Applying Machine Learning

Framing the Main Question

The original question for this capstone project was:

What factors affect how popular a YouTube video will be and, using such factors, can we predict popularity for any video?

To reframe this question as a machine learning problem, the following was used:

Which tags predict popularity for a YouTube video?

This is a supervised learning problem because all the data is labeled and the algorithms employed learn to predict the output from the input data. More specifically, the algorithms, or models, learned from the training data set to make predictions on the unseen data of the testing data set. The best predictive model was selected after it achieved an acceptable level of performance.

For this project, a classification supervised learning problem was used because the output variable, in this case *popularity*, is a categorical variable in that a video was ranked as being either popular or unpopular.

The Variables

Using the original data set, the *tags* factor will be transformed through text analytics to become the independent variables or predictors. The dependent variable or outcome will be a single measure called *popularity*, which will be a combination of three factors: *views*, *likes*, and *comments*.

The Approach: Machine Learning Techniques and Evaluation

For this project, three machine learning techniques were applied in addition to constructing a simple baseline model.

The baseline model

The baseline model that always predicts an unpopular video was built by making tables of just the outcome variable.

In the training set, there are 18,024 observations that are unpopular and 10,641 observations that are popular. The accuracy of the baseline model that always predicts non-popular (or unpopular videos) is 0.6288 or about 63%.

```
table(trainSparse$popular_video)
```

In the testing set, there are 7,724 observations that are unpopular and 4,560 observations that are popular. The accuracy of the baseline model that always predicts non-popular (or unpopular videos) is 0.6287 or about 63%.

```
table(testSparse$popular_video)
```

The logistic regression model

A logistic regression model can be used to predict the categorical outcome (whether a video was popular or not). The logistic regression model computes probabilities that can be used to assess the confidence of the prediction.

```
USvideosLOG <- glm(popular_video ~ ., data = trainSparse, family = binomial)
```

The AIC, or the measure of the quality of the logistic model that accounts for the number of variables used compared to the number of observations is 27,922.

To get the probabilities of the predictions:

```
predictLOG <- predict(USvideosLOG, type= ”response”)
```

To see if the model is predicting higher probabilities for the actual popular videos as expected:

```
tapply(predictLOG, trainSparse$popular_video, mean)
```

For all of the TRUE popular videos, the model predicts an average probability of about 0.59. For all of the FALSE popular videos (meaning unpopular) the model predicts an average probability of about 0.24. This is a good sign because it looks like the model is predicting a higher probability for the actual popular videos.

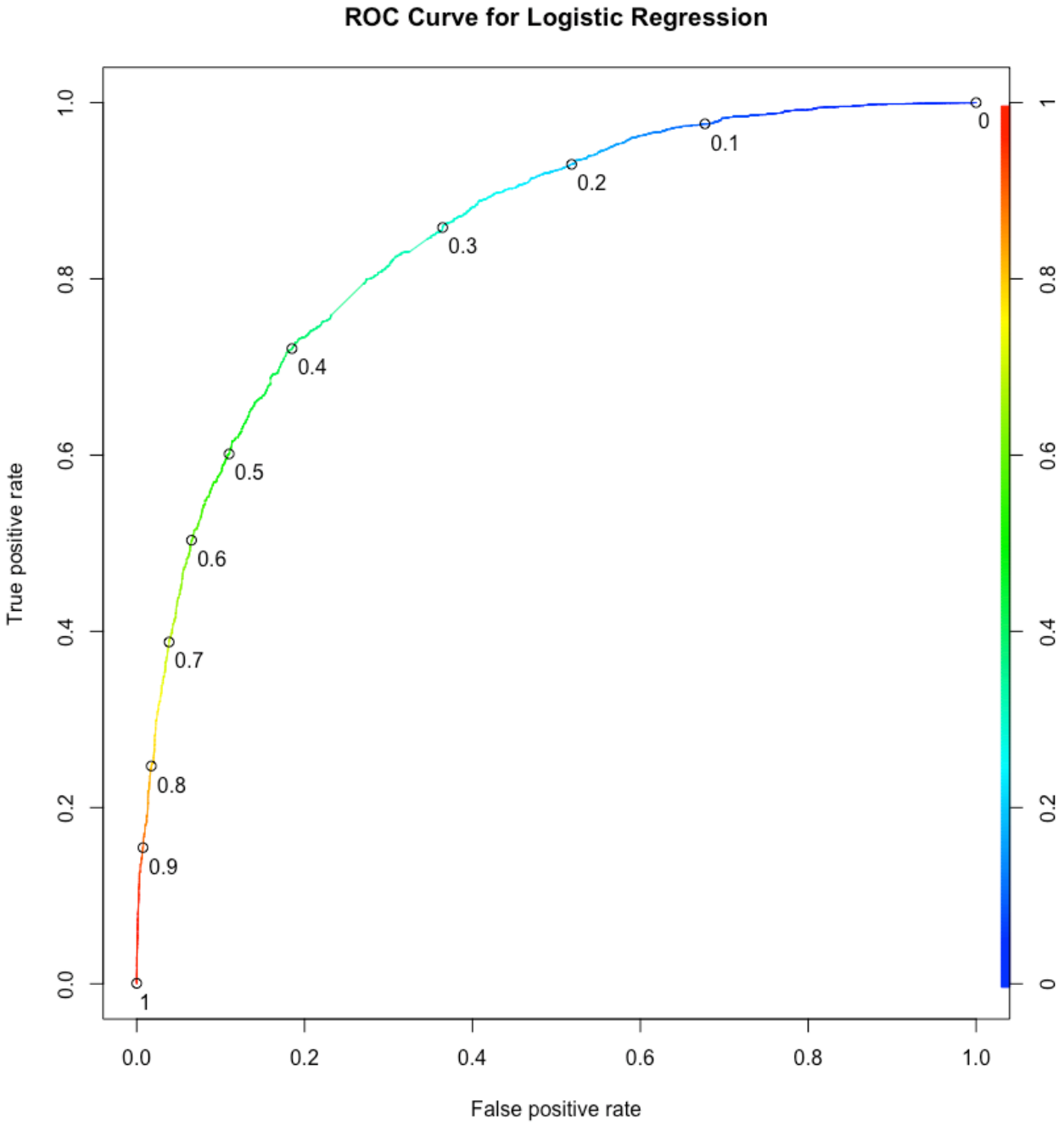
A threshold of 0.5 was initially selected and applied.

```
table(trainSparse$popular_video, predictLOG > 0.5)
```

