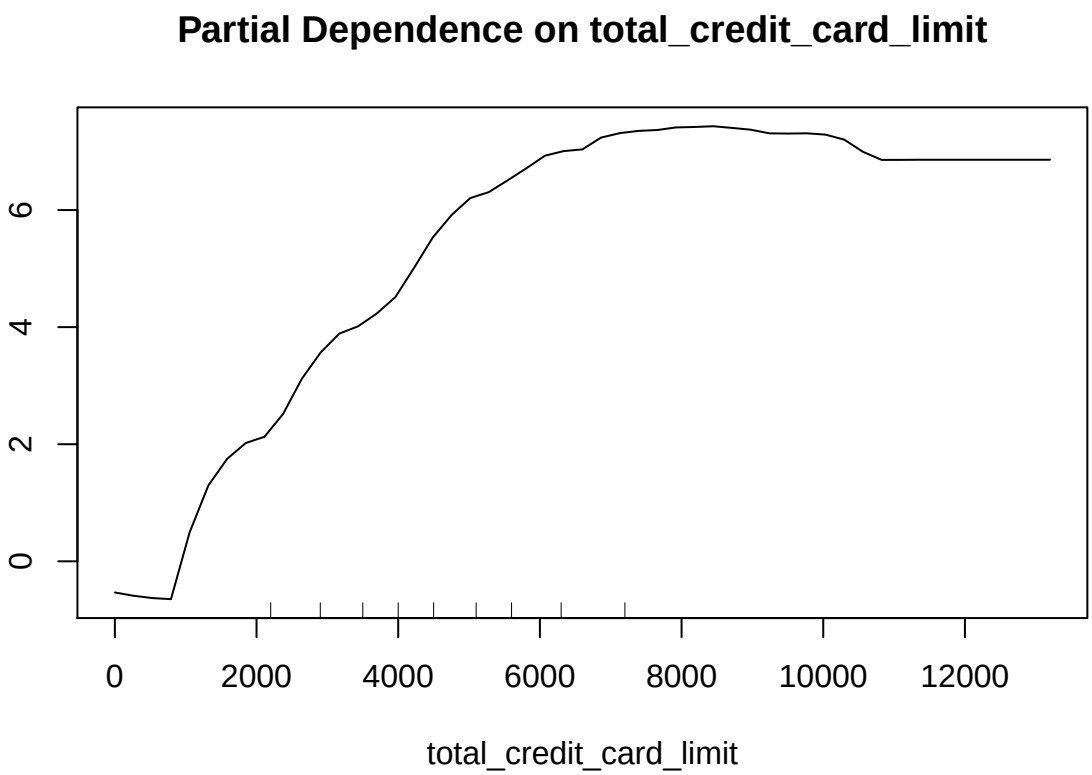

```
## Partial dependence plot of Checking Amount
partialPlot(rf, train_data, checking_amount , 1)
```




```
## Partial dependence plot of yearly salary
partialPlot(rf, train_data, yearly_salary , 1)
```



```
## Partial dependence plot of total credit card limit
partialPlot(rf, train_data, total_credit_card_limit , 1)
```



Impact of the variable "is_employed"

```
## Table for employed and the borrowers who repay the loan
table(test_data$is_employed, test_data$loan_repaid)
```

```
##
##        0    1
##   0  872  249
##   1 3374 9233
```

```
## Number of unemployed who repays the loan
250 / (250 + 883)
```

```
## [1] 0.2206531
```

```
## Number of employed who repay the loan
9307 / (9307 + 3288)
```

```
## [1] 0.738944
```

So, we see that just 22 percent of people who are unemployed repay the loan while 73 percent of the people who are employed repay the loan, given these people are granted the loan. So this variable was definitely significant. But, the variables like Saving Amount, Checking Amount, yearly salary and total credit card limit highly correlates with this variable "is_employed", due to which it appreared to be insignificant in the variable importance plot.

Compare Bank Profitability vs Our Model Profitability

We will compare the profitability only for the test set using the rules given in the challenge, i.e.

1. If you grant the loan and the it doesn't get repaid, you lose 1
2. If you grant the loan and the it does get repaid, you gain 1
3. If you don't grant the loan, you gain 0

Bank Profitability

```
## Bank Profitability using the above rules (test data has all observations where bank granted the loan)
nrow(filter(test_data, loan_repaid == 1)) - nrow(filter(test_data, loan_repaid == 0))
```

```
## [1] 5236
```

