PSTAT 131 Final Project

Matthew Li & Becky Su

6/4/2020

Introduction

The House Prices Dataset from Kaggle compiled by Dean De cock serves as an updated alternative to the Boston Housing Prices dataset that is a record of 1460 housing sales and it's attributes in Ames, Iowa from 2006 to 2009. This dataset has 80 variables that focus on the quality and quantity of many physical attributes of the property. This data serves as relevant information for future home buyers as well as those looking to analyze trends of housing prices in Ames, Iowa.

In this analysis, we will use machine learning techniques such as KNN, decision tree, and random forests to determine what constitutes an expensive, or above average sale relative to the rest of the area. From this, we can also compare the techniques and identify the best method to analyze our objective.

Data Overview

We used sale attributes such as lot size, house condition, etc. to see if we can use machine learning to determine algorithmically why a house is expensive or cheap.

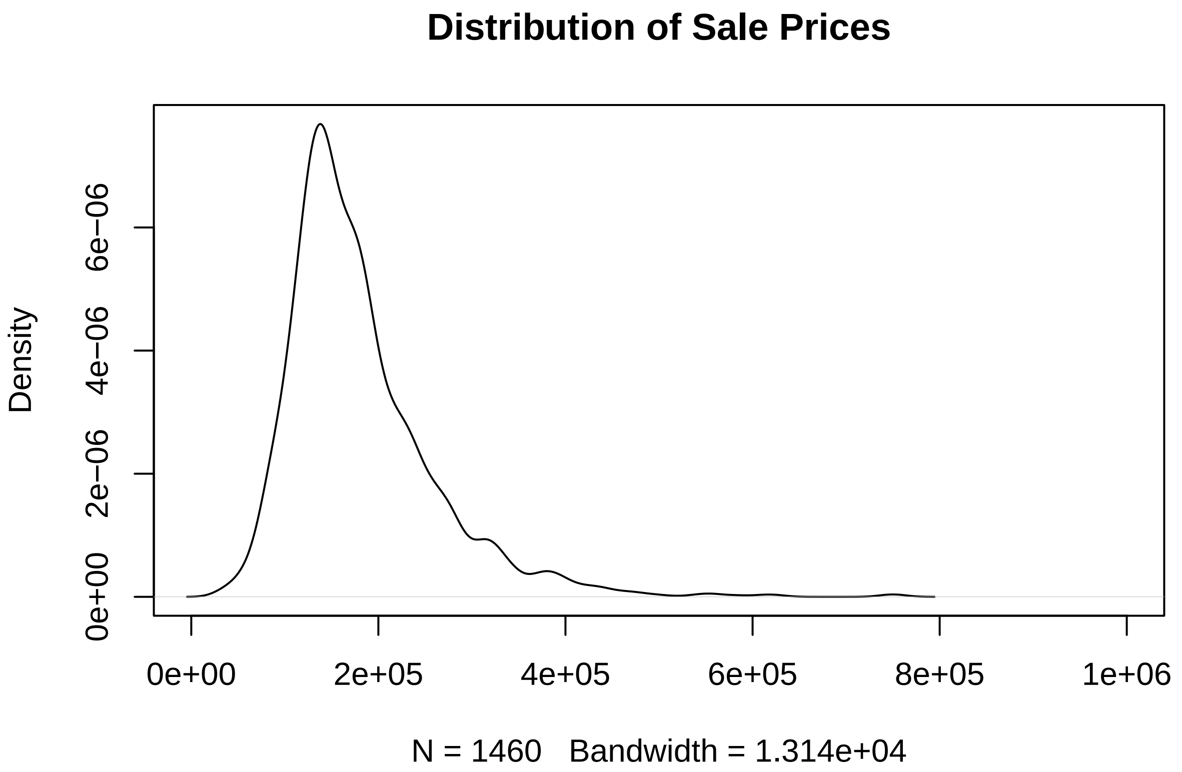
Our dataset was imported from Kaggle and it has 1460 observations and 6 predictors with no missing data values. We convert SalePrice, our response variable, to a new binary column named SaleClass so we can use predictors to classify the response. An above average SalePrice is labelled "expensive" and below average SalePrice is labelled as "cheap".