Results show 0.6014 as the Sensitivity, or the True Positive Rate, which is the percentage of actual popular videos that were classified correctly and 0.8900 as the Specificity, or the True Negative Rate, which is the percentage of actual unpopular videos that were classified correctly. Because this project is more concerned with having a high Sensitivity, or high True Positive Rate, a new threshold was selected and applied to minimize the False Positive Rate and maximize the True Positive Rate. This was done because the goal of this project’s prediction model is to discover which *tags* are associated with popular videos, so being able to more accurately determine which videos are popular is a higher priority.

To help decide which threshold to use, a ROC curve was constructed and then plotted.

```
predLOG <- prediction(predictLOG, trainSparse$popular_video)

prefLOG <- performance(predLOG, “tpr”, “fpr”)

plot(prefLOG, main = “ROC Curve for Logistic Regression”, colorize = TRUE, print.cutoffs.at =
seq(0 ,1, 0.1), text.adj = c(-0.2, 1.7))
```



Based on the ROC curve, a threshold of 0.3 would be best in order to maximize the True Positive Rate while minimizing the False Positive Rate. Using this threshold will increase the Sensitivity or the percentage of actual popular videos the model classified correctly.

After the new threshold of 0.3 is applied:

```
table(trainSparse$popular_video, predictLOG > 0.3)
```

The Sensitivity went up to 0.8583 (compared to 0.6014) while the Specificity went down to 0.6356 (compared to 0.8900).

To get the AUC value on the training set:

```
as.numeric(performance(predLOG, "auc")@y.values)
```

The AUC value on the training set, or area under the curve, is 0.8497.

Next, the logistic regression model was used to make predictions on the testing set.

```
predictLOG_test <- predict(USvideosLOG, type = "response", newdata = testSparse)
```

Using the table function, a confusion matrix was created with a threshold of 0.3, which is the optimized threshold based on the training data.

```
table(testSparse$popular_video, predictLOG_test > 0.3)
```

The accuracy of this model is 0.7095 or about 71%, which is better than the baseline model of 63%.

To get the AUC value on the testing set:

```
predLOG_ROCR <- prediction(predictLOG_test, testSparse$popular_video)

as.numeric(performance(predLOG_ROCR, "auc")@y.values)
```

This gives an AUC value of 0.8373 or about 84% on the testing set, which means that the logistic regression model can differentiate between popular videos and unpopular videos pretty well.

The CART model

Besides a logistic regression model, CART (Classification and Regression Trees) can be used to predict a categorical outcome (in this case *popularity*). An advantage of the CART model is that it is very interpretable. This model produces a CART tree, which is a series of decision rules presented in a way that can easily be explained.

