Lab 9

```r
## Intro to Data Science - Lab 9
# IST687 Section M002
# Professor Anderson
# Enter your name here: Chaithra Kopparam Cheluvaiah
# 1. I did this homework by myself, with help from the book and the professor.

#installing required packages
#install.packages("kernlab")
#install.packages("caret")
```

```r
# importing the packages
library(kernlab)
library(caret)
```

```
## Loading required package: ggplot2

##
## Attaching package: 'ggplot2'

## The following object is masked from 'package:kernlab':
##
##     alpha

## Loading required package: lattice
```

```r
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v tibble  3.1.5     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.0.2     v forcats 0.5.1
## v purrr   0.3.4

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x ggplot2::alpha() masks kernlab::alpha()
## x purrr::cross()   masks kernlab::cross()
## x dplyr::filter()  masks stats::filter()
## x dplyr::lag()     masks stats::lag()
## x purrr::lift()    masks caret::lift()
```

```r
# getting GermanCredit data
data("GermanCredit")
subCredit <- GermanCredit[,1:10] # selecting only first 10 columns
glimpse(subCredit) # exploring the data
```

```
## Rows: 1,000
## Columns: 10
## $ Duration                 <int> 6, 48, 12, 42, 24, 36, 24, 36, 12, 30, 12, 4~
## $ Amount                   <int> 1169, 5951, 2096, 7882, 4870, 9055, 2835, 69~
## $ InstallmentRatePercentage <int> 4, 2, 2, 2, 3, 2, 3, 2, 2, 4, 3, 3, 1, 4, 2,~
## $ ResidenceDuration        <int> 4, 2, 3, 4, 4, 4, 4, 2, 4, 2, 1, 4, 1, 4, 4,~
## $ Age                      <int> 67, 22, 49, 45, 53, 35, 53, 35, 61, 28, 25, ~
## $ NumberExistingCredits    <int> 2, 1, 1, 1, 2, 1, 1, 1, 1, 2, 1, 1, 1, 2, 1,~
## $ NumberPeopleMaintenance  <int> 1, 1, 2, 2, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1,~
## $ Telephone                <dbl> 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1,~
## $ ForeignWorker            <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,~
## $ Class                    <fct> Good, Bad, Good, Good, Bad, Good, Good, Good~
```

1. Examine the data structure that str() reveals. Also use the help() function to learn more about the GermanCredit data set. Summarize what you see in a comment.

```r
help(GermanCredit)
str(subCredit)
```

```
## 'data.frame':    1000 obs. of  10 variables:
##  $ Duration                 : int  6 48 12 42 24 36 24 36 12 30 ...
##  $ Amount                   : int  1169 5951 2096 7882 4870 9055 2835 6948 3059 5234 ...
##  $ InstallmentRatePercentage: int  4 2 2 2 3 2 3 2 2 4 ...
##  $ ResidenceDuration        : int  4 2 3 4 4 4 4 2 4 2 ...
##  $ Age                      : int  67 22 49 45 53 35 53 35 61 28 ...
##  $ NumberExistingCredits    : int  2 1 1 1 2 1 1 1 1 2 ...
##  $ NumberPeopleMaintenance  : int  1 1 2 2 2 2 1 1 1 1 ...
##  $ Telephone                : num  0 1 1 1 1 0 1 0 1 1 ...
##  $ ForeignWorker            : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ Class                    : Factor w/ 2 levels "Bad","Good": 2 1 2 2 1 2 2 2 2 1 ...
```

```r
summary(subCredit)
```

