Intro to Data Science HW 9


```
# Enter your name here: Tao Pang
```

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

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

```
## 载入需要的程辑包：ggplot2
```

```
## 载入需要的程辑包：lattice
```

```
library(kernlab)
```

```
##
## 载入程辑包：'kernlab'
```

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

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.xl
sx")
```

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**:

```
#we are trying to predict success variable inside movies
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.

```
#use nrow function to find out the total number of observations in mov which is 1374
nrow(mov)
```

```
## [1] 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**.

```
#creating two data set one for training and one for testing, trainingset and testings
et
training <- createDataPartition(y=mov$success,p=.40,list=FALSE)
#testing <- createDataPartition(y=mov$success,p=.60,list=FALSE)
trainingset <- mov[training,]
testingset <- mov[-training,]
```

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

```
#use dim function to demonstrate the resulting training data set and test data set co
ntain the appropriate number of cases
dim(trainingset)
```

```
## [1] 551  10
```

```
dim(testingset)
```

```
## [1] 823  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 caret package previously and for this question, just use the model from lab
 to create appropriate model for this question
ksvm_model <- ksvm(data = trainingset, success ~ ., C = 5, cross = 3, prob.model = TR
UE)
```

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

```
# output the model that is done with last question
ksvm_model
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 5
##
## Gaussian Radial Basis kernel function.
##  Hyperparameter : sigma =  0.324469486092678
##
## Number of Support Vectors : 467
##
## Objective Function Value : -1832.523
## Training error : 0.284936
## Cross validation error : 0.475548
## Probability model included.
```

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.**

```
#use the predict function to validate the model against test data
svmPred <-predict(ksvm_model,testingset)
```

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

```
# use head to review the contents of svmPred which is created last step
head(svmPred)
```

```
## [1] 1 0 0 1 1 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.

```
# the four number can add together to be total number
#170 means predicted to fail and actual it is fail
#263 means predicted to success and actual it is success
#227 means predicted to be success but actual is fail
#163 means predicted to fail but actual is success.
table(svmPred, testingset$success)
```

```
##
## svmPred   0   1
##       0 188 192
##       1 209 234
```

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

```
# the accuracy is pretty much the same as the confusion matrix which is about 52.61%,
it is very close to 50%

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

```
## [1] 0.5127582
```

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

```
str(testingset$success)
```

```
##  Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
```

```
#use the function confusionMatrix to compare the calculation, here is  52.61%, it is
 almost the same as the previous one
confusionMatrix(svmPred, testingset$success)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 188 192
##          1 209 234
##
##                Accuracy : 0.5128
##                  95% CI : (0.478, 0.5474)
##     No Information Rate : 0.5176
##     P-Value [Acc > NIR] : 0.6233
##
##                   Kappa : 0.0229
##
##  Mcnemar's Test P-Value : 0.4243
##
##             Sensitivity : 0.4736
##             Specificity : 0.5493
##          Pos Pred Value : 0.4947
##          Neg Pred Value : 0.5282
##              Prevalence : 0.4824
##          Detection Rate : 0.2284
##    Detection Prevalence : 0.4617
##       Balanced Accuracy : 0.5114
##
##        'Positive' Class : 0
##
```

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) if we can have a new dataset where contains testing it will be more useful, tain
ing set is not fully random, so we need a testing set to use inside the prediction
```

```
# 2)it may find out many problems about human beings, when it comes to people we can
 know more detailed information by separate testing from training.
```