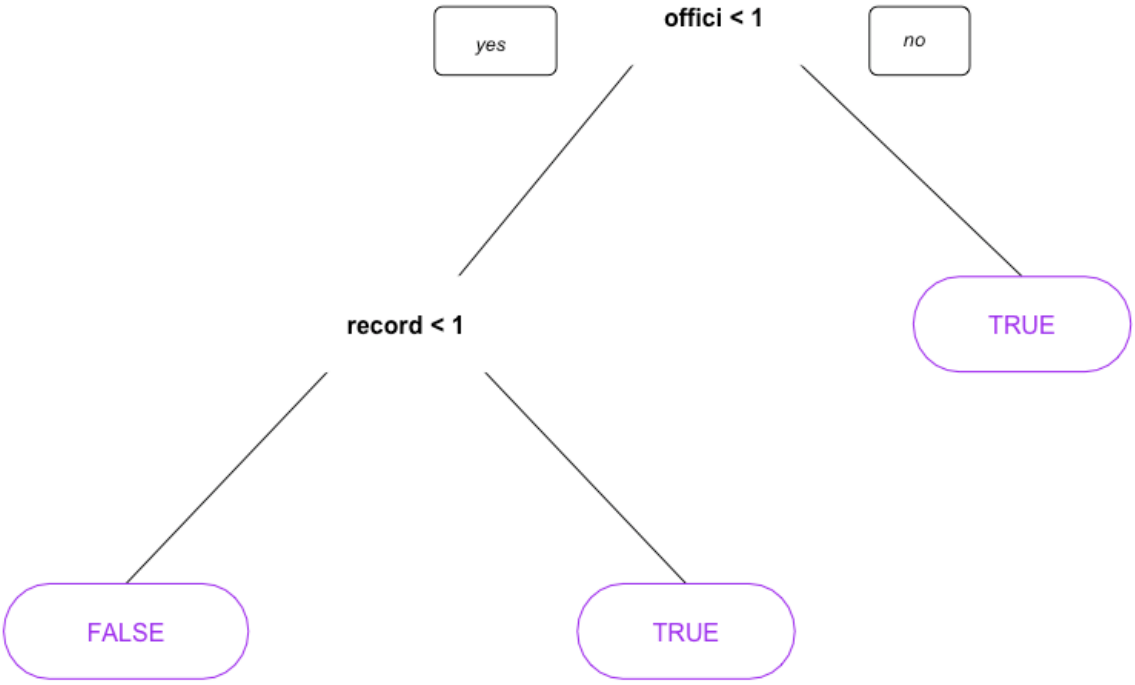
First, the CART model was built:

```
USvideosCART <- rpart(popular_video ~ ., data = trainSparse, method = "class")
```

Then the CART model was plotted:

```
prp(USvideosCART, main = "CART Tree", col = "purple")
```

CART Tree



According to the CART tree, if the *tag* "offici" is in the video *tags*, then predict TRUE or predict that the video is popular. If the *tag* "offici" is not in the video *tags*, but the *tag* "record" is, then again predict TRUE or predict that the video is popular. If the *tags* "offici" and "record" are not in the video *tags*, then predict FALSE or predict that the video is not popular. This intuitively makes sense because the top two video categories in this data set are *Entertainment* (for 9,964 videos) and *Music* (for 6,472 videos), which would include these two *tags*.

To evaluate the performance of the CART model, predictions were made on the training and testing sets:

```
predictCARTTrain <- predict(USvideosCART, newdata = trainSparse, type = "class")
```

```
predictCART <- predict(USvideosCART, newdata = testSparse, type = "class")
```

Confusion matrices were then built to compute the accuracy of the CART model on the training and testing sets:

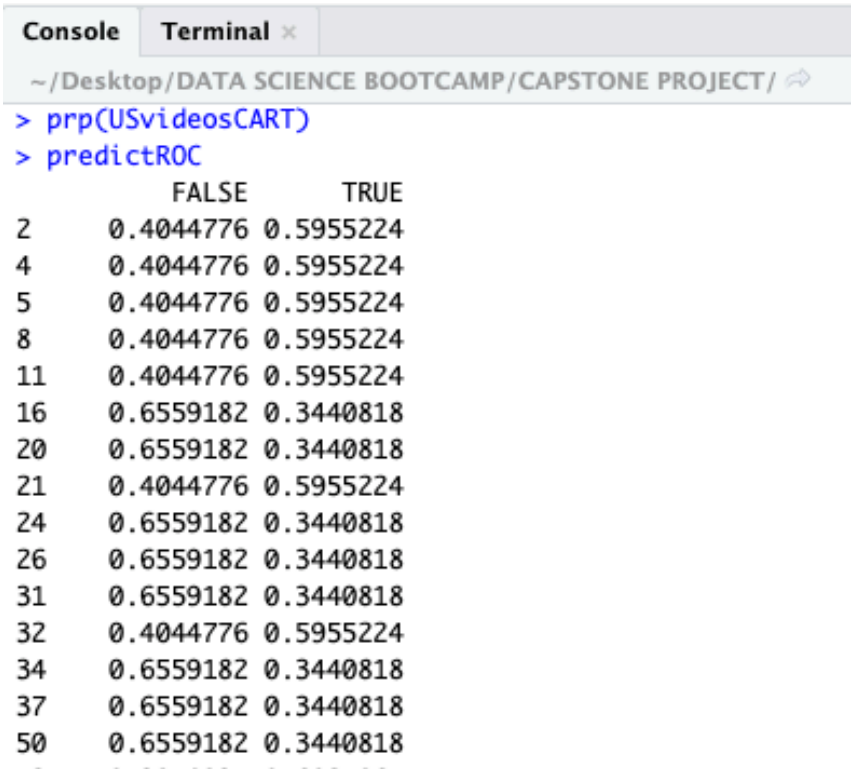
table(trainSparse$popular_video, predictCARTTrain)