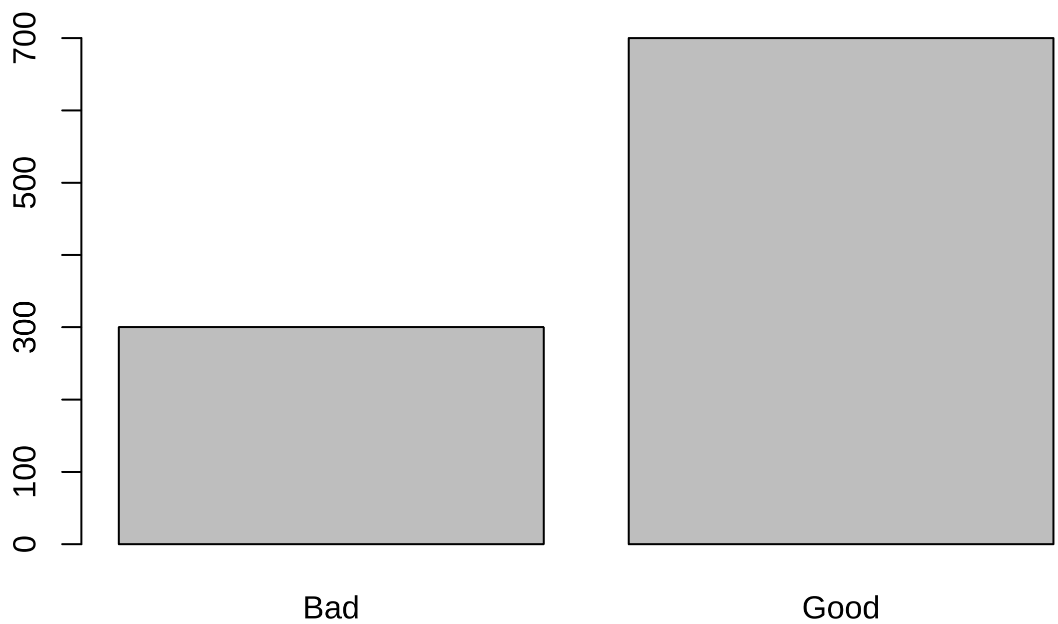
```
##     Duration        Amount      InstallmentRatePercentage ResidenceDuration
##  Min.   : 4.0   Min.   :  250   Min.   :1.000             Min.   :1.000
##  1st Qu.:12.0   1st Qu.: 1366   1st Qu.:2.000             1st Qu.:2.000
##  Median :18.0   Median : 2320   Median :3.000             Median :3.000
##  Mean   :20.9   Mean   : 3271   Mean   :2.973             Mean   :2.845
##  3rd Qu.:24.0   3rd Qu.: 3972   3rd Qu.:4.000             3rd Qu.:4.000
##  Max.   :72.0   Max.   :18424   Max.   :4.000             Max.   :4.000
##       Age        NumberExistingCredits NumberPeopleMaintenance   Telephone
##  Min.   :19.00   Min.   :1.000         Min.   :1.000           Min.   :0.000
##  1st Qu.:27.00   1st Qu.:1.000         1st Qu.:1.000           1st Qu.:0.000
##  Median :33.00   Median :1.000         Median :1.000           Median :1.000
##  Mean   :35.55   Mean   :1.407         Mean   :1.155           Mean   :0.596
##  3rd Qu.:42.00   3rd Qu.:2.000         3rd Qu.:1.000           3rd Qu.:1.000
```

```
##  Max.   :75.00   Max.   :4.000         Max.   :2.000           Max.   :1.000
##  ForeignWorker    Class
##  Min.   :0.000   Bad :300
##  1st Qu.:1.000   Good:700
##  Median :1.000
##  Mean   :0.963
##  3rd Qu.:1.000
##  Max.   :1.000
```

```
table(subCredit$Class)
```

```
##
##  Bad Good
##  300  700
```

```
plot(subCredit$Class)
```




```
# Summarize what you see in a comment
# GermanCredit data set has 1000 rows and 10 columns
# 1) class represents credit worthiness of the customer and it is a dichotomous variable
# having only two possible values-bad and good
# 2) all other variables except class are quantitative variables
# 3) There are 300 customers classified as bad and 700 customers as good
```

