Intro to Data Science HW 8

```
# Enter your name here: Victoria Haley
```

Copyright Jeffrey Stanton, Jeffrey Saltz, and Jasmina Tacheva

Attribution statement: (choose only one and delete the rest)

```
# 1. I did this homework by myself, with help from the book and the professor.
```

Supervised learning means that there is a **criterion one is trying to predict**. The typical strategy is to **divide data** into a **training set** and a **test set** (for example, **two-thirds training** and **one-third test**), train the model on the training set, and then see how well the model does on the test set.

Support vector machines (SVM) are a highly flexible and powerful method of doing **supervised machine learning**.

Another approach is to use **partition trees (rpart)**

In this homework, we will use another banking dataset to train an SVM model, as well as an rpart model, to **classify potential borrowers into 2 groups of credit risk** – **reliable borrowers** and **borrowers posing a risk**. You can learn more about the variables in the dataset here: https://archive.ics.uci.edu/ml/datasets/Statlog+%28German+Credit+Data%29

This kind of classification algorithms is used in many aspects of our lives – from credit card approvals to stock market predictions, and even some medical diagnoses.

Part 1: Load and condition the data

A. Read the contents of the following .csv file into a dataframe called **credit**:

https://intro-datascience.s3.us-east-2.amazonaws.com/GermanCredit.csv

You will also need to install( ) and library( ) several other libraries, such as **kernlab** and **caret**.

```
library(kernlab)
library(caret)
```

```
## Loading required package: ggplot2

##
## Attaching package: 'ggplot2'

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

## Loading required package: lattice
```

```
library(lattice)
library(ggplot2)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --

## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.4.1
## v readr   2.1.3      v forcats 0.5.2
## v purrr   0.3.5
## -- 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()
```

```
credit <- read.csv("https://intro-datascience.s3.us-east-2.amazonaws.com/GermanCredit.csv")
```

B. Which variable contains the outcome we are trying to predict, **credit risk**? For the purposes of this analysis, we will focus only on the numeric variables and save them in a new dataframe called **cred**:

```
cred <- data.frame(duration=credit$duration,
                   amount=credit$amount,
                   installment_rate=credit$installment_rate,
                   present_residence=credit$present_residence,
                   age=credit$age,
                   credit_history=credit$number_credits,
                   people_liable=credit$people_liable,
                   credit_risk=as.factor(credit$credit_risk))
```

```
Error in data.frame(duration = credit$duration, amount = credit$amount, : object 'credit' not found
Traceback:


1. data.frame(duration = credit$duration, amount = credit$amount,
 .     installment_rate = credit$installment_rate, present_residence = credit$present_residence,
 .     age = credit$age, credit_history = credit$number_credits,
 .     people_liable = credit$people_liable, credit_risk = as.factor(credit$credit_risk))
```

C. Although all variables in **cred** except **credit__risk** are coded as numeric, the values of one of them are also **ordered factors** rather than actual numbers. In consultation with the **data description link** from the intro, write a comment identifying the **factor variable** and briefly **describe** each variable in the dataframe.

```
str(cred)
```

```
## 'data.frame':    1000 obs. of  8 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 ...
##  $ installment_rate : int  4 2 2 2 3 2 3 2 2 4 ...
##  $ present_residence: int  4 2 3 4 4 4 4 2 4 2 ...
##  $ age              : int  67 22 49 45 53 35 53 35 61 28 ...
##  $ credit_history   : int  2 1 1 1 2 1 1 1 1 2 ...
##  $ people_liable    : int  1 1 2 2 2 2 1 1 1 1 ...
##  $ credit_risk      : Factor w/ 2 levels "0","1": 2 1 2 2 1 2 2 2 2 1 ...
```

```
#$duration = how many months the credit has been active
#$amount = credit amount in $
#$installment_rate = rate of repayment in % of disposable income
#$present_residence = present residence since
#$age = age of borrower in years
#$credit_history = whether or not borrower has any current credits, late credits, fully paid off credit
#$people_liable = # of people being liable to provide maintenance for loan
#$credit_risk = whether or not a borrower is reliable or risky
#The factor variable is $credit_history, with the factors being 0-4 from no credits taken/all paid back
```

Part 2: Create training and test data sets

A. Using techniques discussed in class, create **two datasets** – one for **training** and one for **testing**.

```
set.seed(111)
trainList <- createDataPartition(y=cred$credit_risk, p=0.70, list = FALSE)
trainData <- cred[trainList,]
testData <- cred[-trainList,]
```

B. Use the dim( ) function to demonstrate that the resulting training data set and test data set contain the appropriate number of cases.

```
sum(dim(trainData)[1],dim(testData)[1])
```

```
## [1] 1000
```

```
#cred has 1000 observations, and the sum of both the training and testing data =1000
```