table(testSparse$popular_video, predictCART)

The accuracy of the CART model on the training set is 0.6499 or about 65% and the accuracy of the CART model on the testing set is 0.6548 or about 65%.

Next, the ROC curve was generated for the CART model:

predictROC <- predict(USvideosCART, newdata = testSparse)



The output displays two columns for each of the testing set observations, labeled FALSE and TRUE. The first column (FALSE) is the percentage of training set data in the same subset as that testing set observation that had outcome FALSE. The second column (TRUE) is the percentage of training set data in the same subset as that testing set observation that had outcome TRUE. The second column (TRUE) can be interpreted as the probability that that testing set observation has outcome TRUE. This second column (TRUE) is used to pick the threshold value when evaluating the CART model.

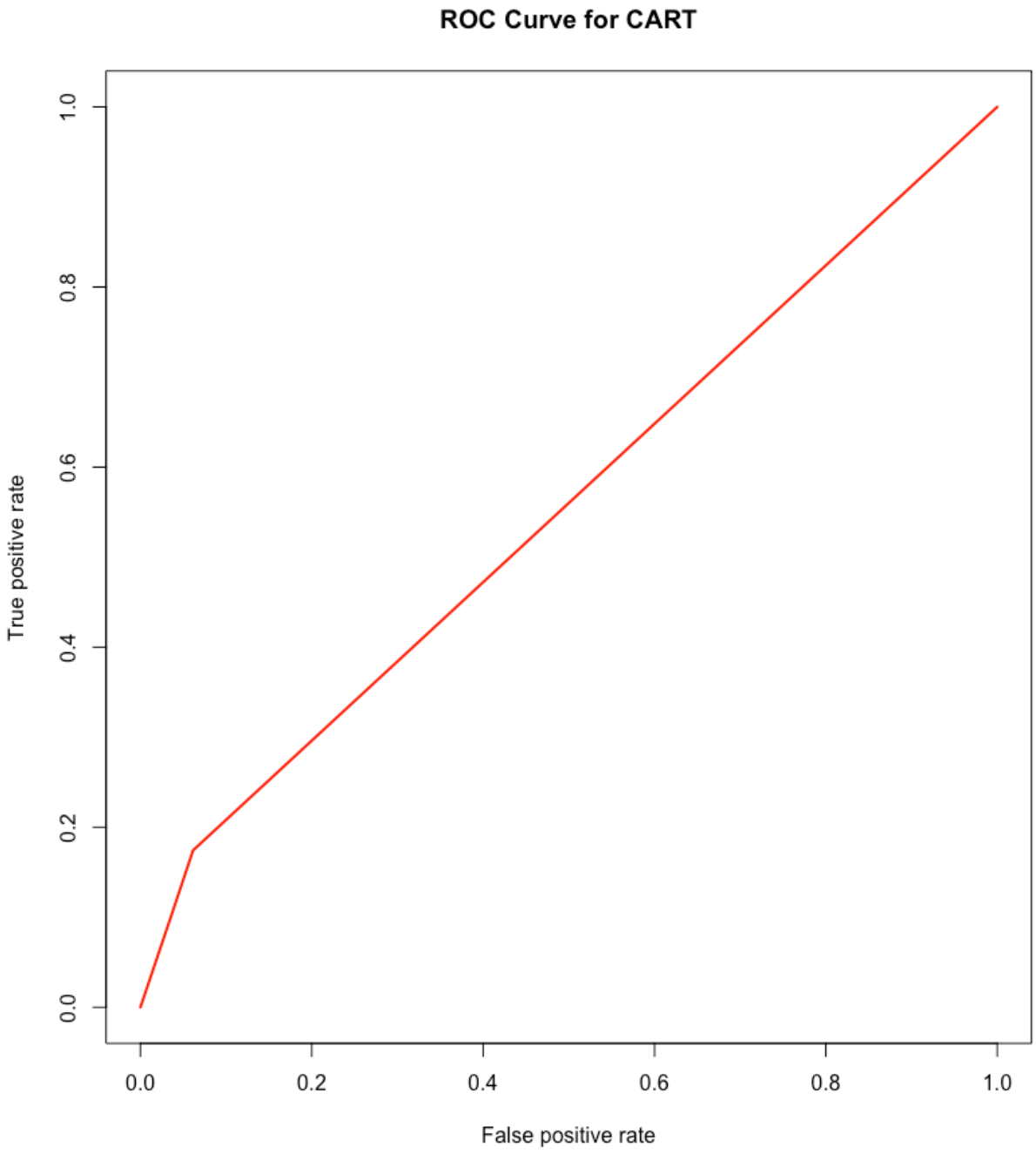
The ROC curve is generated for the CART model just as was done for logistic regression:

pred <- prediction(predictROC[, 2], testSparse$popular_video)

perf <- performance(pred, “tpr”, “fpr”)

Then the ROC curve was plotted:

plot(perf, main = “ROC Curve for CART”, col = 2, lwd =2)



To get the AUC value on the testing set:

as.numeric(performance(pred, “auc”)@y.values)

This gives an AUC value of 0.5564 or about 56% on the testing set, which means that the CART model doesn’t differentiate between popular videos and unpopular videos too well.

To improve upon the accuracy of the CART model, it was necessary to make sure the proper cp was selected. When cross-validation is used in R, a parameter value called cp, known as complexity parameter, is employed. For this project, the k-fold cross-validation method was applied to properly select the parameter value. This method averages the accuracy of the model over the k-folds to determine the final parameter value to use. The cp measures the trade-off between model complexity and accuracy on the training set. If the parameter value is too small, then the accuracy is lower because the model is probably overfit to the training set. If the parameter value is too large, the accuracy is also lower because the model is too simple.

Therefore, cross-validation was used to select the cp value for the CART tree by way of the k-folds method.

First, the number of folds were defined. For this project, 10 folds were applied.

```
fitControl <- trainControl(method = "cv", number = 10)
```

Second, the possible values for the cp were selected.

```
cartGrid <- expand.grid(.cp = (1:50)*0.01)
```

Then cross-validation was performed to validate the parameters for the CART tree.

```
train(popular_video ~ ., data = trainSparse, method = "rpart", trControl = fitControl, tuneGrid = cartGrid)
```

The output showed the accuracy for different values of cp. The final value used for the model was cp = 0.02. A new CART model was created using this value of cp found through cross-validation.

```
USvideosCV <- rpart(popular_video ~ ., data = trainSparse, method = "class", control = rpart.control(cp = 0.02))
```

Next, predictions were made using this new model:

```
predictCV <- predict(USvideosCV, newdata = testSparse, type = "class")
```

A confusion matrix was then built to compute the accuracy of the new CART model on the testing set:

```
table(testSparse$popular_video, predictCV)
```

The accuracy of the cross-validated CART model is 0.6548 or about 65%, which is the same as the accuracy for the original CART model. Although cross-validation is used to ensure that a good parameter value is selected, which will often significantly increase the model's accuracy, in this case this did not hold true. The accuracy for the cross-validated CART model did not increase. Therefore, this k-folds cross-validation method is confirming that a smart parameter value has already been applied to the original CART model.

The CART model has shown improvement over the baseline model (65% accuracy for the CART model compared to 63% accuracy for the baseline model). However, the CART model did not

perform better than the logistic regression model (65% accuracy for the CART model compared to 71% accuracy for the logistic regression model). This may be due to a limitation of the CART model in that these models do not perform well on smaller data sets. Although one may argue that this data set is not particularly small, the sparsity of the data did not help.

The random forest model

The random forest model is similar to CART and was designed to improve the prediction accuracy of CART. This model works by building a large number of CART trees.

```
USvideosRF <- randomForest(popular_video ~ ., data = trainSparse)
```

With the random forest model built, predictions were made on the training set.

```
predictRFTrain <- predict(USvideosRF, newdata = trainSparse)
```

A confusion matrix was then built to compute the accuracy of the random forest model on the training data:

```
table(trainSparse$popular_video, predictRFTrain)
```

Using the training set, the random forest model has an accuracy of 0.9189 or about 92%.

