STATE DATA REVISITED

We will be revisiting the "state" dataset from Week 2. Recall that this dataset has, for each of the fifty U.S. states, the population, per capita income, illiteracy rate, murder rate, high school graduation rate, average number of frost days, area, latitude and longitude, division the state belongs to, region the state belongs to, and two-letter abbreviation. This dataset comes from the U.S. Department of Commerce, Bureau of the Census.

Load the dataset into R and convert it to a data frame by running the following two commands in R:

```
data(state)
statedata = data.frame(state.x77)
```

Inspect the data set using the command:

```
str(statedata)
```

We will try to build a model for life expectancy using regression trees, and employ cross-validation to improve our tree's performance.

PROBLEM 1.1 - LINEAR REGRESSION MODELS (1/1 point)

Let's recreate the **linear regression** models we made in the previous homework question. First, predict *Life.Exp* using all of the other variables as the independent variables (*Population, Income, Illiteracy, Murder, HS.Grad, Frost, Area*). Use the entire dataset to build the model.

What is the **adjusted** R-squared of the model?

0.6922

0.6922

Answer: 0.6922

EXPLANATION

To build the regression model, type the following command in your R console:

RegModel = lm(Life.Exp ~ ., data=statedata)

Then, if you look at the output of summary(RegModel), you should see that the Adjusted R-squared is 0.6922.

Hide Answer *You have used 1 of 3 submissions*

PROBLEM 1.2 - LINEAR REGRESSION MODELS (1/1 point)

Calculate the sum of squared errors (SSE) between the predicted life expectancies using this model and the actual life expectancies:

23.29714

23.29714

Answer: 23.3

EXPLANATION

To make predictions, type in your R console:

Predictions = predict(RegModel)

where "RegModel" is the name of your regression model. You can then compute the sum of squared errors by typing the following in your R console:

sum((statedata$Life.Exp - Predictions)^2)

The SSE is 23.29714.

Alternatively, you can use the following command to get the SSE:

SSE = sum(LinReg1$residuals^2)

Hide Answer *You have used 1 of 3 submissions*

PROBLEM 1.3 - LINEAR REGRESSION MODELS (1/1 point)

Build a second **linear regression** model using just *Population, Murder, Frost, and HS.Grad* as independent variables (the best 4 variable model from the previous homework). What is the **adjusted** R-squared for this model?

0.7126

0.7126

Answer: 0.71

EXPLANATION

You can create this regression model by typing the following into your R console:

RegModel2 = lm(Life.Exp ~ Population + Murder + Frost + HS.Grad, data=statedata)

Then, if you type:

summary(RegModel2)

The Adjusted R-squared is at the bottom right of the output, and is 0.7126

Hide Answer *You have used 2 of 3 submissions*

PROBLEM 1.4 - LINEAR REGRESSION MODELS (1/1 point)

Calculate the sum of squared errors again, using this reduced model:

23.30804

23.30804

Answer: 23.3

EXPLANATION

The sum of squared errors (SSE) can be computed by first making predictions:

Predictions2 = predict(RegModel2)

and then computing the sum of the squared difference between the actual values and the predictions:

sum((statedata$Life.Exp - Predictions2)^2).

Alternatively, you can compute the SSE with the following command:

SSE = sum(RegModel2$residuals^2)

Hide Answer *You have used 1 of 3 submissions*

PROBLEM 1.5 - LINEAR REGRESSION MODELS (1/1 point)

Which of the following is correct?

- (selected) Trying different combinations of variables in linear regression is like trying different numbers of splits in a tree - this controls the complexity of the model. ✔
- Using many variables in a linear regression is **always** better than using just a few.
- The variables we removed were uncorrelated with *Life.Exp*

EXPLANATION

The correct answer is the first one. Trying different combinations of variables in linear regression controls the complexity of the model. This is similar to trying different numbers of splits in a tree, which is also controlling the complexity of the model.

The second answer is incorrect because as we see here, a model with fewer variables actually has a higher adjusted R-squared. If your accuracy is just as good, a model with fewer variables is almost always better.

