Project Report

Department of Datascience
2020047029
JunYeong Ahn

Introduction

This report is about comparing two different models with a single performance score using R language. I selected *Multiple Linear Regression(MLR)* and *Random Forest(RF)* as two models - the start was *MLR* versus *Decision Tree(DT)*, which both are loved for their simplicity in mathematical calculation and interpretation. Tree-based models are very unfamiliar in themselves to me, so I thought that dealing with *RF* (harder version of DT) would be not only challenging but also good for my improved understanding on how tree-based models work.

Body

With each model containing certain variables, I'm going to predict 'medv'(median value of owner-occupied homes in $1000's) using Boston data which is embedded into MASS(one of R package) then compare which model is better. To do so, I optimized my MLR and RF model, respectively through Best Subset selection and 10-fold CV.

● 1. Multiple Linear Regression

Multiple Linear Regression is a linear approach to modeling the relationship between a dependent variable and explanatory variables.

At first, split the Boston data into train set(80%) and test set(20%) with setting certain seed to use these sets in RF as well to assess my models' performance scores by predicting 'medv' with completely unseen data (test set) later on. Then fit our model with the train set, including all the 13 predictors. Seeing p-values of predictors of this model, it is clear that some of them whose value is larger than 0.05 are nuisance, which means we can't reject the Null Hypothesis that the certain predictor has nothing to do with 'medv'.

To get rid of unimportant predictors, I'll use Best Subset selection who compares all possible models using a specified set of predictors and displays the best-fitting models that contain one predictor, two predictors and so on - very unsophisticated, but also very intuitive. Regsubsets() and summary() respectively show us the best model (based on the training data) for each No. of *p* and model evaluation criteria such as RSq, RSS, AdjR2, Cp and BIC.

However, RSq and RSS are inappropriate to estimate test error for this report since they are closely associated with train error and often results in the conclusion that 'The more variables, the better model is.' - they cannot be used to select from among a set of models with different numbers of variables. Other three are applicable approaches which indirectly estimate test error by making an adjustment to the training error to account for the bias due to overfitting.

In this report, I conducted 10-fold cross validation for each possible model to decide which number of *p* is ideal, for this procedure has an advantage relative to AIC, BIC, Cp, and adjusted R2 in that it provides a direct estimate of the test error, and makes fewer assumptions about the true underlying model. 10-fold CV divides the data into 10 groups and uses one of them as a validation set then averages the results – here, MSE (Mean Squared Error : the average squared difference between the estimated values and the actual value) of 9 training folds with a changing validation set - from 10 repetitions.

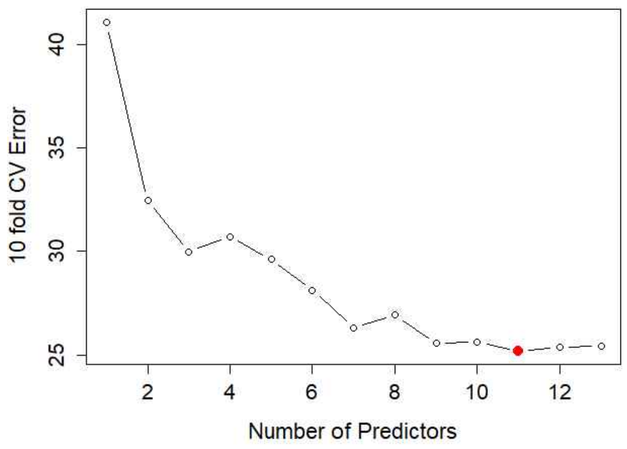
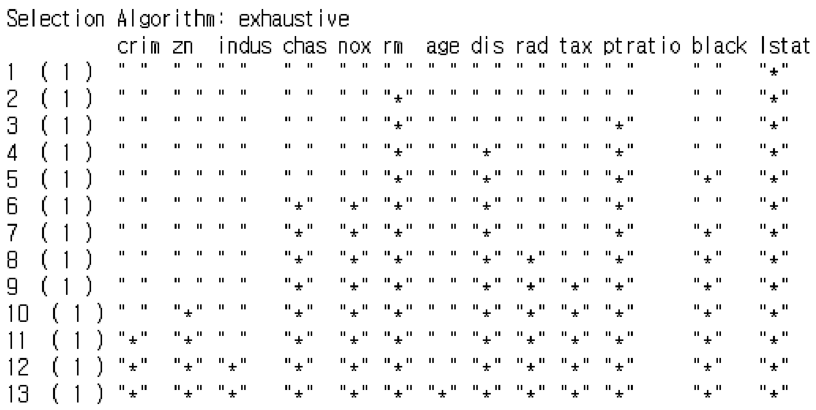


FIGURE 1.1. For each possible model containing a subset of the 13 predictors in the Boston train set, 10 – fold CV Errors are displayed. This shows that there would be the least test error when I choose the 11 predictors among 13.

Based on **FIGURE 1.1.**, it seems we'd better use 11 predictors for this model because it has the minimal 10-fold CV error.