Part 3: Build a Model using SVM

A. Using the caret package, build a support vector model using all of the variables to predict **credit_risk**

```
SVMrisk_model <- train(credit_risk ~., data=trainData,
                       method = "svmRadial",
                       trControl=trainControl(method="none"),
                       preProcess=c("center", "scale"))
```

B. output the model

Hint: explore finalModel in the model that would created in F.

```
SVMrisk_model
```

```
## Support Vector Machines with Radial Basis Function Kernel
##
## 700 samples
##   7 predictor
##   2 classes: '0', '1'
##
## Pre-processing: centered (7), scaled (7)
## Resampling: None
```

```
#700 samples= the number of observations in the trainData object
#7 predictors = all attributes except for credit_risk
#2 classes = reliable or risky
#the data was also centered and scaled, with no resampling
```

Part 4: Predict Values in the Test Data and Create a Confusion Matrix

A. Use the predict( ) function to validate the model against test data. Store the predictions in a variable named **svmPred**.

```
svmPred <- predict(SVMrisk_model, newdata = testData)
```

B. The **svmPred** object contains a list of classifications for reliable (=0) or risky (=1) borrowers. Review the contents of **svmPred** using head( ).

```
head(svmPred)
```

```
## [1] 1 1 1 1 1 1
## Levels: 0 1
```

C. Explore the **confusion matrix**, using the caret package

```
confusionMatrix(svmPred, testData$credit_risk)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0   0   0
##          1  90 210
##
##                Accuracy : 0.7
##                  95% CI : (0.6447, 0.7513)
##     No Information Rate : 0.7
##     P-Value [Acc > NIR] : 0.5284
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.0
##             Specificity : 1.0
##          Pos Pred Value : NaN
##          Neg Pred Value : 0.7
##              Prevalence : 0.3
##          Detection Rate : 0.0
##    Detection Prevalence : 0.0
##       Balanced Accuracy : 0.5
##
##        'Positive' Class : 0
##
```

D. What is the **accuracy** based on what you see in the confusion matrix.

```
#The model predicted "1" correctly 210/300 times, which means it has an accuracy of 70%
```

E. Compare your calculations with the **confusionMatrix()** function from the **caret** package.

```
#The accuracy of the confusion matrix is 70%, which is the same as my calculations above.
```

F. Explain, in a block comment: 1) why it is valuable to have a "test" dataset that is separate from a "training" dataset, and 2) what potential ethical challenges this type of automated classification may pose.

```
#It's valuable to have a test dataset that is separate from a training dataset so that the algorithm do
#A potential ethical challenge this type of automated classification poses is with transparency and wit
```

Part 5: Now build a tree model (with rpart)