The third answer is incorrect because the variables we removed have non-zero correlations with the dependent variable Life.Exp. Illiteracy and Area are negatively correlated with Life.Exp, with correlations of -0.59 and -0.11. Income is positively correlated with Life.Exp, with a correlation of 0.34. These correlations can be computed by typing the following into your R console:

cor(statedata$Life.Exp, statedata$Income)

cor(statedata$Life.Exp, statedata$Illiteracy)

cor(statedata$Life.Exp, statedata$Area)

Hide Answer *You have used 1 of 1 submissions*

PROBLEM 2.1 - CART MODELS (1/1 point)

Let's now build a **CART model** to predict *Life.Exp* using all of the other variables as independent variables (*Population, Income, Illiteracy, Murder, HS.Grad, Frost, Area*). We'll use the default *minbucket* parameter, so don't add the *minbucket* argument. Remember that in this problem we are not as interested in *predicting* life expectancies for new observations as we are understanding how they relate to the other variables we have, so we'll use all of the data to build our model. You shouldn't use the method="class" argument since this is a regression tree.

Plot the tree. Which of these variables appear in the tree?

- [ ] Population
- [x] Murder ✔
- [ ] Frost
- [ ] HS.Grad
- [ ] Area

EXPLANATION

You can create the tree in R by typing the following command:

CARTmodel = rpart(Life.Exp ~ ., data=statedata)

Be sure to load the "rpart" and "rpart.plot" packages with the library command if they are not already loaded.

You can then plot the tree by typing:

prp(CARTmodel)

We can see that the only variable used in the tree is "Murder".

Hide Answer *You have used 1 of 3 submissions*

PROBLEM 2.2 - CART MODELS (1/1 point)

Use the regression tree you just built to predict life expectancies (using the predict function), and calculate the sum-of-squared-errors (SSE) like you did for linear regression. What is the SSE?

28.99848

28.99848

Answer: 29.0

EXPLANATION

You can make predictions using the CART model by typing the following line in your R console (assuming your model is called "CARTmodel"):

PredictionsCART = predict(CARTmodel)

Then, you can compute the sum of squared errors (SSE) by typing the following:

sum((statedata$Life.Exp - PredictionsCART)^2)

The SSE is 28.99848.

Hide Answer *You have used 1 of 3 submissions*

PROBLEM 2.3 - CART MODELS (1/1 point)

The error is higher than for the linear regression models. One reason might be that we haven't made the tree big enough. Set the *minbucket* parameter to 5, and recreate the tree.

Which variables appear in this new tree?

- [ ] Population
- [x] Murder ✔
- [ ] Frost
- [x] HS.Grad ✔
- [x] Area ✔

EXPLANATION

You can create a tree with a minbucket value of 5 with one of the following two commands:

CARTmodel2 = rpart(Life.Exp ~ ., data=statedata, minbucket=5)

CARTmodel2 = rpart(Life.Exp ~ ., data=statedata, control=rpart.control(minbucket=5))

Then, if you plot the tree using prp(CARTmodel2), you can see that Murder, HS.Grad, and Area are all used in this new tree.

Hide Answer *You have used 1 of 3 submissions*

PROBLEM 2.4 - CART MODELS (1/1 point)

Do you think the default minbucket parameter is smaller or larger than 5 based on the tree that was built?

- ○ Smaller
- ◉ Larger ✔

EXPLANATION

Since the tree now has more splits, it must be true that the default minbucket parameter was limiting the tree from splitting more before. So the default minbucket parameter must be larger than 5.

Hide Answer *You have used 1 of 1 submissions*

PROBLEM 2.5 - CART MODELS (1/1 point)

What is the SSE of this tree?

23.64283

23.64283

Answer: 23.6

EXPLANATION

You can compute the SSE of this tree by first making predictions:

PredictionsCART2 = predict(CARTmodel2)

and then computing the sum of the squared differences between the actual values and the predicted values:

sum((statedata$Life.Exp - PredictionsCART2)^2)

The SSE is 23.64283

This is much closer to the linear regression model's error. By changing the parameters we have improved the fit of our model.

