In this video, we'll build a CART model to predict healthcare cost.

First, let's make sure the packages rpart and rpart.plot are loaded with the library function.

You should have already installed them in the previous lecture on predicting Supreme Court decisions.

Now, let's build our CART model.

We'll call it ClaimsTree.

And we'll use the rpart function to predict bucket2009, using as independent variables: age, arthritis, alzheimers, cancer, copd, depression, diabetes, heart.failure, ihd, kidney, osteoporosis, and stroke.

We'll also use bucket2008 and reimbursement2008.

The data set we'll use to build our model is ClaimsTrain.

And then we'll add the arguments, method = "class", since we have a classification problem here, and cp = 0.00005.

Note that even though we have a multi-class classification problem here, we build our tree in the same way as a binary classification problem.

So go ahead and hit Enter.

The cp value we're using here was selected through cross-validation on the training set.

We won't perform the cross-validation here, because it takes a significant amount of time on a data set of this size.

Remember that we have almost 275,000 observations in our training set.

But keep in mind that the R commands needed for cross-validation here are the same as those used in the previous lecture on predicting Supreme Court decisions.

So now that our model's done, let's take a look at our tree with the prp function.

It might take a while to load, because we have a huge tree here.

This makes sense for a few reasons.

One is the large number of observations in our training set.

Another is that we have a five-class classification problem, so the classification is more complex than a binary classification case, like the one we saw in the previous lecture.

The trees used by D2Hawkeye were also very large CART trees.

While this hurts the interpretability of the model, it's still possible to describe each of the buckets of the tree according to the splits.

So now, let's make predictions on the test set.

So go back to your R console, and we'll call our predictions PredictTest, where we'll use the predict function for our model ClaimsTree, and our newdata is ClaimsTest.

And we want to add type = "class" to get class predictions.

And we can make our classification matrix on the test set to compute the accuracy.

So we'll use the table function, where the actual outcomes are ClaimsTest$bucket2009, and our predictions are PredictTest.

So to compute the accuracy, we need to add up the numbers on the diagonal and divide by the total number of observations in our test set.

So we have 114141 + 16102 + 118 + 201 + 0.

And we'll divide by the number of rows in ClaimsTest.

So the accuracy of our model is 0.713.

For the penalty error, we can use our penalty matrix like we did in the previous video.

So scroll up to the classification matrix command and surround the table function by the as.matrix function, and then we'll multiply by PenaltyMatrix.

So remember that this takes each entry in our classification matrix and multiplies it by the corresponding number in the penalty matrix.

So now we just need to add up all of the numbers in this matrix by surrounding it by the sum function and then dividing by the total number of observations in our test set, or nrow(ClaimsTest).

So our penalty error is 0.758.

In the previous video, we saw that our baseline method had an accuracy of 68% and a penalty error of 0.74.

So while we increased the accuracy, the penalty error also went up.

Why?

By default, rpart will try to maximize the overall accuracy, and every type of error is seen as having a penalty of one.

Our CART model predicts 3, 4, and 5 so rarely because there are very few observations in these classes.

So we don't really expect this model to do better on the penalty error than the baseline method.

So how can we fix this?

The rpart function allows us to specify a parameter called loss.

This is the penalty matrix we want to use when building our model.

So let's scroll back up to where we built our CART model.

At the end of the rpart function, we'll add the argument params = list(loss=PenaltyMatrix).

This is the name of the penalty matrix we created.

Close the parentheses and hit Enter.

So while our model is being built, let's think about what we expect to happen.

If the rpart function knows that we'll be giving a higher penalty to some types of errors over others, it might choose different splits when building the model to minimize the worst types of errors.

We'll probably get a lower overall accuracy with this new model.

But hopefully, the penalty error will be much lower too.

So now that our model is done, let's regenerate our test set predictions by scrolling up to where we created PredictTest and hitting Enter, and then recreating our classification matrix by scrolling up to the table function and hitting Enter again.

Now let's add up the numbers on the diagonal, 94310 + 18942 + 4692 + 636 + 2, and divide by the number of rows in ClaimsTest.

And hit Enter.

So the accuracy of this model is 0.647.

And we can scroll up and compute the penalty error here by going back to the sum command and hitting Enter.

So the penalty error of our new model is 0.642.

Our accuracy is now lower than the baseline method, but our penalty error is also much lower.

Note that we have significantly fewer independent variables than D2Hawkeye had.

If we had the hundreds of codes and risk factors available to D2Hawkeye, we would hopefully do even better.

In the next video, we'll discuss the accuracy of the models used by D2Hawkeye and how analytics can provide an edge.