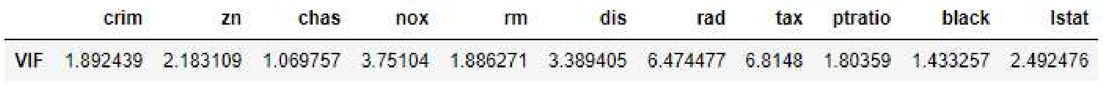


```
Selection Algorithm: exhaustive
          crim zn  indus chas nox rm  age dis rad tax ptratio black lstat
1  ( 1 )  " "  " " " "   " "  " " " " " " " " " " " " " "     " "   "*"
2  ( 1 )  " "  " " " "   " "  " " "*" " " " " " " " " " "     " "   "*"
3  ( 1 )  " "  " " " "   " "  " " "*" " " " " " " " " "*"     " "   "*"
4  ( 1 )  " "  " " " "   " "  " " "*" " " "*" " " " " "*"     " "   "*"
5  ( 1 )  " "  " " " "   " "  " " "*" " " "*" " " " " "*"     "*"   "*"
6  ( 1 )  " "  " " " "   "*"  "*" "*" " " "*" " " " " "*"     " "   "*"
7  ( 1 )  " "  " " " "   "*"  "*" "*" " " "*" " " " " "*"     "*"   "*"
8  ( 1 )  " "  " " " "   "*"  "*" "*" " " "*" "*" " " "*"     "*"   "*"
9  ( 1 )  " "  " " " "   "*"  "*" "*" " " "*" "*" "*" "*"     "*"   "*"
10  ( 1 ) " "  "*" " "   "*"  "*" "*" " " "*" "*" "*" "*"     "*"   "*"
11  ( 1 ) "*"  "*" " "   "*"  "*" "*" " " "*" "*" "*" "*"     "*"   "*"
12  ( 1 ) "*"  "*" "*"   "*"  "*" "*" " " "*" "*" "*" "*"     "*"   "*"
13  ( 1 ) "*"  "*" "*"   "*"  "*" "*" "*" "*" "*" "*" "*"     "*"   "*"
```

FIGURE 1.2. The result of summary() function on Best Subset selection method

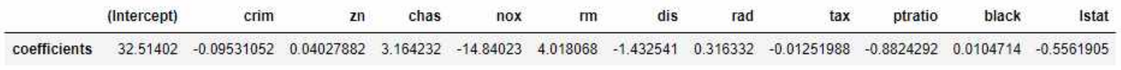
(regsubsets()) with all the predictors (13). When No. of *p* is 11, only 'crim' amd 'age' are excluded from our best MLR model.

Thus, seeing the **FIGURE 1.2.**, we can know that **crim, zn, chas, nox, rm, dis, rad, tax, ptratio, black and lstat** predictor are left after Best Subset selection. To reflect the result, I fit the model with those 11 predictors and train set, which was named **lm.fit**. As a final test of this linear regression model, VIF (Variance Inflation Factor) for each predictor is calculated to avoid Multicollinearity problem – of course it's a problem because it undermines the statistical significance of an independent variable - where one independent variable is well predicted by another, or highly correlated with another one.



	crim	zn	chas	nox	rm	dis	rad	tax	ptratio	black	lstat
VIF	1.892439	2.183109	1.069757	3.75104	1.886271	3.389405	6.474477	6.8148	1.80359	1.433257	2.492476

TABLE 1.1. For each selected predictor VIF value is displayed. There is no fixed figure, but *generally* it is said it has Multicollinearity problem if the VIF value of certain predictor is higher than 10. No one has higher VIF value than 10, which could result in the conclusion that '*lm.fit has passed Multicollinearity test*' .



	(Intercept)	crim	zn	chas	nox	rm	dis	rad	tax	ptratio	black	lstat
coefficients	32.51402	-0.09531052	0.04027882	3.164232	-14.84023	4.018068	-1.432541	0.316332	-0.01251988	-0.8824292	0.0104714	-0.5561905

TABLE 1.2. A regression coefficients table for my final model **lm.fit**.

Finally after using predict() function over **lm.fit** model with test set then storing the result into y_hat, I get the manually-calculated MSE for later **Comparison** and also will elaborate on why I pick MSE as the performance score for comparing two models. Anyway, using the final model **lm.fit who contains 11 variables on TABLE 1.1. to predict 'medv'**, calculated MSE with test data is 17.23434.

● 2. Random Forest

Random Forest (RF) regression is an ensemble learning method that operates by constructing a multitude of Decision Trees (DTs) at training time and outputting the class that is the mode of the mean/average prediction of the individual trees. When constructing multiple DTs, each time a split in a tree is considered, a random sample of m predictors is chosen as split candidates from the full set of p predictors.