Hide Answer *You have used 1 of 3 submissions*

PROBLEM 2.6 - CART MODELS (1/1 point)

Can we do even better? Create a tree that predicts *Life.Exp* using **only** *Area*, with the *minbucket* parameter to 1. What is the SSE of this newest tree?

9.312442

9.312442

Answer: 9.3

EXPLANATION

You can create this third tree by typing:

CARTmodel3 = rpart(Life.Exp ~ Area, data=statedata, minbucket=1)

Then to compute the SSE, first make predictions:

PredictionsCART3 = predict(CARTmodel3)

And then compute the sum of squared differences between the actual values and the predicted values:

sum((statedata$Life.Exp - PredictionsCART3)^2)

The SSE is 9.312442.

Note that the SSE is not zero here - we still make some mistakes. This is because there are other parameters in rpart that are also trying to prevent the tree from overfitting by setting default values. So our tree doesn't necessarily have one observation in each bucket - by setting minbucket=1 we are just allowing the tree to have one observation in each bucket.

Hide Answer *You have used 1 of 3 submissions*

PROBLEM 2.7 - CART MODELS (1/1 point)

This is the lowest error we have seen so far. What would be the best interpretation of this result?

- ○ Trees are much better than linear regression for this problem because they can capture nonlinearities that linear regression misses.
- ◉ We can build almost perfect models given the right parameters, even if they violate our intuition of what a good model should be. ✔
- ○ Area is obviously a very meaningful predictor of life expectancy, given we were able to get such low error using just Area as our independent variable.

EXPLANATION

The correct answer is the second one. By making the minbucket parameter very small, we could build an almost perfect model using just one variable, that is not even our most significant variable. However, if you plot the tree using prp(CARTmodel3), you can see that the tree has 22 splits! This is not a very interpretable model, and will not generalize well.

The first answer is incorrect because our tree model that was not overfit performed similarly to the linear regression. Trees only look better than linear regression here because we are overfitting the model to the data.

The third answer is incorrect because Area is not actually a very meaningful predictor. Without overfitting the tree, our model would not be very accurate only using Area.

Hide Answer *You have used 1 of 1 submissions*

PROBLEM 3.1 - CROSS-VALIDATION (1/1 point)

Adjusting the variables included in a linear regression model is a form of model tuning. In Problem 1 we showed that by removing variables in our linear regression model (tuning the model), we were able to maintain the fit of the model while using a simpler model. A rule of thumb is that simpler models are more interpretable and generalizeable. We will now tune our regression tree to see if we can improve the fit of our tree while keeping it as simple as possible.

Load the *caret* library, and set the seed to 111. Set up the controls exactly like we did in the lecture (10-fold cross-validation) with *cp* varying over the range 0.01 to 0.50 in increments of 0.01. Use the *train* function to determine the best *cp* value. What value of cp does the train function recommend? (Remember that the train function tells you to pick the largest value of cp with the lowest error when there are ties, and explains this at the bottom of the output.)

0.12

0.12

Answer: 0.12

EXPLANATION

You can load the library caret and set the seed by typing the following commands:

library(caret)

set.seed(111)

Then, you can set up the cross-validation controls by typing:

fitControl = trainControl(method = "cv", number = 10)

cartGrid = expand.grid(.cp = seq(0.01, 0.5, 0.01))

You can then use the train function to find the best value of cp by typing:

train(Life.Exp ~ ., data=statedata, method="rpart", trControl = fitControl, tuneGrid = cartGrid)

At the bottom of the output, it says that the best value of cp is 0.12.

Hide Answer *You have used 2 of 4 submissions*

PROBLEM 3.2 - CROSS-VALIDATION (2/2 points)

Create a tree with this value of *cp*. You'll notice that this is actually quite similar to the first tree we created with the initial model. Interpret the tree: we predict the life expectancy to be 70 if the murder rate is greater than or equal to

6.6

6.6

Answer: 6.6

and is less than

11

11

Answer: 11

EXPLANATION

You can create a new tree with cp=0.12 by typing:

CARTmodel4 = rpart(Life.Exp ~ ., data=statedata, cp=0.12)

Then, if you plot the tree using prp(CARTmodel4), you can see that the life expectancy is predicted to be 70 if Murder is greater than or equal to 6.6 (the first split) and less than 11 (the second split).

