Lab- Single and Ensemble Tree Models

Jay Singhvi

04/27/2024

Decision Tree Models: Single and Ensemble Methods.

In this lab you will fit and evaluate a single decision tree model and compare it to an random forest ensemble tree model, and a boosted tree model.

Data Processing. The data set comes from a study of how certain demographics affect house prices in Boston, MA. Our goal will be to predict the median value of a house, medv.

crim: per capita crime rate by town.
zn: proportion of residential land zoned for lots over 25,000 sq.ft.
indus: proportion of non-retail business acres per town.
chas: Charles River dummy variable (= 1 if tract bounds river; 0 otherwise).
nox: nitrogen oxides concentration (parts per 10 million).
rm: average number of rooms per dwelling.
age: proportion of owner-occupied units built prior to 1940.
dis: weighted mean of distances to five Boston employment centres.
rad: index of accessibility to radial highways.
tax: full-value property-tax rate per $10,000.
ptratio: pupil-teacher ratio by town.
lstat: lower status of the population (percent).
medv: median value of owner-occupied homes in $1000s.

QUESTION 1 (2 parts, 4pts)- 28 total Lab points MAP TO GRADESCOPE

1a(2pts): Write a statement in the chunk below that reads in the data file "boston_house_prices.csv". There are two variables that are categorical. Write statements to assign them to factors.

```
house.df = read.csv("boston_house_prices.csv")
house.df$chas = factor(house.df$chas)
house.df$rad = factor(house.df$rad)
```

Training and Testing Sets
1b(2pts): Use the createDataPartition function from the caret library to create subsets for training and testing, with 75% of the data used for training, and 25% held out for testing. Assign the training set to a variable called train.data, and the test set to test.data.

```
RNGversion("4.1.2")
set.seed(123456)

index.train = createDataPartition(y = house.df$medv, p = 0.75, list = FALSE)
```

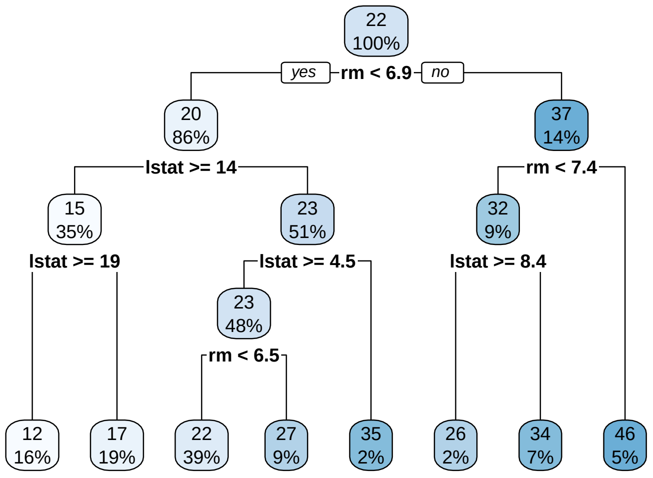
```r
train.data = house.df[index.train,]
test.data = house.df[-index.train,]
```

QUESTION 2 (4 parts, 8pts) MAP TO GRADESCOPE

Fitting and Evaluating a Regression Tree. **2a(2pts):** Use the rpart function to fit a regression tree model to the training data. The predicted variable is medv. The model formula includes all of the predictors. Remember the method for a regression tree is "anova".

Add another statement that calls the rpart.plot function to print the tree.

```r
house.rpart = rpart(medv ~ ., data = train.data, method = "anova")
rpart.plot(house.rpart)
```



2b(2pts): List the predictors that are included in the model. What does each predictor measure?

rm : average number of rooms per dwelling , lstat : lower status of the population (percent)

2c(2pts): What predictor is used for the first split?

rm

Prediction Performance of Single Tree.
2d(2pts): Call predict function, passing in the fitted model and the test data.

```r
single.tree.pred  <- predict(house.rpart, test.data)
```

These statements calculate the RMSE and MAE for the model's predictions compared to the actual values of the predicted variable in the test set.

```r
sqrt(mean((single.tree.pred-test.data$medv) ^2))
```

```
## [1] 4.970967
```

```r
mean(abs(single.tree.pred-test.data$medv))
```

```
## [1] 3.330831
```

QUESTION 3 (3 part2, 6pts) MAP TO GRADESCOPE

Ensemble Method: Random Forest. One issue with single decision trees is that they tend to overfit and have lower predictive accuracy. They also tend to select only the most significant predictors and exclude any other possible contributions from other predictors. Ensemble tree methods such as random forest address these issues by creating many trees and sampling the set of predictors at each step in the creation of each tree. The prediction is a combination of all of the single tree predictions.