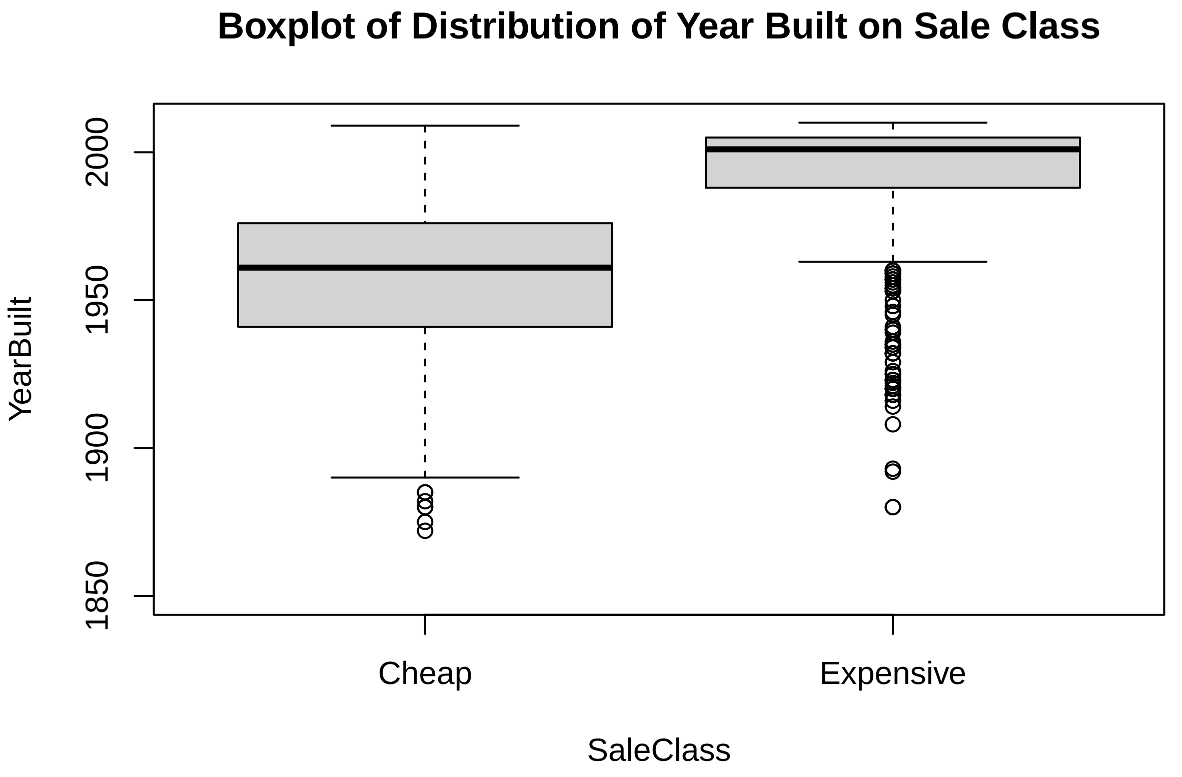
We have 6 numeric predictors and 1 two level categorical variable, (SaleClass). OverallQual is a rating of the quality of materials used to build the house on a scale from 1 to 10, while OverallCond is a rating of the actual condition and upkeep of the house. The other predictor variables should be self explanatory. We're going to apply a few different classification methods in order to firstly determine which the best model for predicting is in terms of the relevant variables, and secondly to find the best classification algorithm for this data.