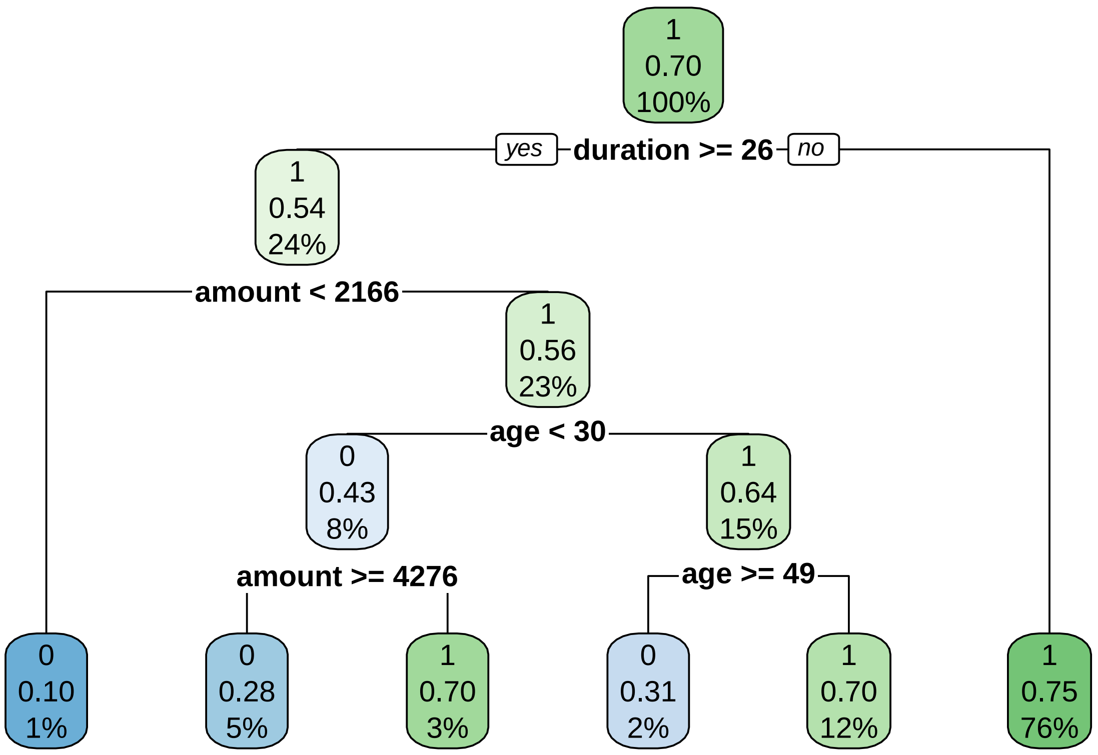
A. Build a model with rpart Note: you might need to install the e1071 package

```
library(e1071)
library(rpart)
model.rpart <- train(credit_risk ~.,
                     data=trainData,
                     method = "rpart",
                     trControl=trainControl(method="repeatedcv", number = 10),
                     tuneLength=50)
model.rpart
```

```
## CART
##
## 700 samples
##   7 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 1 times)
## Summary of sample sizes: 630, 630, 630, 630, 630, 630, ...
## Resampling results across tuning parameters:
##
##   cp            Accuracy   Kappa
##   0.0000000000  0.6585714  0.12931077
##   0.0005830904  0.6585714  0.12931077
##   0.0011661808  0.6585714  0.12694425
##   0.0017492711  0.6585714  0.12694425
##   0.0023323615  0.6614286  0.12831118
##   0.0029154519  0.6642857  0.13065182
##   0.0034985423  0.6657143  0.13322593
##   0.0040816327  0.6685714  0.13859772
##   0.0046647230  0.6685714  0.13859772
##   0.0052478134  0.6685714  0.13859772
##   0.0058309038  0.6814286  0.14354813
##   0.0064139942  0.6814286  0.14354813
##   0.0069970845  0.6814286  0.14136189
##   0.0075801749  0.6785714  0.11377442
##   0.0081632653  0.6871429  0.12313905
##   0.0087463557  0.6871429  0.12313905
##   0.0093294461  0.6942857  0.12640219
##   0.0099125364  0.6942857  0.12640219
##   0.0104956268  0.6942857  0.12640219
##   0.0110787172  0.7014286  0.13727001
##   0.0116618076  0.7014286  0.13727001
##   0.0122448980  0.7014286  0.13727001
##   0.0128279883  0.7014286  0.13727001
##   0.0134110787  0.7071429  0.13436895
##   0.0139941691  0.7071429  0.13436895
##   0.0145772595  0.7071429  0.13436895
##   0.0151603499  0.7071429  0.13436895
##   0.0157434402  0.7071429  0.13436895
##   0.0163265306  0.7085714  0.12821963
##   0.0169096210  0.7085714  0.12821963
##   0.0174927114  0.7085714  0.12821963
##   0.0180758017  0.7085714  0.12821963
##   0.0186588921  0.7085714  0.11796322
##   0.0192419825  0.7085714  0.11796322
##   0.0198250729  0.7085714  0.11796322
##   0.0204081633  0.7085714  0.11796322
##   0.0209912536  0.7085714  0.11796322
##   0.0215743440  0.6985714  0.06712783
##   0.0221574344  0.6985714  0.06712783
##   0.0227405248  0.6985714  0.06712783
##   0.0233236152  0.6985714  0.06712783
##   0.0239067055  0.6957143  0.05119863
##   0.0244897959  0.6957143  0.05119863
##   0.0250728863  0.6957143  0.05119863
##   0.0256559767  0.6957143  0.05119863
##   0.0262390671  0.6957143  0.05119863
##   0.0268221574  0.7000000  0.05197741
##   0.0274052478  0.7000000  0.05197741
##   0.0279883382  0.6957143  0.03023828
##   0.0285714286  0.6957143  0.03023828
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.02099125.
```

B. Visualize the results using rpart.plot()

```
library(rpart.plot)
rpart.plot(model.rpart$finalModel)
```



C. Use the **predict()** function to predict the testData, and then generate a confusion matrix to explore the results

```
predictValues <- predict(model.rpart,
                         newdata=testData)
confusionMatrix(predictValues, testData$credit_risk)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 12 14
##          1  78 196
##
##                Accuracy : 0.6933
##                  95% CI : (0.6378, 0.745)
##     No Information Rate : 0.7
##     P-Value [Acc > NIR] : 0.6264
##
##                   Kappa : 0.0837
##
##  Mcnemar's Test P-Value : 5.093e-11
##
##             Sensitivity : 0.13333
##             Specificity : 0.93333
##          Pos Pred Value : 0.46154
##          Neg Pred Value : 0.71533
##              Prevalence : 0.30000
##          Detection Rate : 0.04000
##    Detection Prevalence : 0.08667
##       Balanced Accuracy : 0.53333
##
##       'Positive' Class : 0
##
```

D. Review the attributes being used for this credit decision. Are there any that might not be appropriate, with respect to fairness? If so, which attribute, and how would you address this fairness situation. Answer in a comment block below

```
#If considering all of the attributes in the credit dataframe (not just cred), the attributes that migh
#I would remove these attributes from my model because the reason for the loan and the personal status
```