Perform the Random Forest analysis of the credit data

Step 1: Collecting data

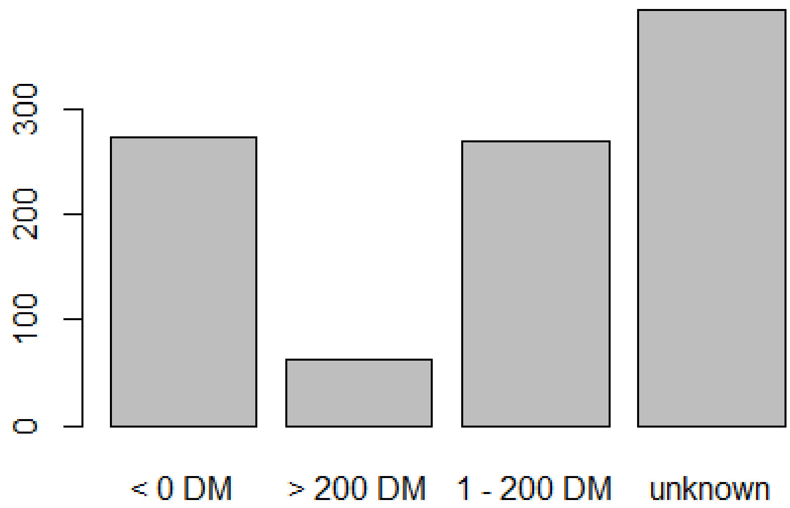
```
credit <- read.csv("http://www.sci.csueastbay.edu/~esuess/classes/Statistics_
6620/Presentations/ml7/credit.csv")

str(credit)
```

```
## 'data.frame':    1000 obs. of  17 variables:
##  $ checking_balance    : Factor w/ 4 levels "< 0 DM","> 200 DM",..: 1 3 4
1 1 4 4 3 4 3 ...
##  $ months_loan_duration: int  6 48 12 42 24 36 24 36 12 30 ...
##  $ credit_history      : Factor w/ 5 levels "critical","good",..: 1 2 1 2
4 2 2 2 2 1 ...
##  $ purpose             : Factor w/ 6 levels "business","car",..: 5 5 4 5 2
4 5 2 5 2 ...
##  $ amount              : int  1169 5951 2096 7882 4870 9055 2835 6948 3059
5234 ...
##  $ savings_balance     : Factor w/ 5 levels "< 100 DM","> 1000 DM",..: 5 1
1 1 1 5 4 1 2 1 ...
##  $ employment_duration : Factor w/ 5 levels "< 1 year","> 7 years",..: 2 3
4 4 3 3 2 3 4 5 ...
##  $ percent_of_income   : int  4 2 2 2 3 2 3 2 2 4 ...
##  $ years_at_residence  : int  4 2 3 4 4 4 4 2 4 2 ...
##  $ age                 : int  67 22 49 45 53 35 53 35 61 28 ...
##  $ other_credit        : Factor w/ 3 levels "bank","none",..: 2 2 2 2 2 2
2 2 2 2 ...
##  $ housing             : Factor w/ 3 levels "other","own",..: 2 2 2 1 1 1
2 3 2 2 ...
##  $ existing_loans_count: int  2 1 1 1 2 1 1 1 1 2 ...
##  $ job                 : Factor w/ 4 levels "management","skilled",..: 2 2
4 2 2 4 2 1 4 1 ...
##  $ dependents          : int  1 1 2 2 2 2 1 1 1 1 ...
##  $ phone               : Factor w/ 2 levels "no","yes": 2 1 1 1 1 2 1 2 1
1 ...
##  $ default             : Factor w/ 2 levels "no","yes": 1 2 1 1 2 1 1 1 1
2 ...
```

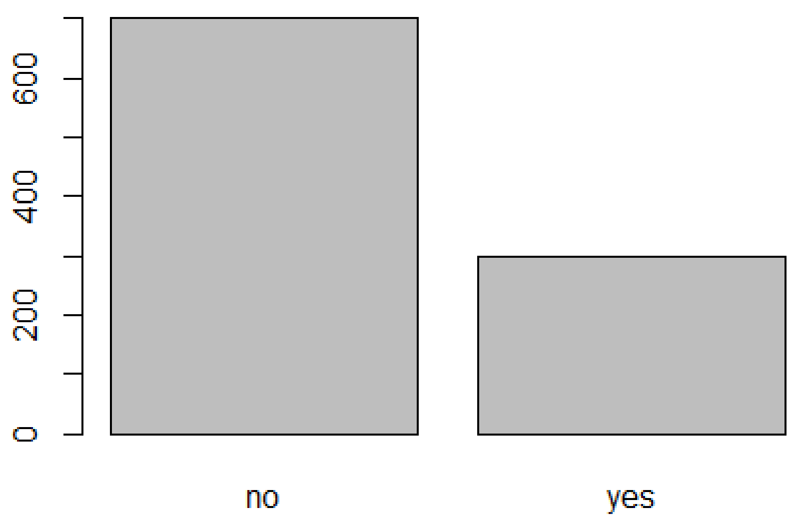
Step 2: Exploring and preparing the data

- Creating bar plots for checking balance and default

```
barplot(table(credit$checking_balance))
```




```r
barplot(table(credit$default))
```




```r
table(credit$default)
```

```
##
##  no yes
## 700 300
```

- Creating train and test data set

```
set.seed(123)
samp <- sample(nrow(credit), 0.7 * nrow(credit))
train <- credit[samp, ]
test <- credit[-samp, ]
```

Step 3: Training a model on the data

```
library(randomForest)

## randomForest 4.6-12

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

model <- randomForest(default ~ ., data = train, ntree=1000, mtry=5)
model

##
## Call:
##  randomForest(formula = default ~ ., data = train, ntree = 1000,      mtry
= 5)
##                Type of random forest: classification
##                      Number of trees: 1000
## No. of variables tried at each split: 5
##
##         OOB estimate of  error rate: 24.71%
## Confusion matrix:
##      no yes class.error
## no  443  48  0.09775967
## yes 125  84  0.59808612
```

Step 4: Evaluating model performance

- Predict test data based on the train data model

```
pred <- predict(model, newdata = test)
```

- Creating a confusion matric

```
table(pred, test$default)

##
## pred   no yes
##   no  195  57
##   yes  14  34
```

- Calculating the accuracy which is about 76% accuracte

```
sum(pred==test$default) / nrow(test)

## [1] 0.7633333
```

Step 5: Improving model performance

- auto-tune a random forest
- We don't see any improvement as we change mtry (split) from 2, 4, 8, 16 with different kappa.

grid_rf <- expand.grid(.mtry = c(2, 4, 8, 16))

set.seed(300) m_rf <- train(default ~ ., data = test, method = "rf", metric = "Kappa", trControl = ctrl, tuneGrid = grid_rf) m_rf

```
Random Forest


300 samples

 16 predictor

  2 classes: 'no', 'yes'


No pre-processing

Resampling: Cross-Validated (10 fold, repeated 10 times)

Summary of sample sizes: 270, 270, 271, 270, 270, 270, ...

Resampling results across tuning parameters:


  mtry  Accuracy   Kappa

   2    0.7247438  0.1333683

   4    0.7374027  0.2642532

   8    0.7287775  0.2761908

  16    0.7347668  0.3139176


Kappa was used to select the optimal model using  the largest value.

The final value used for the model was mtry = 16.
```