

**BUDT 758T**

**assignment #4: 100 PTS**

The goal of this homework is to introduce you to working with classification and regression trees, as well as introducing the concepts of k-nearest neighbors (kNN) modeling. As with Assignment 3, you will need to create random partitions of a data set, build your model on the training data set, and then compute prediction errors using the test data set. However, this assignment will require an additional data partition step: validation data. You are required to complete this assignment in R—be sure to include the code you used and output any results you use!

**The Data**

Average theater movie revenues have been in decline in recent years, even as major blockbusters and superhero franchises soar. To investigate this phenomenon, major motion picture studios in the United States want to identify factors that lead to a movie being financially successful (where “successful” is defined as a movie making at least twice in revenue than it had for a budget). To this end, studio executives have collected information on thousands of movies from 2000 to 2017, with information collected from the popular movie database site TMDb (The Movie Database).

This data is collected in the CSV file *movies\_data.csv* on ELMS. It is based on a popular Kaggle database that can be found at <https://www.kaggle.com/tmdb/tmdb-movie-metadata/version/2>.

The variables in *movie\_data.csv* are defined as follows:

|  |  |
| --- | --- |
| **Variable** | **Description** |
| *id* | An ID number describing the location of the movie in the data set |
| *title* | The title of the movie (when released) |
| *genre* | A primary genre of the movie |
| *budget* | The budget of the movie |
| *revenue* | The total revenue of the movie |
| *production\_companies* | The number of production companies involved in making the movie |
| *united\_states* | Whether any part of the film was produced in the United States ("Yes") or not ("No") |
| *english* | Whether the film's dialogue is in English ("Yes") or not ("No") |
| *title\_change* | Whether the film was released under a different title than it was originally produced ("Yes") or not ("No") |
| *popularity* | A TMDb value for how popular the movie is on their site (where a larger number means the movie is more popular) |
| *vote\_average* | The average voter rating for the movie |
| *vote\_count* | The total number of votes the movie has received |
| *month* | The month in which the movie was released (1 = January through 12 = December) |
| *year* | The year in which the movie was released |
| *runtime* | The total runtime of the movie (in minutes) |
| *successful* | Was the movie financially successful (1) or not (0) |

**Assignment**

Please answer all questions in the dedicated space and upload on Canvas. Please ensure that your numbering of questions matches those below. You must include your R code, but because this assignment has significantly more R code than previous assignments, you are welcome to include your full code at the end of the assignment file rather than including it with the appropriate question. However, you should make sure any output that is requested or necessary to answer the question is including with the question. Any additional output you wish to provide may be included at the end of your assignment in an appendix, if you wish.

Remember: you are allowed to consult with others in the class on this assignment, but all submitted work must be your own (and don’t forget to include the names of anyone you consulted in the last question!).

1. **10 points: Data Preparation**
   1. Read the data set into R.

data <- read.csv("movies\_data.csv")

* 1. Change the variable *successful* to a factor variable. Use this new variable for the remainder of the assignment. (Remember to attach it to your data set before partitioning!)

data$successful <- as.factor(data$successful)

* 1. Set the seed in R to 3730.

set.seed(3730)

* 1. Randomly partition the data set in the following order (note that if you do *not* follow this order, many of the questions in this assignment will not make sense to you!):
     1. Split 25% of the observations in *movies\_data* to use as testing data. Using these observations, create a testing data set called *movies\_test.*

num\_obs <- nrow(data)

test\_obs <- sample(num\_obs, 0.25\*num\_obs)

movies\_test <- data[test\_obs, ]

* + 1. Save the remaining 75% of the data as *movies\_rest*.

movies\_rest <- data[-test\_obs, ]

* + 1. Split 25% of the observations in *movies\_rest* to use as validation data. Using these observations, create a validation data set called *movies\_valid.*

num\_obs <- nrow(movies\_rest)

valid\_obs <- sample(num\_obs, 0.25\*num\_obs)

movies\_valid <- movies\_rest[valid\_obs, ]

* + 1. Save the remaining data as *movies\_train*.

movies\_train <- movies\_rest[-valid\_obs, ]

* 1. How many observations do you now have in (1) the full data set, (2) the training data set, (3) the validation data set, and (4) the testing data set?