Next, predictions were made on the testing set:

```
predictRF <- predict(USvideosRF, newdata = testSparse)
```

A confusion matrix was then built to compute the accuracy of the random forest model on the testing data:

```
table(testSparse$popular_video, predictRF)
```

Using the testing set, the random forest model has an accuracy of 0.9064 or about 90.6%. This is significantly better than the CART model, which only had an accuracy of 65% and the baseline model, which only had an accuracy of 63%. The random forest model also showed improvement over the logistic regression model, which had an accuracy of 71%.

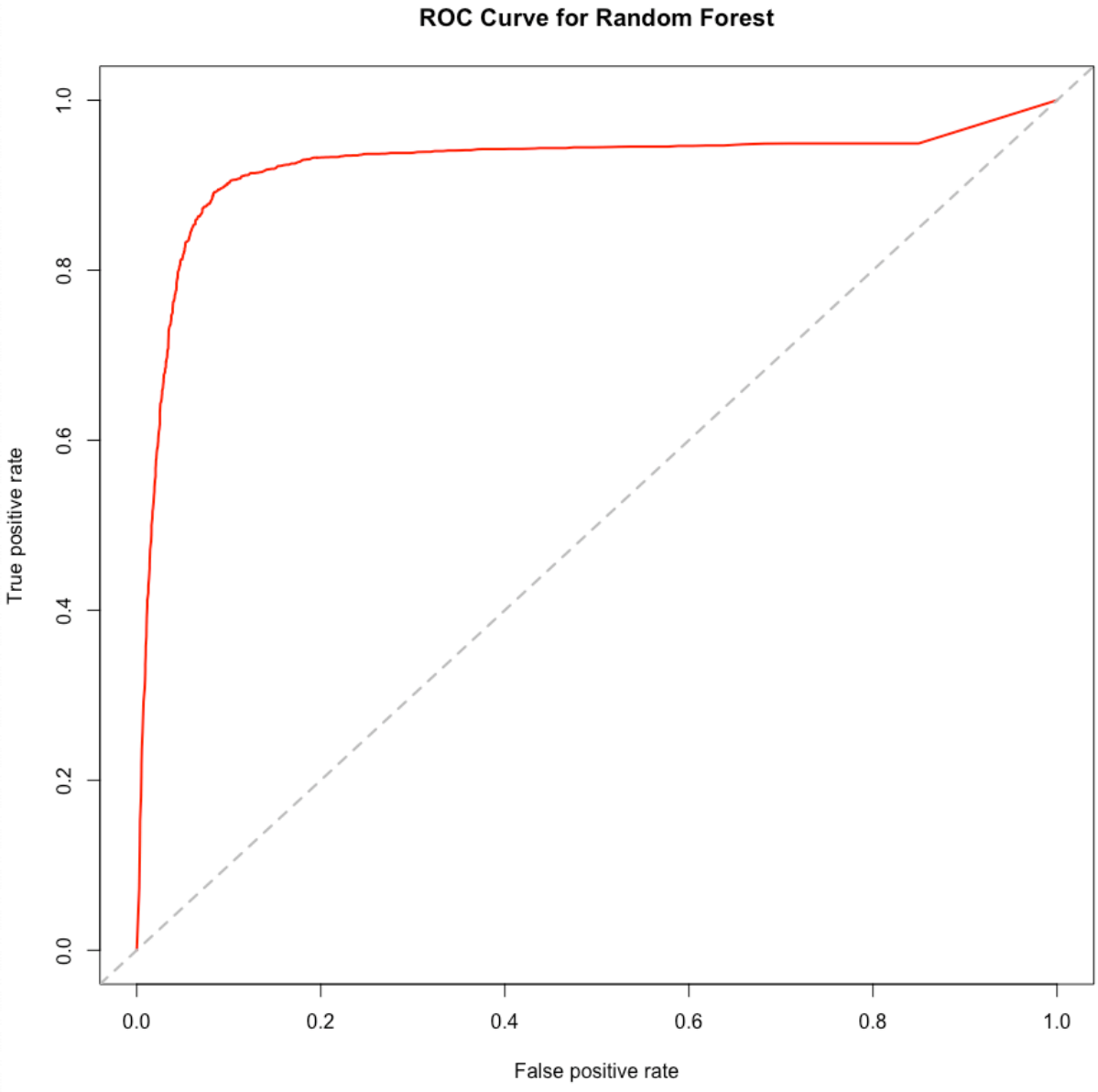
The ROC curve is generated for the random forest model:

```
predictROC_RF <- predict(USvideosRF, type = “prob”, newdata = testSparse)[, 2]

predRF <- prediction(predictROC_RF, testSparse$popular_video)

perfRF <- performance(predRF, “tpr”, “fpr”)
```

Then the ROC curve was plotted:

plot(perfRF, main = “ROC Curve for Random Forest”, col = 2, lwd = 2)

abline(a = 0, b = 1, lwd = 2, lty = 2, col = “gray”)



To interpret this ROC curve, recall that perfect classification happens at 100% True Positive Rate and 0% False Positive Rate. Thus, perfect classification happens at the upper left-hand corner of the graph. The closer the red line comes to that corner, the better the model is at classification. The gray diagonal dashed line represents random guess. The distance of the red line over the gray line represents how much better the random forest model is doing at making predictions than random guess. As shown by the ROC curve in the graph above, the random forest model is significantly better than random guess.