2. Use the createDataPartition() function to generate a list of cases to include in the training data. This function is conveniently provided by caret and allows one to directly control the number of training cases. It also ensures that the training cases are balanced with respect to the outcome variable. Try this: trainList <- createDataPartition(y=subCredit$Class,p=.40,list=FALSE)

```
set.seed(111)
# randomly sample for training dataset elements
# 40% of data from subCredit = 400 cases will used for building the model
trainList <- createDataPartition(y=subCredit$Class,p=.40,list=FALSE)
```

3. Examine the contents of trainList to make sure that it is a list of case numbers. With p=0.40, it should have 400 case numbers in it.

```
trainList[1:20] # first 20 case numbers
```

```
##  [1]  4  5  7  8 13 14 15 17 18 20 24 27 31 35 41 43 45 46 50 53
```

```
length(trainList)
```

```
## [1] 400
```

4. What is trainList? What do the elements in trainList represent? Which attribute is balanced in the trainList dataset?

```
# What is trainList?
#trainList is a vector consisting of case numbers that will used for building the SVM model

# What do the elements in trainList represent?
# case numbers/row indices from subCredit dataframe

# Which attribute is balanced in the trainList dataset?
# Class attribute
```

5. Use trainList and the square brackets notation to create a training data set called “trainSet” from the subCredit data frame. Look at the structure of trainSet to make sure it has all of the same variables as subCredit. The trainSet structure should be a data frame with 400 rows and 10 columns.

```
trainSet <- subCredit[trainList,]
head(trainSet)
```

```
##    Duration Amount InstallmentRatePercentage ResidenceDuration Age
## 4        42   7882                         2                 4  45
## 5        24   4870                         3                 4  53
## 7        24   2835                         3                 4  53
## 8        36   6948                         2                 2  35
## 13       12   1567                         1                 1  22
## 14       24   1199                         4                 4  60
##    NumberExistingCredits NumberPeopleMaintenance Telephone ForeignWorker Class
## 4                      1                       2         1             1  Good
## 5                      2                       2         1             1   Bad
```

```
## 7                          1                        1         1             1  Good
## 8                          1                        1         0             1  Good
## 13                         1                        1         0             1  Good
## 14                         2                        1         1             1   Bad
```

```r
dim(trainSet)
```

```
## [1] 400  10
```

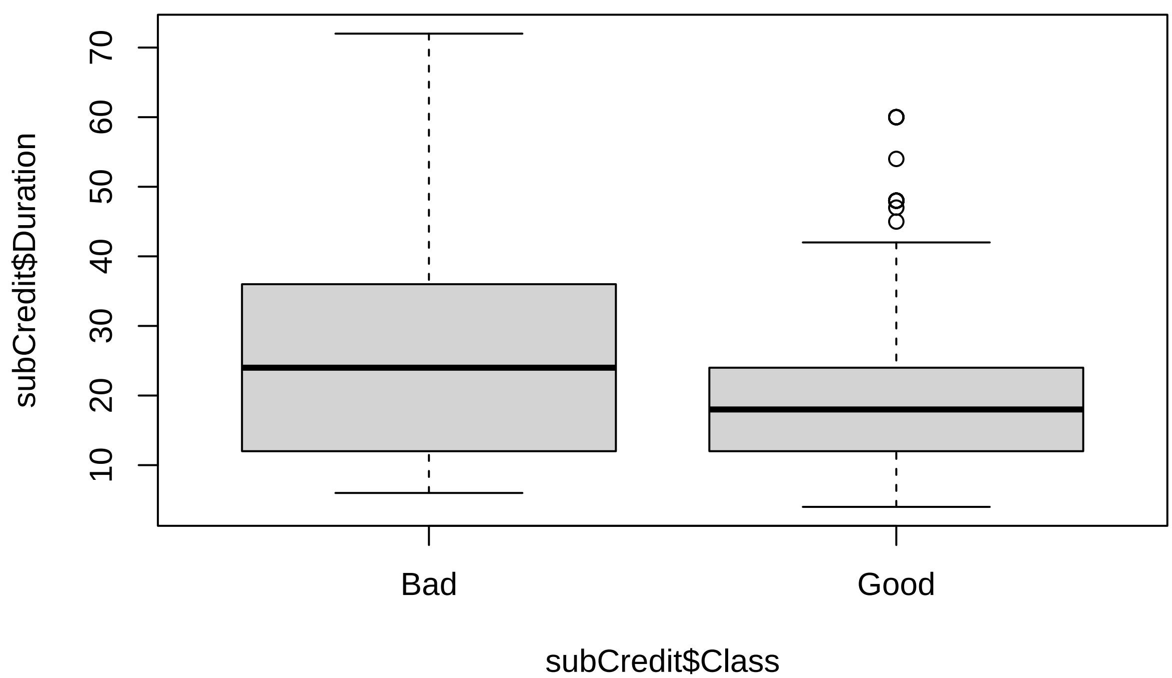
6. Use trainList and the square brackets notation to create a testing data set called "testSet" from the subCredit data frame. The testSet structure should be a data frame with 600 rows and 10 columns and should be a completely different set of cases than trainSet.

```r
testSet <- subCredit[-trainList,]
dim(testSet)
```

```
## [1] 600  10
```