(1) the full data set - 3492

(2) the training data set - 1965

(3) the validation data set - 654

(4) the testing data set - 873

1. **20 points: Preparation to build a classification tree to predict *successful*.**
   1. Based on the definitions of the variables given above:
      1. Should we include both *budget* and *revenue* in our tree as possible predictor (X) variables? Why or why not?

No, we should use only one of them because both budget and revenue together define our target ‘success’.

* + 1. Should we include *title* or *id* in our tree as possible predictor (X) variables? Why or why not?

No, because intuitively they are not adding any value to the model. They can’t be considered as features.

* 1. We changed *successful* to a factor variable in Question 1, part (b) above.
     1. Do we need to change any other variables in our data set to factor before we run our tree? Why or why not?

Yes, month and year to factors.

* + 1. If we had not changed *successful* to a factor, how would our modeling change? Could we still predict *successful* using a tree? Explain your answer.

If we had not changed successful to a factor, our modeling is no longer a classification problem and it’s a regression problem. We could be using a regression tree instead of classification tree to predict successful.

It gives the mean for each terminal node which is value between 0 and 1. If this value is greater than 0.5 we could say all those data points can be classified as successful=1 and vice-versa.

1. **15 points: Build a classification tree to predict *successful* using all other variables except *id, title,* and *revenue* using the training data. Run a complete tree rather than defining a stopping rule.**
   1. Plot the full tree. What size is the tree?

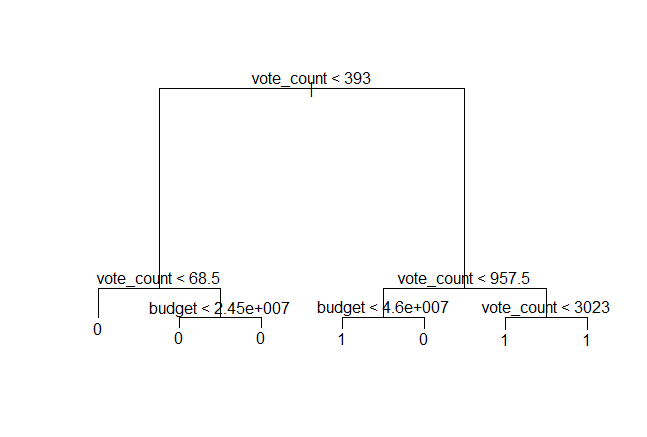
library(tree)

movies.tree=tree(successful~.-id-title-revenue,movies\_train)

summary(movies.tree)

plot(movies.tree)

text(movies.tree,pretty=1)



The size of tree is number of terminal nodes i.e. 7.

* 1. According to this tree, which variable in the data set do you think is the most important for deciding if a movie will be very successful or not? Explain your answer.

vote\_count is the most important variable in the data set for deciding if a movie will be successful or not because it is making most of the distinctive splits.

* 1. Are all the splits in the tree useful at a cutoff of 0.5? Explain your answer. (This means did every split in the tree give us additional information/predictive power, assuming we were going to use a cutoff of 0.5 to classify predictions.)

No, all the splits in the tree are not useful. Some splits like (vote\_count<68.5, budget < 2.45e\_007 & vote\_count < 3023) give rise to nodes both having the same labels. Hence, these splits are not giving us additional predictive power at cutoff of 0.5.

1. **30 points: Using your classification tree from Question 3, consider all possible tree sizes (up to the full tree size) to determine which tree size is best.**
   1. Before answering this question:
      1. Which data set should you use to answer the question?

Validation data set (movies\_valid) to compare the accuracies among the trained models.