Methods and Model Building

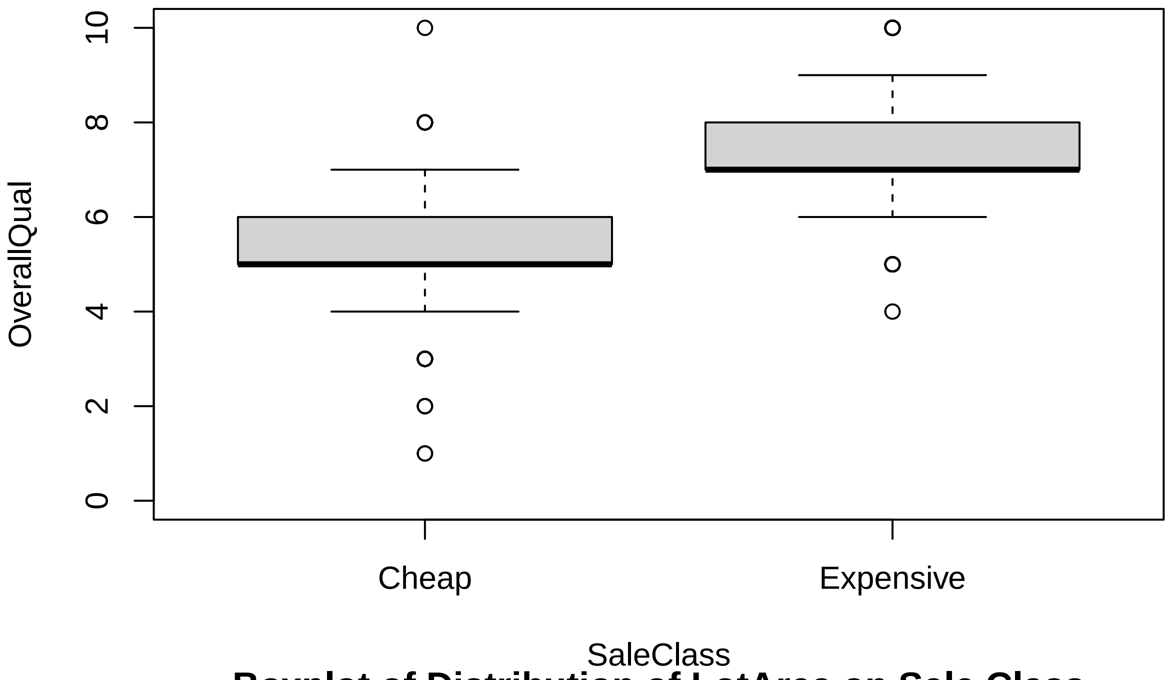
We first plot a density function of sale price to see the distribution of sale prices; we also plot a boxplot of SaleClass with each of the predictor variables.The explanatory and response variables are generally correlated as we expected. Houses are more likely to be in the expensive class when they are higher quality, with a larger lot size, etc.

