Intro to Data Science HW 9


```
# Enter your name here: Bhavya Shah
```

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

```
# 1. I did this homework with Shruti Rao and Pooja Kasar, but did not cut or paste any code.
```

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 a movie dataset to train an SVM model, as well as an rpart model, to **classify movies into 2 box office groups success** or **failure**.

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. The code below reads the contents of an Excel file into a dataframe called movies:

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

```
#install.packages('rio')
library(rio)
movies = rio::import("https://data-science-intro.s3.us-east-2.amazonaws.com/movies.xlsx")
head(movies)
```

```
##     id belongs_to_collection  budget homepage original_language_en
## 1 2798                     0 4.0e+06        0                    1
## 2  804                     0 1.5e+07        0                    1
## 3 2132                     0 2.7e+07        1                    1
## 4  724                     0 1.1e+07        0                    1
## 5 2649                     0 6.5e+07        1                    1
## 6  780                     0 0.0e+00        1                    1
##      original_title overview popularity production_companies release_date
## 1       The Misfits        1   0.884241                    0   2010-04-15
## 2        Robin Hood        1   0.414793                    1   2006-10-12
## 3       About a Boy        1   0.201582                    1   2014-06-05
## 4 Money for Nothing        1   1.833185                    1   2006-03-16
## 5    This Means War        1   7.321152                    1   2016-10-12
## 6 Definitely, Maybe        1   1.760091                    1   2004-12-23
##   runtime   status tagline                                 title success
## 1       0 Released       0   Hooked on the Game 2. The Next Level       0
## 2       0 Released       0                             Mechenosets       0
## 3       0 Released       0                             All at Once       0
## 4      65 Released       0                Nikitich and The Dragon       0
## 5      66 Released       1                  My Life as a Zucchini       0
## 6      72 Released       0 Alesha Popovich and Tugarin the Dragon       0
```

B. Which variable contains the outcome we are trying to predict, **whether a movie is a financial success or not**? For the purposes of this analysis, we will focus only on the numeric variables and save them in a new dataframe called **mov**:

```
mov <- data.frame(belongs_to_collection=movies$belongs_to_collection,
                  budget=movies$budget,
                  homepage=movies$homepage,
                  original_language_en=movies$original_language_en,
                  overview=movies$overview,
                  popularity=movies$popularity,
                  production_companies=movies$production_companies,
                  runtime=movies$runtime,
                  tagline=movies$tagline,
                  success=as.factor(movies$success))
```

C. What is the total number of observations in **mov**? Show your code.

```
str(mov)
```

```
## 'data.frame':    1374 obs. of  10 variables:
##  $ belongs_to_collection: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ budget               : num  4.0e+06 1.5e+07 2.7e+07 1.1e+07 6.5e+07 0.0 5.5e+07 3.2e+07 1.
5e+07 2.3e+07 ...
##  $ homepage             : num  0 0 1 0 1 1 1 1 1 0 ...
##  $ original_language_en : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ overview             : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ popularity           : num  0.884 0.415 0.202 1.833 7.321 ...
##  $ production_companies : num  0 1 1 1 1 1 1 1 1 ...
##  $ runtime              : num  0 0 0 65 66 72 75 76 76 76 ...
##  $ tagline              : num  0 0 0 0 1 0 1 1 1 1 ...
##  $ success              : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
```

```
# The total number of observations are 1374.
```

Part 2: Create training and test data sets

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

```
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
library(kernlab)
```

```
##
## Attaching package: 'kernlab'
```

```
## The following object is masked from 'package:ggplot2':
##
##     alpha
```

```
trainlib <- createDataPartition(y=movies$success,p=.40,list=FALSE)
trainSet <- mov[trainlib,]
testSet <- mov[-trainlib,]
```

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

```
dim(trainSet)
```

```
## [1] 550  10
```

```
dim(testSet)
```

```
## [1] 824  10
```

Part 3: Build a Model using SVM

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

```
library(kernlab)
library(caret)
trainctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)
#It controls the computational nuances of the train() method.
set.seed(3233)
svmMod <- train(success ~., data = mov,
 method = "svmLinear", trControl=trainctrl,
 preProcess = c("center", "scale"), tuneLength = 10)
```