* + 1. What is the baseline for this data set?

The baseline for this data set is the most common class in the validation data. Here it is ‘0’.

* + 1. What is the baseline accuracy for this data set?

The baseline accuracy for this data set is the validation accuracy from a tree of size 1. Here it is close to 61%.

* 1. If you run the *prune.tree()* command with *best=1* and try to calculate accuracy on the tree, you will get an error in R. Why? Is this a problem for us if we need to evaluate a tree size of 1?

R throws an error because a tree of size 1 is same as no splits at all or considering the whole data set all together. This won’t be a problem because conceptually we can evaluate the tree size of 1 using the baseline class or the majority class in the data.

* 1. Choose the best tree size based on accuracy with a cutoff of 0.5.

for(i in 2:7) {

name <- paste("movies.pruned", i, sep="\_")

name=prune.tree(movies.tree,best=i)

tree\_preds <- predict(name,newdata=movies\_valid)

tree\_probs=tree\_preds[,2]

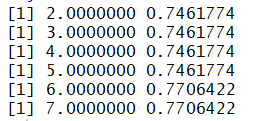
tree\_class=ifelse(tree\_probs>0.5,1,0)

accuracy <- paste("acc\_tree", i, sep="\_")

accuracy=sum(ifelse(tree\_class==movies\_valid$successful,1,0))/nrow(movies\_valid)

print(c(i,accuracy))

}



* + 1. Which size tree did you choose? Why?

Tree of size 6 because it has the highest validation accuracy of 0.7706422.

Since, tree of size 6 and size 7 have the same validation accuracy, it’s best practice always to go for the smaller tree with the same accuracy.

* + 1. Do you think R made a good choice at what size to stop growing the tree? Why or why not?

No, because it should have stopped at size 6 that has the same highest accuracy as size 7 tree. The best practice is to always go for the smaller tree.

* + 1. Do your accuracy values support your answer to Question 3, part (c) above? Why or why not?

Yes, the accuracy values support my answer to Question 3, part (c) above. Because some new splits formed haven’t added any additional predictive power. For example, the tree of size 7 has the same accuracy as tree of size 6.

* 1. Retrain the classification tree using the tree size you chose in part (c) and using the full *movies\_rest* data set.

movies.tree\_retrain=tree(successful~.-id-title-revenue,movies\_rest)

movies.pruned\_retrain=prune.tree(movies.tree\_retrain,best=6)

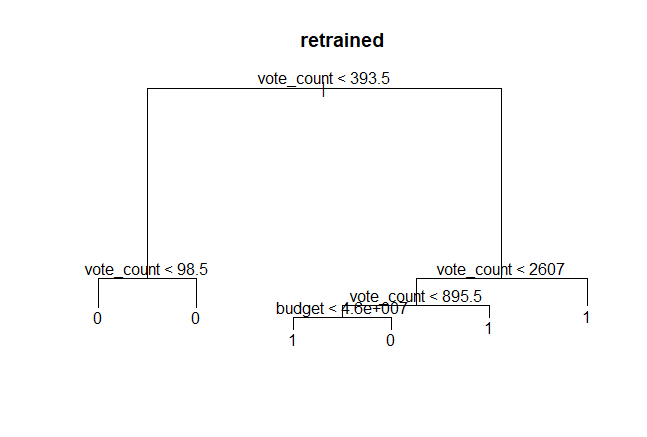
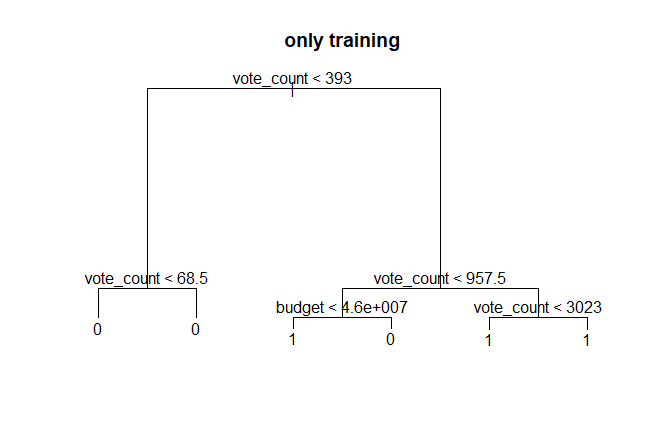
summary(movies.pruned\_retrain)