Bagging, or Bootstrap Aggregation, is also used, The data is sampled a number of times to create testing sets. Each bootstrapped sample is modeled by a single, unpruned tree. The results of all of the trees are averaged together. For each training sample selected by bootstrapping, the remaining data is used for testing that model. The "bag" refers to the training data, and the "out of bag", or OOB, refers to the test data. The prediction error for all trees combined is called the "out of bag error", or OOBE.

The "mtry" parameter determines the number of variables the model can use at each selection point (node).

3a(2pts): In the code chunk below, fit a random forest to the training data. Call the randomForest function, passing in the same formula as the single regression tree, the training data, specify mtry=6, and importance=TRUE. The last parameter is so we get a read out of the relative importance of the predictors across all of the trees.

```
RNGversion("4.1.2")
set.seed(123456)
rndm.frst = randomForest(medv ~ ., data = train.data, mtry = 6, importance = TRUE )
```

3b(2pts): How many trees were grown by the random forest model? The attribute that stores this value can be found in the list of attributes in the Value section of the Help page that you can access by executing ?randomForest in the console.

500

3c(2pts): Obtain the predictions from the model, then calculate the RMSE and MAE for the Random Forest model's predictions vs the actual values in the test set. Use the code from the single tree as an example.

```
rndm.frst.pred  <- predict(rndm.frst, test.data)
sqrt(mean((rndm.frst.pred-test.data$medv) ^2))
```

```
## [1] 3.133001
```

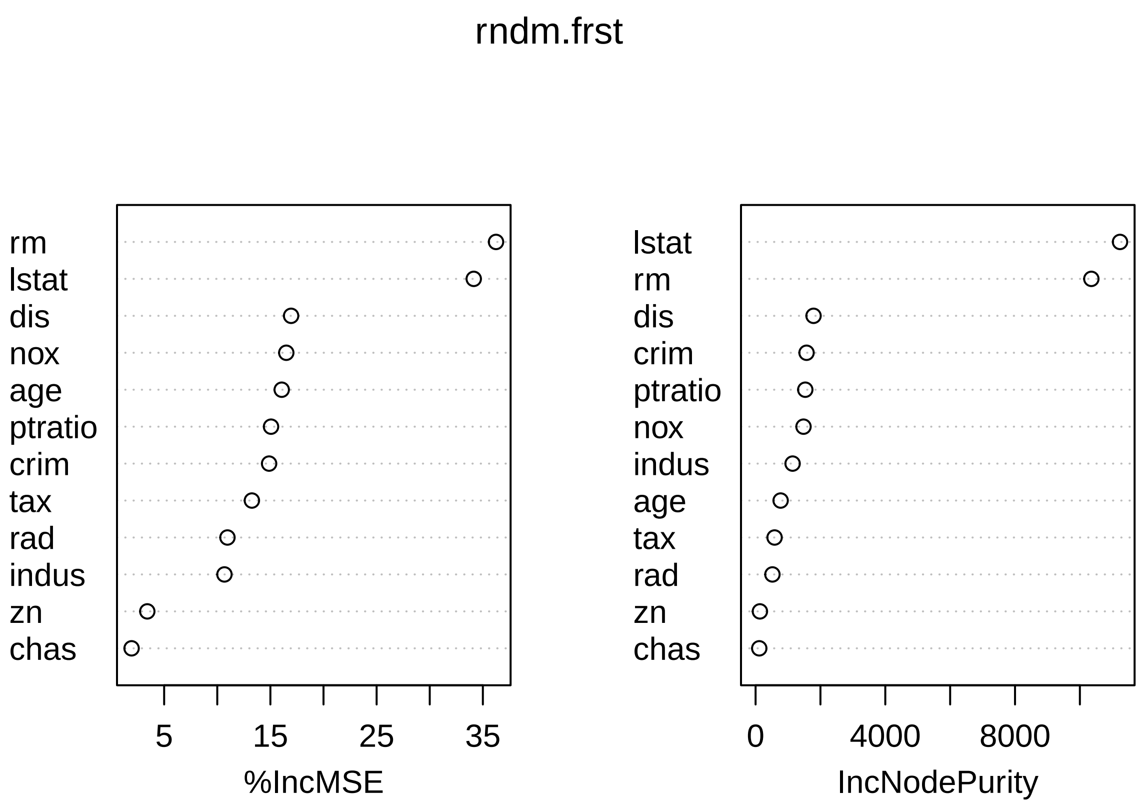
```
mean(abs(rndm.frst.pred-test.data$medv))
```

```
## [1] 2.14391
```