Exploratory Graphics

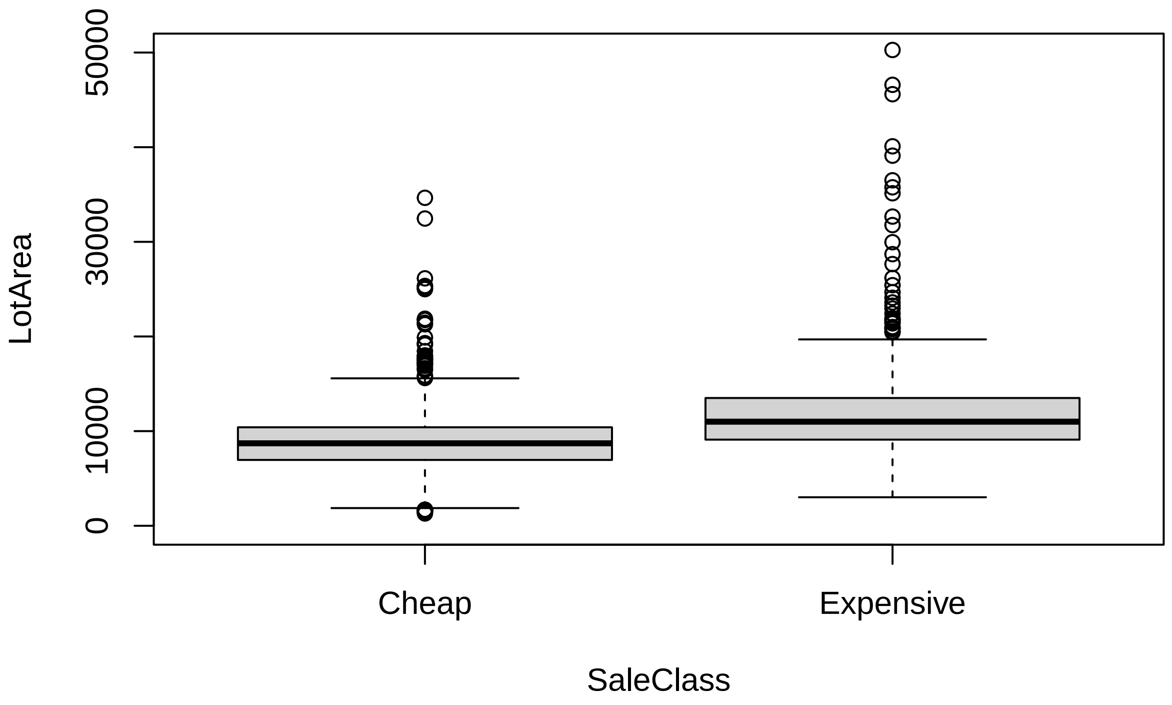


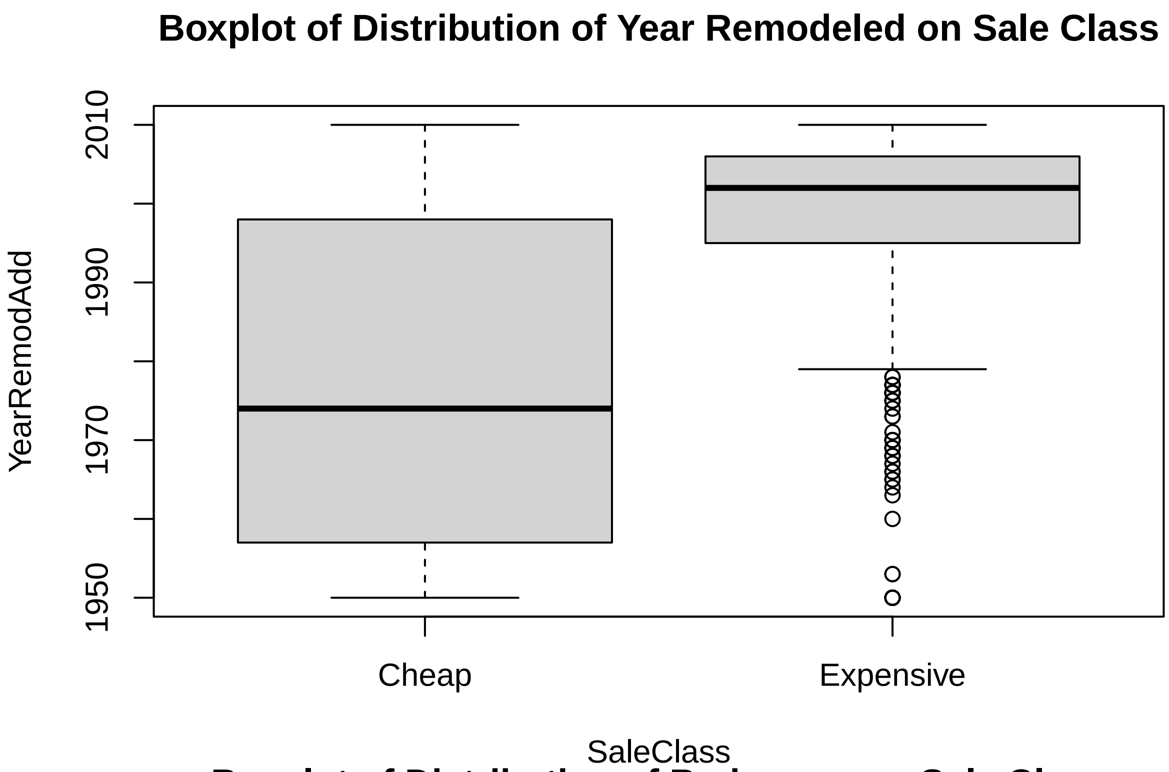


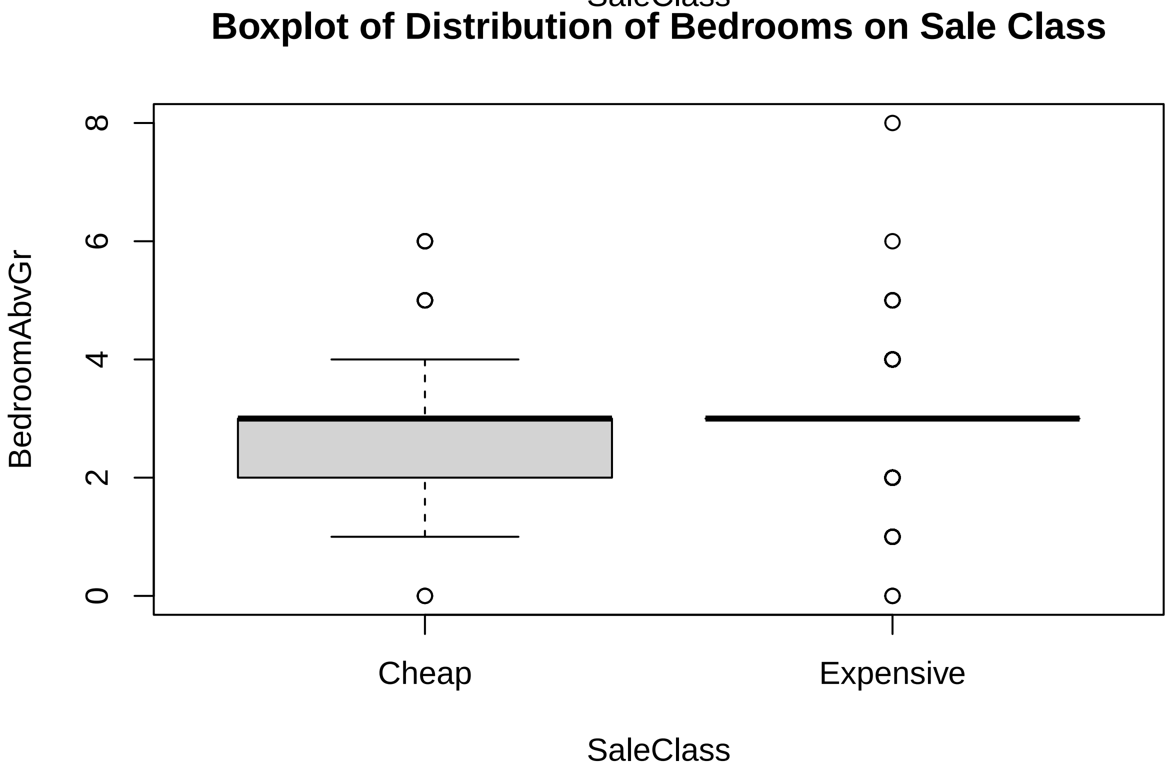
Boxplot of Distribution of Overall Quality on Sale Class