Hide Answer *You have used 1 of 4 submissions*

PROBLEM 3.3 - CROSS-VALIDATION (1/1 point)

Calculate the SSE of this tree:

32.86549

32.86549

Answer: 32.9

EXPLANATION

To compute the SSE, first make predictions:

PredictionsCART4 = predict(CARTmodel4)

and then compute the sum of squared differences between the actual values and the predicted values:

sum((statedata$Life.Exp - PredictionsCART4)^2)

The SSE is 32.86549

Hide Answer *You have used 1 of 3 submissions*

PROBLEM 3.4 - CROSS-VALIDATION (1/1 point)

Recall the first tree (default parameters), second tree (minbucket = 5), and the third tree (selected with cross validation) we made. Given what you have learned about cross-validation, which of the three models would you expect to be better if we did use it for prediction on a test set? For this question, suppose we had actually set aside a few observations (states) in a test set, and we want to make predictions on those states.

- ○ The first model
- ○ The second model
- ◉ The model we just made with the "best" cp ✔

EXPLANATION

The purpose of cross-validation is to pick the tree that will perform the best on a test set. So we would expect the model we made with the "best" cp to perform best on a test set.

Hide Answer *You have used 1 of 1 submissions*

PROBLEM 3.5 - CROSS-VALIDATION (1/1 point)

At the end of Part 2 we made a very complex tree using just Area. Use *train* with the same parameters as before but just using Area as an independent variable to find the best cp value (set the seed to 111 first). Then build a new tree using just Area and this value of cp.

How many splits does the tree have?

4

4

Answer: 4

EXPLANATION

To find the best value of cp when using only Area, use the following command:

set.seed(111)

train(Life.Exp ~ Area, data=statedata, method="rpart", trControl = fitControl, tuneGrid = cartGrid)

Then, build a new CART tree by typing:

CARTmodel5 = rpart(Life.Exp ~ Area, data=statedata, cp=0.02)

You can plot the tree with prp(CARTmodel5), and see that the tree has 4 splits.

Hide Answer *You have used 1 of 4 submissions*

PROBLEM 3.6 - CROSS-VALIDATION (2/2 points)

The lower left leaf (or bucket) corresponds to the lowest predicted Life.Exp, (70). Observations in this leaf correspond to states with area greater than

9579

9579

Answer: 9579

and area less than

51000

51000

Answer: 51000

EXPLANATION

To get to this leaf, we go through 3 splits:

Area less than 62,000

Area greater than or equal to 9,579

Area less than 51,000

This means that this leaf is composed of states that have an area greater than 9,579 and less than 51,000.

Hide Answer *You have used 1 of 4 submissions*

PROBLEM 3.7 - CROSS-VALIDATION (1/1 point)

We have simplified the previous "Area tree" considerably by using cross-validation. Calculate the SSE of the cross-validated "Area tree", and select the correct statements:

- [ ] The best model in this whole question is the first "Area tree" because it had the lowest SSE.
- [x] The Area variable is not as predictive as Murder rate. ✔
- [ ] Cross-validation is intended to decrease the SSE for a model on the training data, compared to a tree that isn't cross-validated.
- [ ] Cross-validation will always improve the SSE of a model on unseen data, compared to a tree that isn't cross-validated.

EXPLANATION

You can calculate the SSE by first making predictions and then computing the SSE:

PredictionsCART5 = predict(CARTmodel5)

sum((statedata$Life.Exp - PredictionsCART5)^2)

The original Area tree was overfitting the data - it was uninterpretable. Area is not as useful as Murder - if it was, it would have been in the cross-validated tree. Cross-validation will never improve the fit on the training data, but it won't necessarily make it significantly worse. Cross-validation cannot guarantee improving the SSE on unseen data, although it often helps.

Hide Answer *You have used 1 of 2 submissions*

Please remember not to ask for or post complete answers to homework questions in this discussion forum.

Show Discussion New Post