To get the AUC value on the testing set:

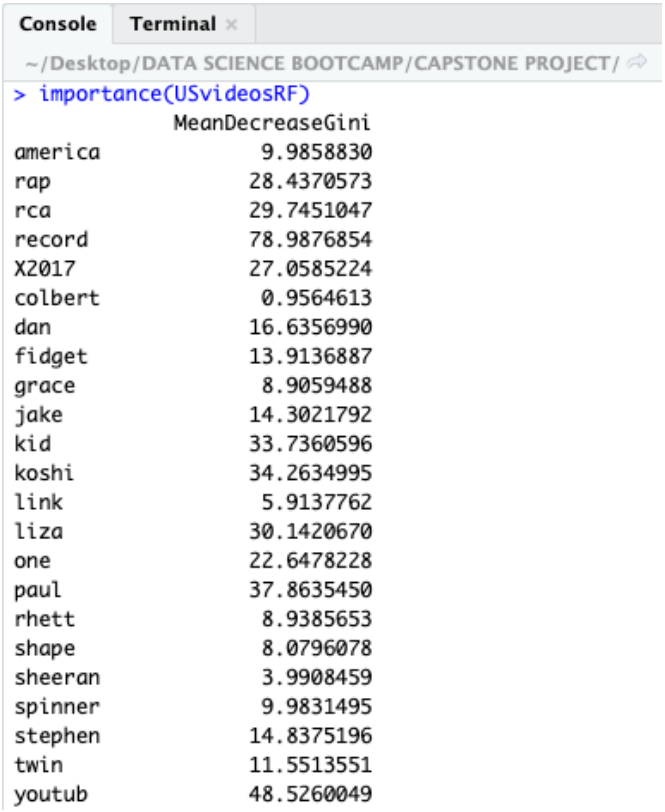
```
as.numeric(performance(predRF, "auc")@y.values)
```

This gives an AUC value of 0.9234 or about 92% on the testing set, which means that the random forest model differentiates between popular videos and unpopular videos very well.

To see which *tags* are important in classifying *popularity* based on the random forest model, two functions were used.

The first function gives a textual representation of how important the *tag* variables are in classifying *popularity*:

```
importance(USvideosRF)
```