Our Model Profitability

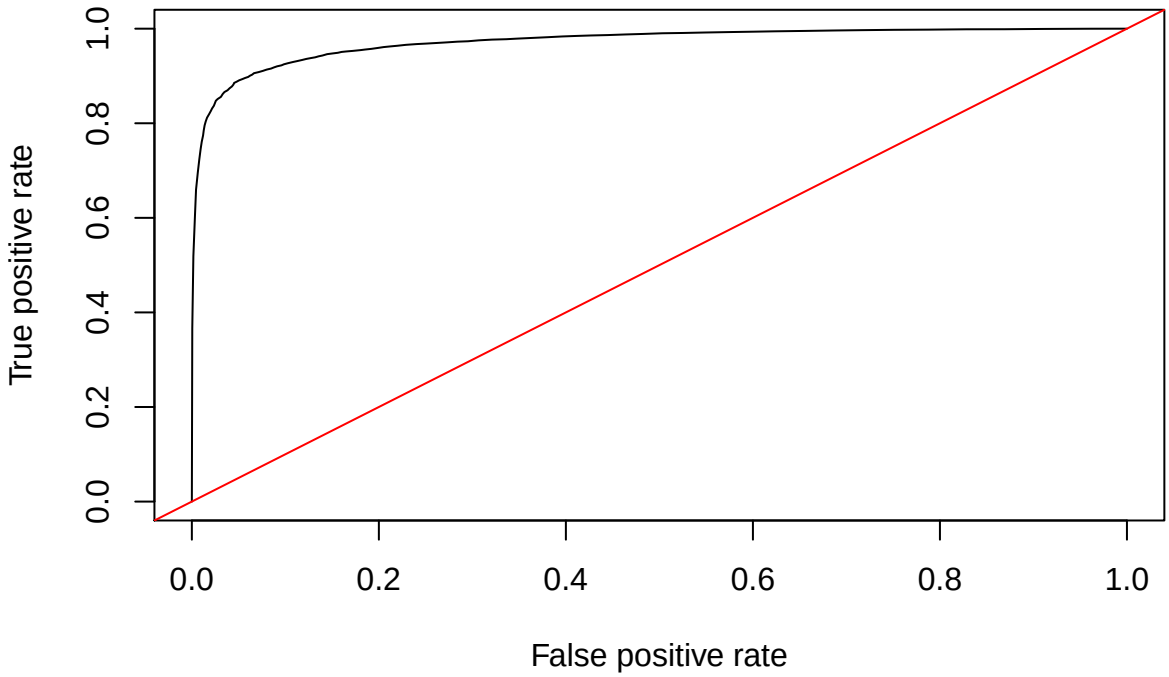
```
## Load ROCR
library(ROCR)
```

```
## Loading required package: gplots
```

```
##
## Attaching package: 'gplots'
```

```
## The following object is masked from 'package:stats':
##
##     lowess
```

```
## Random Forest Predictions
rf_results <- data.frame(actually_repaid = test_data$loan_repaid, predictions = rf$test$votes[,2])

## ROC Curve to look at tpr vs fpr
pred <- prediction(rf_results$predictions, rf_results$actually_repaid)
perf <- performance(pred, measure = "tpr", x.measure = "fpr")
plot(perf) + abline(a = 0, b = 1, col = "red")
```




```
## numeric(0)
```

Here, according to the rules given to evaluate the model, we care more about the difference between True Positives and the False Positives. False Negatives doesn't make any difference here according to the method of evaluation.

We can decrease the False Positives by increasing the threshold. But at the same time, the True Positives would also decrease. We need to increase the TPs and decrease the FPs.

I have shown the results using four different thresholds of 0.4, 0.5, 0.6 and 0.7 here.

```
## Choosing four different thresholds
rf_results$prediction_threshold_0.4 <- ifelse(rf_results$predictions > 0.4, 1, 0)
rf_results$prediction_threshold_0.5 <- ifelse(rf_results$predictions > 0.5, 1, 0)
rf_results$prediction_threshold_0.6 <- ifelse(rf_results$predictions > 0.6, 1, 0)
rf_results$prediction_threshold_0.7 <- ifelse(rf_results$predictions > 0.7, 1, 0)

## Confusion Matrix for the threshold of 0.4
table(rf_results$actually_repaid, rf_results$prediction_threshold_0.4)
```

```
##
##        0    1
##   0 3479  767
##   1  429 9053
```

```
## Our model Profitability for the threshold of 0.4
9069 - 713
```

```
## [1] 8356
```

```
## Confusion Matrix for the threshold of 0.5
table(rf_results$actually_repaid, rf_results$prediction_threshold_0.5)
```

```
##
##        0    1
##   0 3704  542
##   1  588 8894
```

```
## Our model Profitability for the threshold of 0.5
8897 - 476
```

```
## [1] 8421
```

```
## Confusion Matrix for the threshold of 0.6
table(rf_results$actually_repaid, rf_results$prediction_threshold_0.6)
```

```
##
##        0    1
##   0 3886  360
##   1  799 8683
```

```
## Our model Profitability for the threshold of 0.4
8683 - 306
```

```
## [1] 8377
```

```
## Confusion Matrix for the threshold of 0.7
table(rf_results$actually_repaid, rf_results$prediction_threshold_0.7)
```

```
##
##        0    1
##   0 4031  215
##   1 1035 8447
```

```
## Our model Profitability for the threshold of 0.4
8447 - 183
```

```
## [1] 8264
```

Probably the model with the threshold of 0.5 would be the best here as the Profitability is highest for that value.

Other Variables we would like to include in the model

1. Loan Amount
2. Loan Amount Term
3. Credit History
4. Co-applicant Income