7. Create and interpret boxplots of all the predictor variables in relation to the outcome variable (Class).

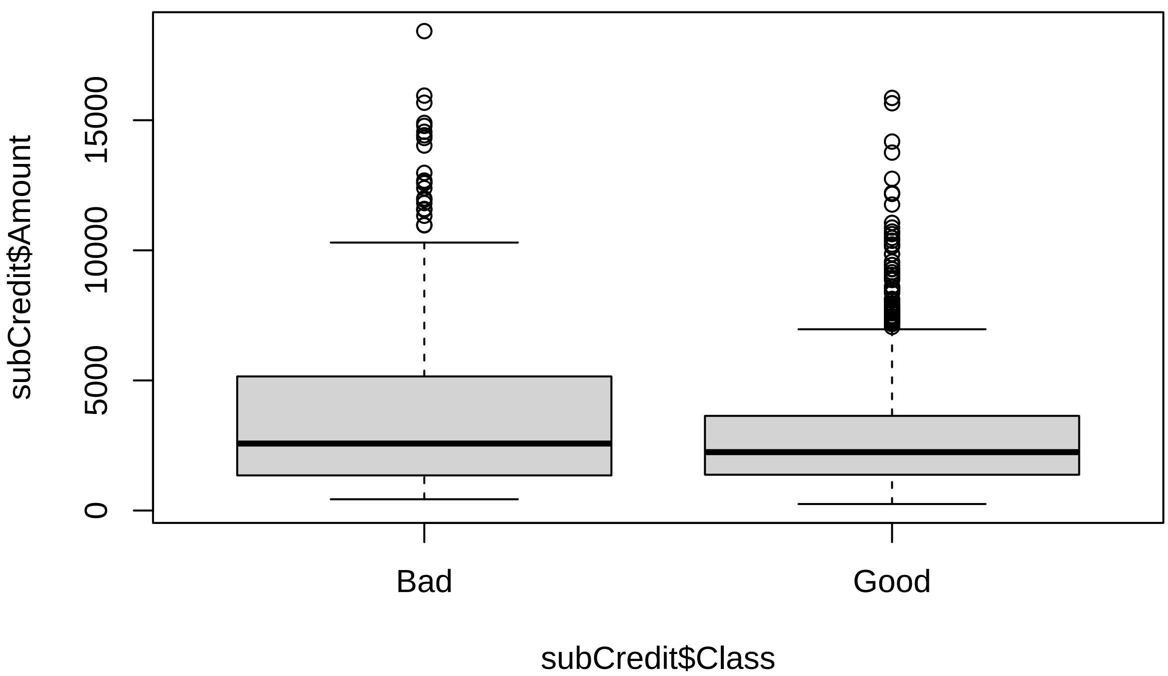
```r
boxplot(subCredit$Duration~subCredit$Class,data=subCredit)
```



```r
# Duration with class value Bad: data is not spread out. from the plot, we can
# see first and second quartile are close to median with no outliers. This implies
# most of the data is around mean/median

# Duration with class value Good: data is not spread out. from the plot, we can
# see first and second quartile are close to median but it has many outliers
# this results in right skewed distribution

boxplot(subCredit$Amount~subCredit$Class,data=subCredit)
```




```r
# Amount variable: from the plot we can notice that there are many outliers in
# the data resulting right skewed distribution

boxplot(subCredit$Age~subCredit$Class,data=subCredit)
```


```
# Age Variable is also having many outliers in the data resulting in right skewed
# distribution

# we need to handle outliers in order to get more accurate model while using these
# features in the model building
```