Here I fit our model using randomForest() function with the train set. Unlike MLR, however, our main interest is the best value of 'mtry (No. of variables randomly sampled as candidates at each split)' argument of the function – total 500 trees will be created by default and I made no change of the default because this part is not for finding the best model but for identifying how 'mtry' is important. Although the default value is *p*/3; so 13/3 -> 4 would be default of the argument, I'll find the optimized value of the 'mtry' by 10-fold cross validating.

Therefore, this part can be separated into broad three steps : 1. 10-fold CV / 2. Modelling and Interpretation / 3. Calculating MSE as the performance score.

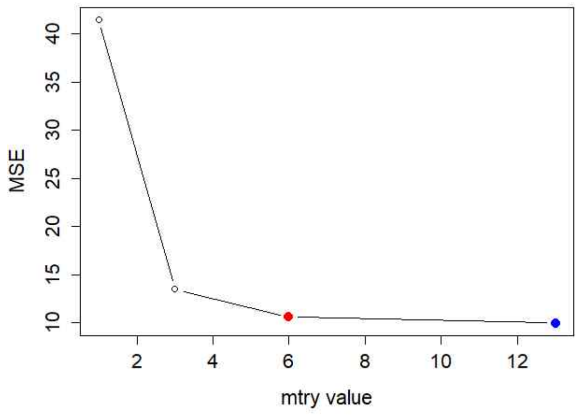


FIGURE 2.1. Through 10-fold CV using rfcv() function from 'randomForest' package, for each possible No. of variables for 'mtry' MSEs are displayed. Although the blue point, the lowest MSE (9.9007), is shown with 13 variables, the red point (6, 11.23100) would be the better one because it only takes less than a half of the full variables but there' s almost no gap between MSEs of red point and blue point.

To reflect the result of **FIGURE 2.2.**, I fit the RF model on train data with 'mtry' being 6, which was named **rf.boston**.

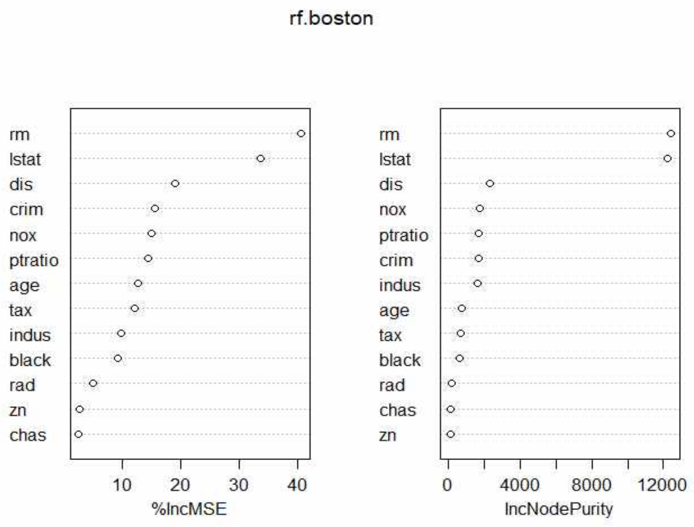


FIGURE 2.2. %IncMSE : Based on the average reduction in predicted accuracy for testing samples when a given variable is excluded from the model. / IncNodePurity : The average value for all trees of the total reduction in Node impurity due to division into a given variable. And Node impurity is measured by Training RSS. Both panels are indicating that 'rm' and 'lstat' are important in predicting 'medv'.

Calculating in the same way as MLR, MSE of **rf.boston** with test data is 11.90841.

● **Comparison**

I separate this Comparison part to do compare two models and also argue why I chose MSE as a performance score; 1. R^2 and adjusted R^2, which are what we've usually used for model accuracy evaluation, cannot be used in comparing two different models containing different No, of *p* because the value of them lose their meaning apart from the model – number vs. number is invalid. 2. These models are for predicting values through regression so I wanted to see how well my model predicts the 'medv' when novel observations are given. 3. CV of MSE is still not enough to meet my needs because CV doesn't tell us the performance with totally novel data – my models are already optimized with train set(80% of Boston data) to yield predictions on test set(20%) which is completely unseen before. So considering MSE alone, RF(11.90841) is better than MLR (17.23434).

Conclusion

Considering MSE, Random Forest turns out to be better than Linear Regression at predicting 'medv'(median value of owner-occupied homes in $1000's) with test set of Boston data. Again, the test set MSE associated with the RF was 11.90841 but this is the sum of 'squared form' of errors. The square root of the MSE is therefore around 3.45, indicating that this model leads to test predictions that are within around $3450 of the true median home value for the suburb. Given that the average of 'medv' values in Boston data is about 22.5 (so $22500 on average) and the standard deviation is around 9.1 ($9100), prediction of RF model is quite decent and better than that of MLR in predicting the median home price in Boston.

In addition, RF has the advantage that the estimation of importance could be used for variable selection in the study of complex areas (although this kind of advantage was not considered in comparison). And given that I didn't do some more sophisticated job like analysis on OOB estimation or ntree (the number of trees whose default was 500 in this report) manipulation to proceed the study in the most classic way, this model has a chance to evolve to yield even more precise predictions on Boston housing price.