plot(movies.pruned\_retrain)

text(movies.pruned\_retrain,pretty=1)

title(main='retrained')

* + 1. Plot both the new tree and the equivalent tree from the training data (the tree of the same size) below. Do you think the trees agree with one another? Explain your answer.

Yes, I think the trees agree with each other. Because their class predictions are the same for all the terminal nodes. There is a just a little difference in the variable dimension at the splits (e.g. vote\_count < 98.5 vs vote\_count < 685.)

* + 1. In your opinion (note that there is no right or wrong answer here!), was retraining the tree a useful step here? Why or why not?

Yes, I think retraining the tree is a useful step here because it is giving us a more polished tree with little bit of extra information at the varibles split. Also, it is best practice to retrain the model.

1. **15 points: Run the kNN algorithm for classification on the training data with the numeric variables in the data set (except *revenue*). Run the algorithm five times with five different values of k: 1, 3, 5, 10, and 25.**

library(class)

colnames(data)

str(data)

train.X=movies\_train[,c(4,6,10,11,12,15)]

valid.X=movies\_valid[,c(4,6,10,11,12,15)]

test.X=movies\_test[,c(4,6,10,11,12,15)]

rest.X=movies\_rest[,c(4,6,10,11,12,15)]

train.target=movies\_train$successful

valid.target=movies\_valid$successful

test.target=movies\_test$successful

rest.target=movies\_rest$successful

for(i in c(1,3,5,10,25)) {

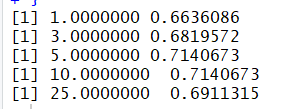
knn.pred=knn(train.X,valid.X,train.target,k=i)

cm = table(valid.target,knn.pred,dnn=c("Actual","Predicted"))

accuracy = (cm[1,1] + cm[2,2])/sum(cm)

print(c(i,accuracy))

}



* 1. Using accuracy and a cutoff of 0.5, what is the best value of k? How do you know?

The best value of k is 5. We know this because it has got the highest validation accuracy.

* 1. Based on your accuracy results (and a cutoff of 0.5), would you suggest trying a value of k higher than 25? Why or why not?

No, because the validation accuracy will keep decreasing further after k=25.

1. **10 points: Consider your retrained tree and your best kNN (meaning kNN with your optimal value of k).**
   1. Are these guaranteed to be the best tree model and the best kNN for this data? Explain your answer.

Yes, we can gurantee them to be the best tree and best kNN for this data because they have the highest validation accuracies.

* 1. Which of the two models would you suggest we use in practice? Explain your answer.

## best tree

name <- paste("movies.pruned", i, sep="\_")

name=prune.tree(movies.tree,best=i)

tree\_preds <- predict(movies.pruned\_retrain,newdata=movies\_test)

tree\_probs=tree\_preds[,2]

tree\_class=ifelse(tree\_probs>0.5,1,0)

tree\_accuracy=sum(ifelse(tree\_class==movies\_test$successful,1,0))/nrow(movies\_test)

tree\_accuracy

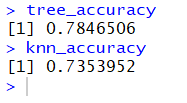
## best knn

knn.pred=knn(rest.X,test.X,rest.target,k=5)

cm = table(test.target,knn.pred,dnn=c("Actual","Predicted"))

knn\_accuracy = (cm[1,1] + cm[2,2])/sum(cm)

knn\_accuracy



We should use the tree model because it has a higher accuracy on testing data compared to the knn.