8. Train a support vector machine with the ksvm() function from the kernlab package. Make sure that you have installed and libraries the kernlab package. Have the cost be 5, and have ksvm do 3 cross validations (hint: try prob.model = TRUE)

```
?ksvm
svmModel <- ksvm(Class~.,data=trainSet,C=5,cross=3,prob.model=TRUE)
```

9. Examine the ksvm output object. In particular, look at the cross-validation error for an initial indication of model quality. Add a comment that gives your opinion on whether this is a good model.

```
svmModel
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 5
##
## Gaussian Radial Basis kernel function.
```

```
##  Hyperparameter : sigma =  0.124568955390236
##
## Number of Support Vectors : 251
##
## Objective Function Value : -846.3583
## Training error : 0.155
## Cross validation error : 0.279954
## Probability model included.
```

```
# cross validation error is approx 28%
# This indicates overall prediction error of all the 3 folds (cross=3)

# also, training error is 15.5%
# considering both the errors, it seems to be bad model as accuracy is
# is only ~56.5% (100% - (cross validaton error%+training error%)) on training data set
```

10. Predict the training cases using the predict command

```
preOut <- predict(svmModel,newdata=testSet, type="response")
preOut[1:20]
```

```
##  [1] Good Bad  Good Good Good Good Good Bad  Good Good Good Good Good Good Good
## [16] Good Good Bad  Good Good
## Levels: Bad Good
```

11. Examine the predicted out object with str( ). Then, calculate a confusion matrix using the table function.

```
str(preOut)  # preOut is a vector of factor variable with two levels-Bad/Good
```

```
##  Factor w/ 2 levels "Bad","Good": 2 1 2 2 2 2 2 1 2 2 ...
```

```
table(preOut,testSet$Class) #creating confusion matrix
```

```
##
## preOut Bad Good
##   Bad   49   52
##   Good 131  368
```

12. Interpret the confusion matrix and in particular calculate the overall accuracy of the model. The diag( ) command can be applied to the results of the table command you ran in the previous step. You can also use sum( ) to get the total of all four cells.

```
accuracy <- sum(diag(table(preOut,testSet$Class)))/sum(table(preOut,testSet$Class))
accuracy
```

```
## [1] 0.695
```

```r
# Interpret the confusion matrix
# diagonal elements indicate the right prediction of both values of Class
# off-diagonal elements are the wrong predictions
# so, accuracy = sum of all the right predictions/total predictions
```

13. Check you calculation with confusionMatrix() function in the caret package.

```r
confusionMatrix(preOut,testSet$Class)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction Bad Good
##       Bad   49   52
##       Good 131  368
##
##                Accuracy : 0.695
##                  95% CI : (0.6564, 0.7316)
##     No Information Rate : 0.7
##     P-Value [Acc > NIR] : 0.6244
##
##                   Kappa : 0.1697
##
##  Mcnemar's Test P-Value : 8.121e-09
##
##             Sensitivity : 0.27222
##             Specificity : 0.87619
##          Pos Pred Value : 0.48515
##          Neg Pred Value : 0.73747
##              Prevalence : 0.30000
##          Detection Rate : 0.08167
##    Detection Prevalence : 0.16833
##       Balanced Accuracy : 0.57421
##
##        'Positive' Class : Bad
##
```

```r
# accuracy=69.5% is same as that was calculated in previous steps
```