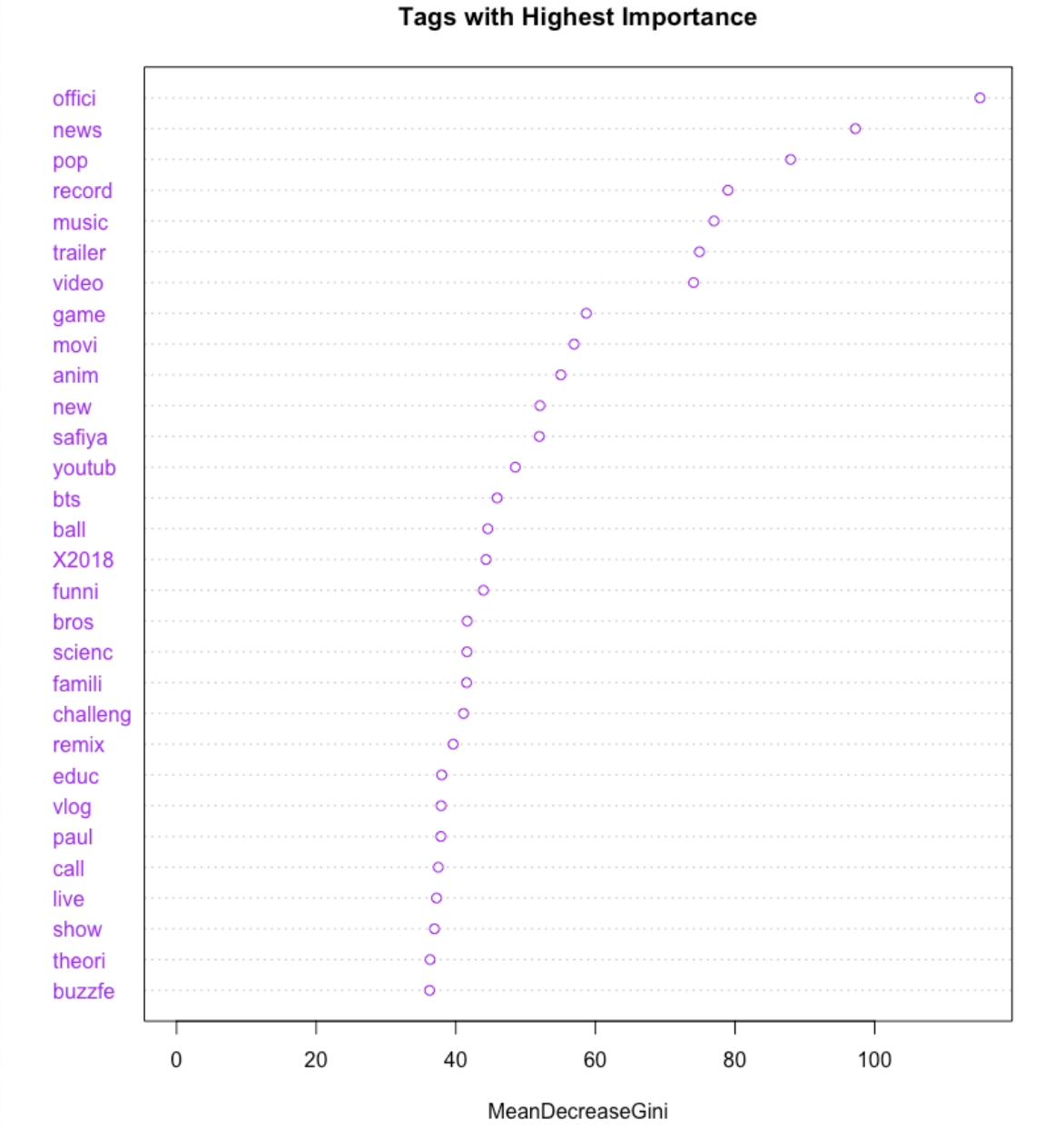


```
Console  Terminal
~/Desktop/DATA SCIENCE BOOTCAMP/CAPSTONE PROJECT/
> importance(USvideosRF)
            MeanDecreaseGini
america            9.9858830
rap               28.4370573
rca               29.7451047
record            78.9876854
X2017             27.0585224
colbert            0.9564613
dan               16.6356990
fidget            13.9136887
grace              8.9059488
jake              14.3021792
kid               33.7360596
koshi             34.2634995
link               5.9137762
liza              30.1420670
one               22.6478228
paul              37.8635450
rhett              8.9385653
shape              8.0796078
sheeran            3.9908459
spinner            9.9831495
stephen           14.8375196
twin              11.5513551
youtub            48.5260049
```

It should be explained here that Mean Decrease in Gini is the average or mean of a variable's total decrease in node impurity, weighted by the proportion of samples reaching that node in each individual decision tree in the random forest. This means that a *higher Mean Decrease in Gini score* indicates *higher variable importance*. Therefore, *tags* with a higher Mean Decrease in Gini number are more likely to be associated with higher ranking popular videos.

The second function gives a graphical representation of how important the *tag* variables are in classifying *popularity*:

```
varImpPlot(USvideosRF, main = "Tags with Highest Importance", col = "purple")
```



Based on the Variable Importance Plot, the most important *tags* included: "offici", "news", "pop", "record", "music", "trailer", and "video". This intuitively makes sense given that the CART tree also showed "offici" and "record" as *tags* predicting *popularity* and that the majority of these *tags* are associated with the top two video categories of the data set, *Entertainment* (with 9,964 videos) and *Music* (with 6,472 videos).

Using the threshold and parameters established in the training set, which showed significant accuracy in the testing set as well, the final random forest model will be retrained to make predictions on all of the data.

```
USvideosRF <- randomForest(popular_video ~ ., data = tagsSparse)

predictRFFM <- predict(USvideosRF, newdata = tagsSparse)
```

A confusion matrix is created to compute the accuracy of the final model on all of the data:

```
table(tagsSparse$popular_video, predictRFFM)
```

After running the final random forest model on all of the data, the model now has an accuracy of 0.9151 or about 91.5%. This percentage falls nicely between the two training (accuracy of 92%) and testing (accuracy of 90.6%) data sets.

Findings

Of the machine learning techniques applied to this project, it turns out that the random forest model is the most accurate of all the models tested to determine a video's *popularity* based on the *tags* used. In addition to having an accuracy of 91.5%, the model has an AUC of 92%, which means that the model can differentiate between popular and unpopular videos very well. Thus, it's fair to say that this random forest model can reasonably pick *popularity* given our data set. However, it should be noted that the random forest algorithm took over 10 hours to run. Depending on the data scientist's timeline, the logistic regression model may be the best practical choice.