Boxplot of Distribution of LotArea on Sale Class







The density plot and boxplots gives us insight on the distribution of each predictor on the response variable "SaleClass". In each plot, we can see that there is a trend that the earlier year built, the cheaper; the lower the overall quality, the cheaper; the less lot area, the cheaper; the earlier the year remodeled, the cheaper; and the house price doesn't significantly vary between number of bedrooms.

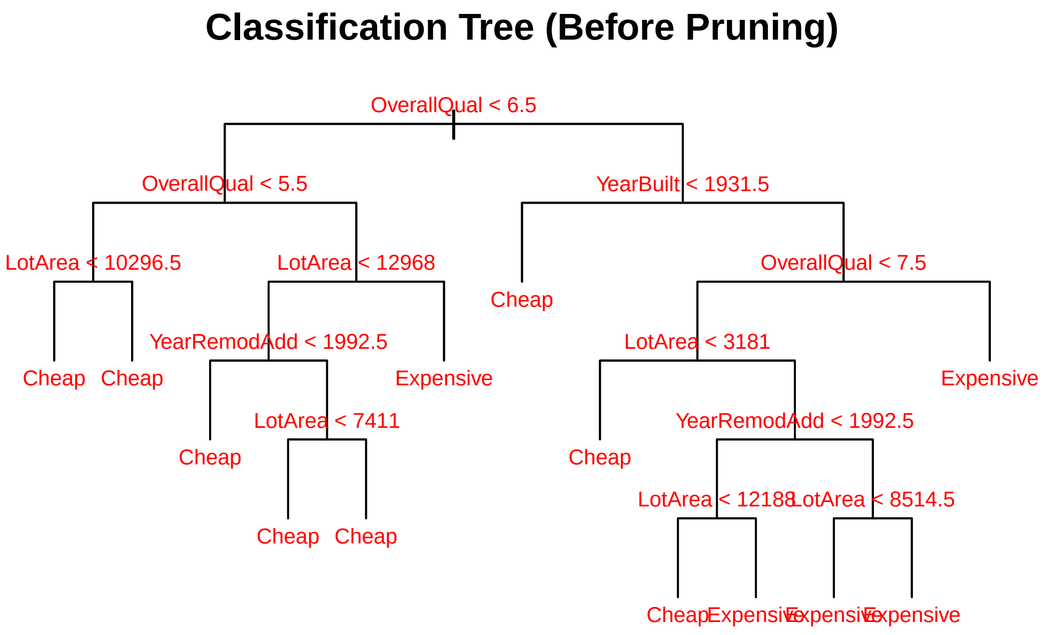
We're going to create a table to easily compare the quality of the different classification methods we're going to utilize going forward, namely decision trees (with k-fold cross validation to prune the tree), k-nearest neighbor, and randomForest.

In order to apply machine learning algorithms to this dataset, we need to stratify the dataset into a training set and a test set. The training set will be used to teach the classification model how to predict. We then apply the algorithm to the test set, and see how accurate the classification was.