B. Output the model you created in the previous step.

```
svmMod
```

```
## Support Vector Machines with Linear Kernel
##
## 1374 samples
##    9 predictor
##    2 classes: '0', '1'
##
## Pre-processing: centered (9), scaled (9)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 1237, 1237, 1236, 1236, 1236, 1237, ...
## Resampling results:
##
##   Accuracy  Kappa
##   0.553401  0.1050096
##
## Tuning parameter 'C' was held constant at a value of 1
```

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

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

```
svmPred <- predict(svmMod, newdata = testSet)
```

B. The **svmPred** object contains a list of classifications for successful (=1) or unsuccessful (=0) movies. Review the contents of **svmPred** using **head( )**.

```
head(svmPred)
```

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

C. Create a **confusion matrix**, using the **table()** function. Write a comment to explain what each of the 4 numbers means.

```
table(svmPred, testSet$success)
```

```
##
## svmPred   0   1
##       0 206 189
##       1 176 253
```

```
#For each model, we can compare its predicted values to the actual values in the test dataset. T
his comparison gives us an idea of how well our model is working. The rows are the actual value
s, whereas the columns are the anticipated values.
#The model has predicted 0 as 0, 210 times and 0 as 1, 170 times.
#The model has predicted 1 as 0, 181 times and 1 as 1, 263 times.
```

D. What is the **accuracy** based on what you see in the confusion matrix? Show your calculation.

```
sum(diag(table(svmPred, testSet$success)) / sum(table(svmPred, testSet$success)))
```

```
## [1] 0.5570388
```

```
# The accuracy of the model is 57.40%
```

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

```
library(caret)
confusionMatrix(svmPred, testSet$success)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 206 189
##          1 176 253
##
##                Accuracy : 0.557
##                  95% CI : (0.5224, 0.5913)
##     No Information Rate : 0.5364
##     P-Value [Acc > NIR] : 0.1245
##
##                   Kappa : 0.1114
##
##  Mcnemar's Test P-Value : 0.5299
##
##             Sensitivity : 0.5393
##             Specificity : 0.5724
##          Pos Pred Value : 0.5215
##          Neg Pred Value : 0.5897
##              Prevalence : 0.4636
##          Detection Rate : 0.2500
##    Detection Prevalence : 0.4794
##       Balanced Accuracy : 0.5558
##
##        'Positive' Class : 0
##
```

```
# We obtained the expected accuracy of 57.40 from the calculation based on the above matrix.
```

F. Explain, in 2 comments:
1) why it is valuable to have a test dataset that is separate from a training dataset, and
2) what potential ethical challenges may this type of automated classification pose? E.g., if it is used on people rather than movies?

```
# 1)To avoid overfitting your model and to appropriately evaluate it.
```

```
# 2)If we utilize the same dataset for training and testing, the results will be skewed, giving the impression that the model is more accurate.It could be biased towards persons from a given l ocation, race, gender, or color and provide output that is biased towards only that training pop ulation.
```