QUESTION 4 (1 part, 2pts) MAP TO GRADESCOPE

Execute this code which allows you to view the importance of the predictors for the random forest model in a plot.

```r
varImpPlot(rndm.frst)
```



4a(2pts): What are the top two most important variables? How does this compare to the top two most important variables for the single tree models?

rm and lstat, which are the same perdictors used by single tree model

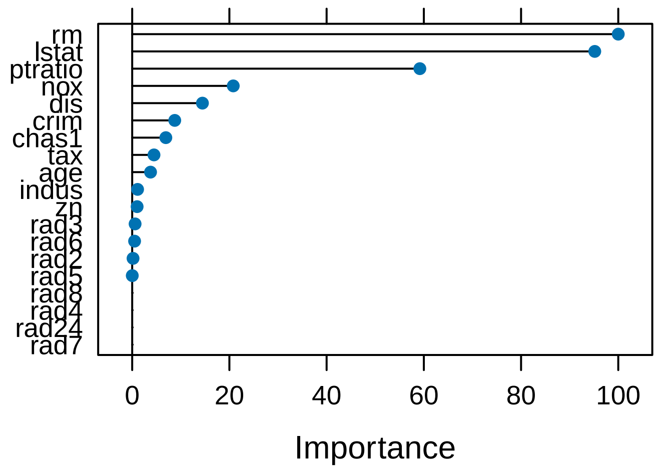
QUESTION 5 (2 parts, 4pts) MAP TO GRADESCOPE

Boosting Boosting is similar to Bagging and Random Forests except that a single tree is grown sequentially, where the next tree uses error information from the previous tree. Thus, the next tree gets a "boost" from the previous tree.

This code calls the train function to create the boosted tree model using the xgboost library. Note that there are a lot of computations involved, and this may take some time to complete.

Execute this code to produce a plot of the predictors and their importance in splitting the data for the boosted tree model.

```
pred.imp <- varImp(boost.boston)
plot(pred.imp)
```



5a(2pts): What are the top two predictors listed?

rm and lstat

5b(2pts): Now obtain the predictions from the boosted model and calculate the RMSE and MAE for the boosted model.

```
boost.boston.pred = predict(boost.boston, test.data)
sqrt(mean((boost.boston.pred-test.data$medv) ^2))
```

```
## [1] 3.408525
```

```
mean(abs(boost.boston.pred-test.data$medv))
```

```
## [1] 2.295926
```

QUESTION 6 (2 parts, 4pts) MAP TO GRADESCOPE

Summary.
6a(2pts): Fill in this table which summarizes the results for the three models. Round the values for RMSE and MAE to the nearest hundredth: x.xx

Model	RMSE	MAE
Single Tree	4.97	3.33
Random Forest	3.13	2.14
Boosted model	3.41	2.30

6a(2pts): Briefly summarize the results from this analysis. Is the computational complexity required (long running time) for the boosted model worth it?

From the table it is clearly visible that the rmse and mae for random forest is less compared to single tree and boosted model so the computational complexity for the boosted model was not worth it as it still produced more rmse and mae compared to random forest.