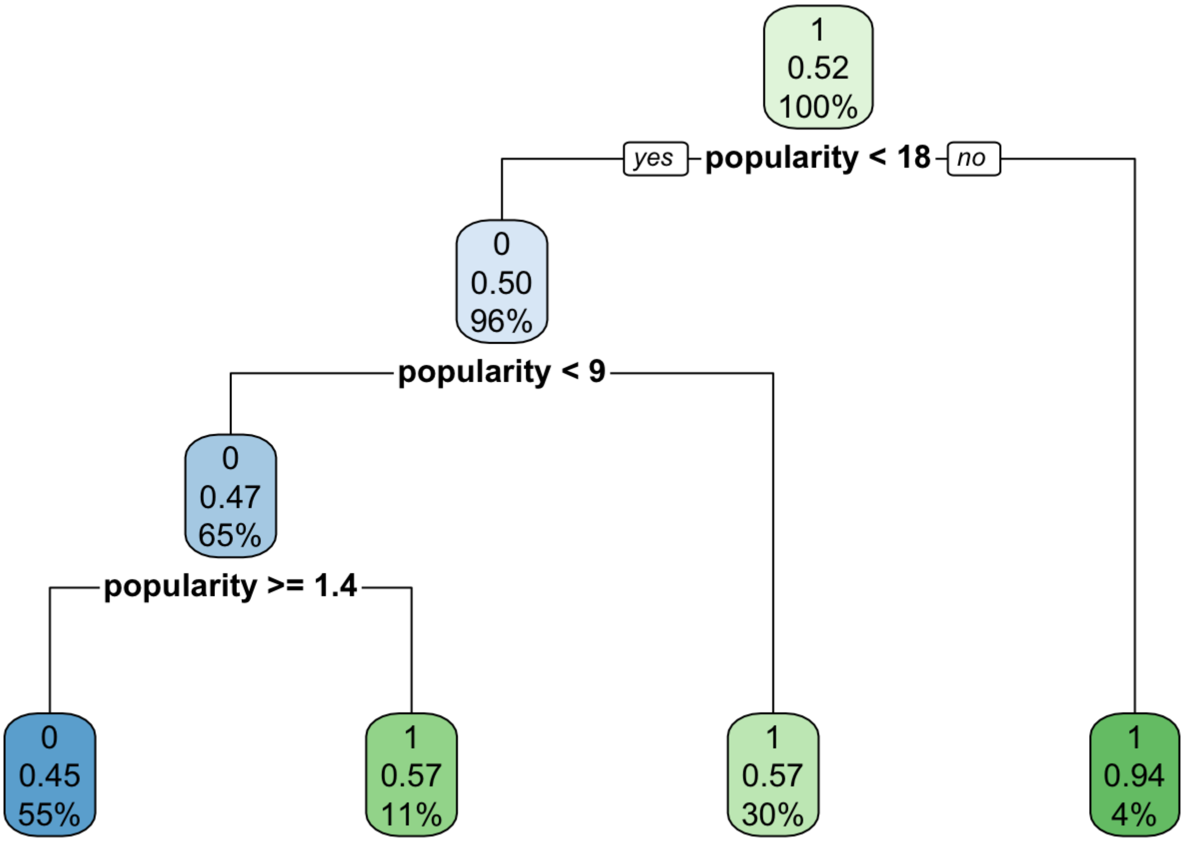
Part 5: Now build a tree model (with rpart)

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

```
#install two packages e1071 and rpart first and then library both of them, afther tha
t use the rpart function
library(e1071)
library(rpart)
cartTree <- train(success ~., data=testingset, method="rpart")
```

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

```
#use rpart.plot to show the tree model
library(rpart.plot)
rpart.plot(cartTree$finalModel)
```



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

```
#use predict function to predict the data and also use confusion matrix to show the r
esults
prediction <- predict(cartTree, testingset)
str(prediction)
```

```
##  Factor w/ 2 levels "0","1": 2 2 1 1 1 1 2 1 1 1 ...
```

```
confusionMatrix(prediction, testingset$success)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 249 201
##          1 148 225
##
##                Accuracy : 0.5759
##                  95% CI : (0.5413, 0.61)
##     No Information Rate : 0.5176
##     P-Value [Acc > NIR] : 0.0004484
##
##                   Kappa : 0.1547
##
##  Mcnemar's Test P-Value : 0.0053776
##
##             Sensitivity : 0.6272
##             Specificity : 0.5282
##          Pos Pred Value : 0.5533
##          Neg Pred Value : 0.6032
##              Prevalence : 0.4824
##          Detection Rate : 0.3026
##    Detection Prevalence : 0.5468
##       Balanced Accuracy : 0.5777
##
##        '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.

```
# make the data inside the tree model more accurate, i think the dataset inside the t
ree model can be well revised in order to showing a more accurate model.
```