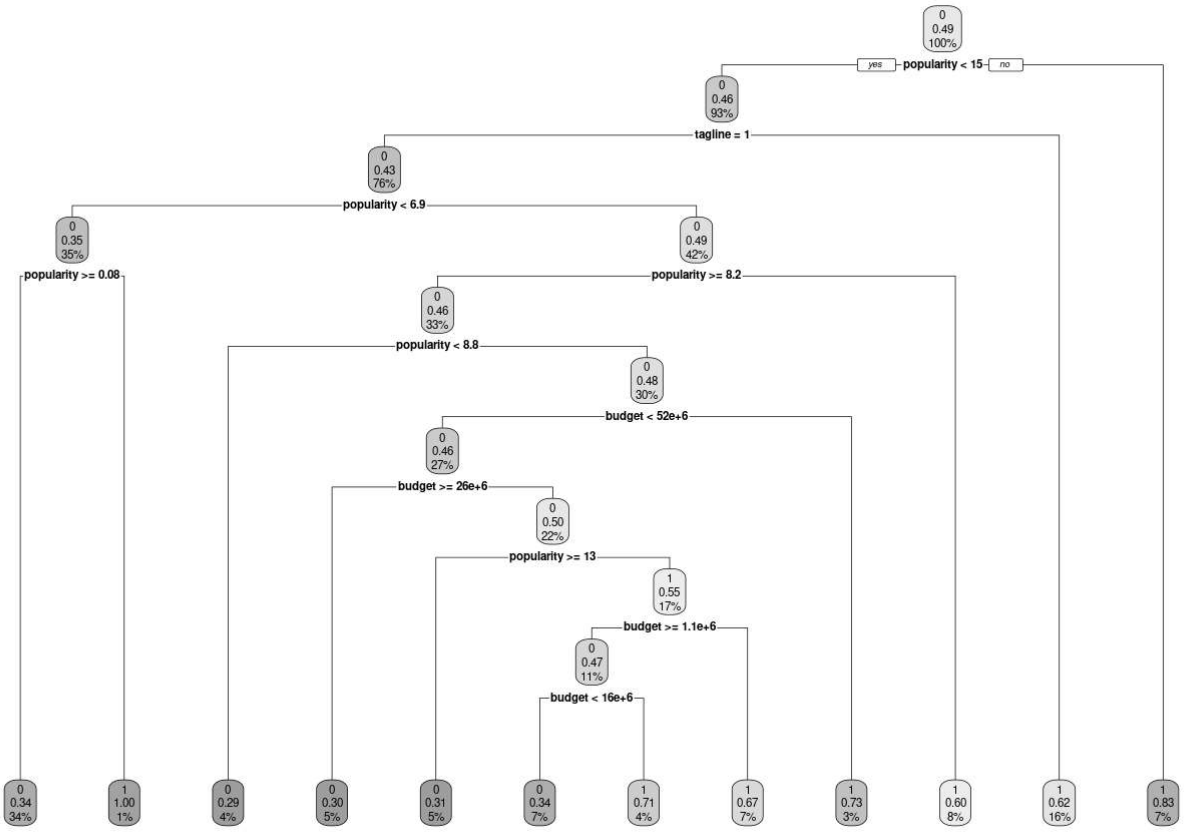
Part 5: Now build a tree model (with rpart)

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

```
library(e1071)
library(rpart)
set.seed(100)
tree_mod = rpart(success ~., data = trainSet, method="class", minsplit = 10, minbucket=3)
```

B. Visualize the results using **rpart.plot()**

```
library(rpart.plot)
rpart.plot(tree_mod)
```



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

```
predOu <- predict(tree_mod,newdata=trainSet, type="class")
table(trainSet$success, predOu)
```

```
##    predOu
##       0   1
##   0 200  81
##   1  99 170
```

```
(187+173)/(187+173+79+112) #The accuracy of training dataset is 65.33%
```

```
## [1] 0.6533575
```

```
predOu2 <- predict(tree_mod, newdata = testSet, type = "class")
table(testSet$success, predOu2)
```

```
##    predOu2
##       0   1
##   0 200 182
##   1 211 231
```

```
(220+217)/(220+217+209+177) #The accuracy of test dataset is 53.30%
```

```
## [1] 0.5309842
```

```
confusionMatrix(predOu2, testSet$success)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 200 211
##          1 182 231
##
##                Accuracy : 0.5231
##                  95% CI : (0.4883, 0.5576)
##     No Information Rate : 0.5364
##     P-Value [Acc > NIR] : 0.7892
##
##                   Kappa : 0.0459
##
##  Mcnemar's Test P-Value : 0.1578
##
##             Sensitivity : 0.5236
##             Specificity : 0.5226
##          Pos Pred Value : 0.4866
##          Neg Pred Value : 0.5593
##              Prevalence : 0.4636
##          Detection Rate : 0.2427
##    Detection Prevalence : 0.4988
##       Balanced Accuracy : 0.5231
##
##        'Positive' Class : 0
##
```

D. Review the accuracy of the two models - it is not very high. What are some strategies you could use to improve the quality of the models? Answer in a comment block below.

```
# The accuracy of the SVM model is 57.40%, whereas the accuracy of the tree model is 53.30%.
#Methods for Improving Model Quality
# 1) Avoidable bias should be around zero. A single number evaluation metric can be used to opti
mize the model.
# 2) By dividing the data into four datasets( train, train-dev, validation, and test), we can en
sure that the training set has sufficient number of observations and that the validation and tes
t sets are drawn from the same distribution.
# 3) Using evaluation measure we can assess the error with human error serving as a baseline.
```