Now that we've prepared our data set, let's use CART to build a predictive model.

First, we need to load the necessary packages in our R Console by typing library(rpart), and then library(rpart.plot).

Now let's build our model.

We'll call it tweetCART, and we'll use the rpart function to predict Negative using all of the other variables as our independent variables and the data set trainSparse.

We'll add one more argument here, which is method = "class" so that the rpart function knows to build a classification model.

We're just using the default parameter settings so we won't add anything for minbucket or cp.

Now let's plot the tree using the prp function.

Our tree says that if the word "freak" is in the tweet, then predict TRUE, or negative sentiment.

If the word "freak" is not in the tweet, but the word "hate" is, again predict TRUE.

If neither of these two words are in the tweet, but the word "wtf" is, also predict TRUE, or negative sentiment.

If none of these three words are in the tweet, then predict FALSE, or non-negative sentiment.

This tree makes sense intuitively since these three words are generally seen as negative words.

Now, let's go back to our R Console and evaluate the numerical performance of our model by making predictions on the test set.

We'll call our predictions predictCART.

And we'll use the predict function to predict using our model tweetCART on the new data set testSparse.

We'll add one more argument, which is type = "class" to make sure we get class predictions.

Now let's make our confusion matrix using the table function.

We'll give as the first argument the actual outcomes, testSparse$Negative, and then as the second argument, our predictions, predictCART.

To compute the accuracy of our model, we add up the numbers on the diagonal, 294 plus 18-- these are the observations we predicted correctly-- and divide by the total number of observations in the table, or the total number of observations in our test set.

So the accuracy of our CART model is about 0.88.

Let's compare this to a simple baseline model that always predicts non-negative.

To compute the accuracy of the baseline model, let's make a table of just the outcome variable Negative.

So we'll type table, and then in parentheses, testSparse$Negative.

This tells us that in our test set we have 300 observations with non-negative sentiment and 55 observations with negative sentiment.

So the accuracy of a baseline model that always predicts non-negative would be 300 divided by 355, or 0.845.

So our CART model does better than the simple baseline model.

How about a random forest model?

How well would that do?

Let's first load the random forest package with library(randomForest), and then we'll set the seed to 123 so that we can replicate our model if we want to.

Keep in mind that even if you set the seed to 123, you might get a different random forest model than me depending on your operating system.

Now, let's create our model.

We'll call it tweetRF and use the randomForest function to predict Negative again using all of our other variables as independent variables and the data set trainSparse.

We'll again use the default parameter settings.

The random forest model takes significantly longer to build than the CART model.

We've seen this before when building CART and random forest models, but in this case, the difference is particularly drastic.

This is because we have so many independent variables, about 300 different words.

So far in this course, we haven't seen data sets with this many independent variables.

So keep in mind that for text analytics problems, building a random forest model will take significantly longer than building a CART model.

So now that our model's finished, let's make predictions on our test set.

We'll call them predictRF, and again, we'll use the predict function to make predictions using the model tweetRF this time, and again, the new data set testSparse.

Now let's make our confusion matrix using the table function, first giving the actual outcomes, testSparse$Negative, and then giving our predictions, predictRF.

To compute the accuracy of the random forest model, we again sum up the cases we got right, 293 plus 21, and divide by the total number of observations in the table.

So our random forest model has an accuracy of 0.885.

This is a little better than our CART model, but due to the interpretability of our CART model, I'd probably prefer it over the random forest model.

If you were to use cross-validation to pick the cp parameter for the CART model, the accuracy would increase to about the same as the random forest model.

So by using a bag-of-words approach and these models, we can reasonably predict sentiment even with a relatively small data set of tweets.