Decision Tree

The first method we are going to perform on this dataset, is decision trees. The decision tree is a non-parametric classification method which uses a set of rules to predict that each observation belongs to the most commonly occurring class label of training data. We use SaleClass as the response, and each of the 6 numeric attributes as predictors.

```
##
## Classification tree:
## tree(formula = SaleClass ~ . - SalePrice, data = housing.train)
## Variables actually used in tree construction:
## [1] "OverallQual"  "LotArea"      "YearRemodAdd" "YearBuilt"
## Number of terminal nodes:  13
## Residual mean deviance:  0.4663 = 334.3 / 717
## Misclassification error rate: 0.09178 = 67 / 730
```



We can see from this summary, that 4 out of the 6 predictors were used in constructing this tree: OverallQual, LotArea, YearRemodAdd, YearBuilt. Now we are actually going to plot the tree to visualize this. We can build a confusion matrix after using the data to predict on the test set, and then find the accuracy rate and the error rate.

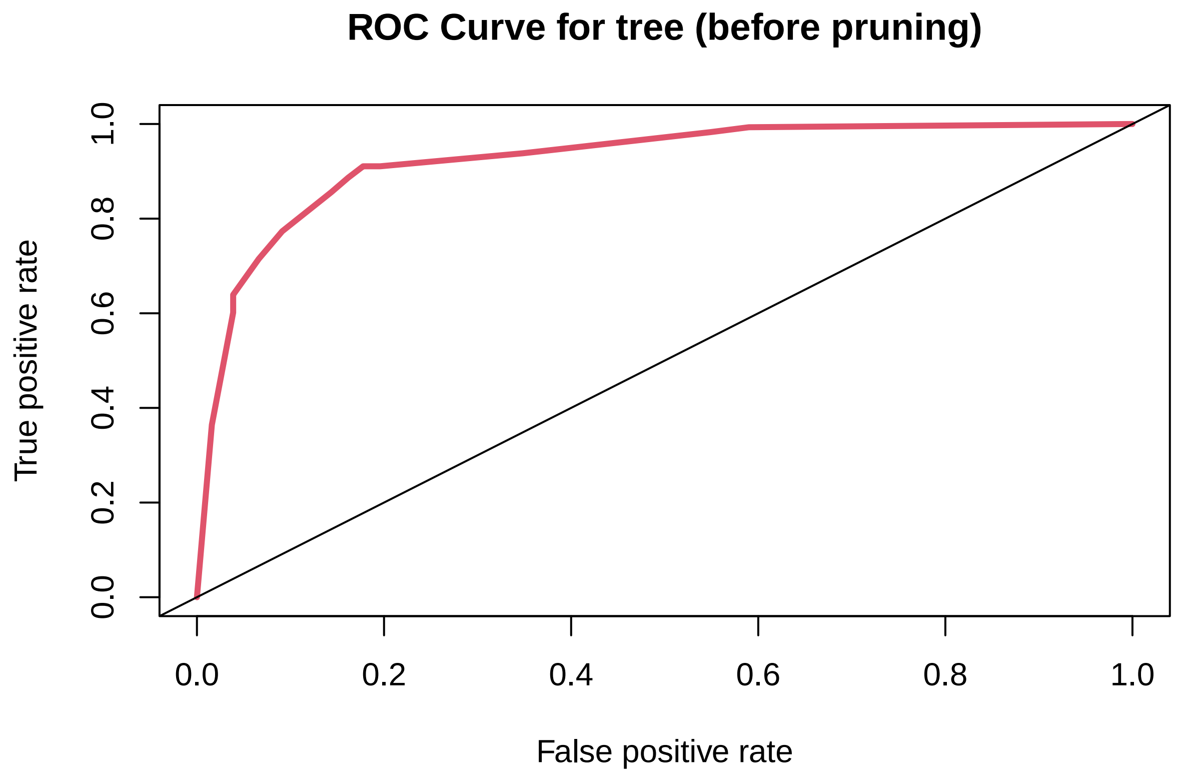
```
##
## yhat.test   Cheap Expensive
##   Cheap       399        66
##   Expensive    40       225
```

```
## [1] 0.8547945
```

```
## [1] 0.1452055
```

With an accuracy rate of 0.855, and an error rate of 0.145, this decision tree model is not bad. It will classify correctly more than 4 out of 5 times on average. Alternatively We can use the Receiver Operating

Characteristic (ROC) curve and the area underneath it (AUC). The ROC curve plots the false positive rate against the true positive rate, and the area underneath it falls between either 0.5 or 1, 0.5 being the worst (random classification), and 1 being the best (perfect classification).

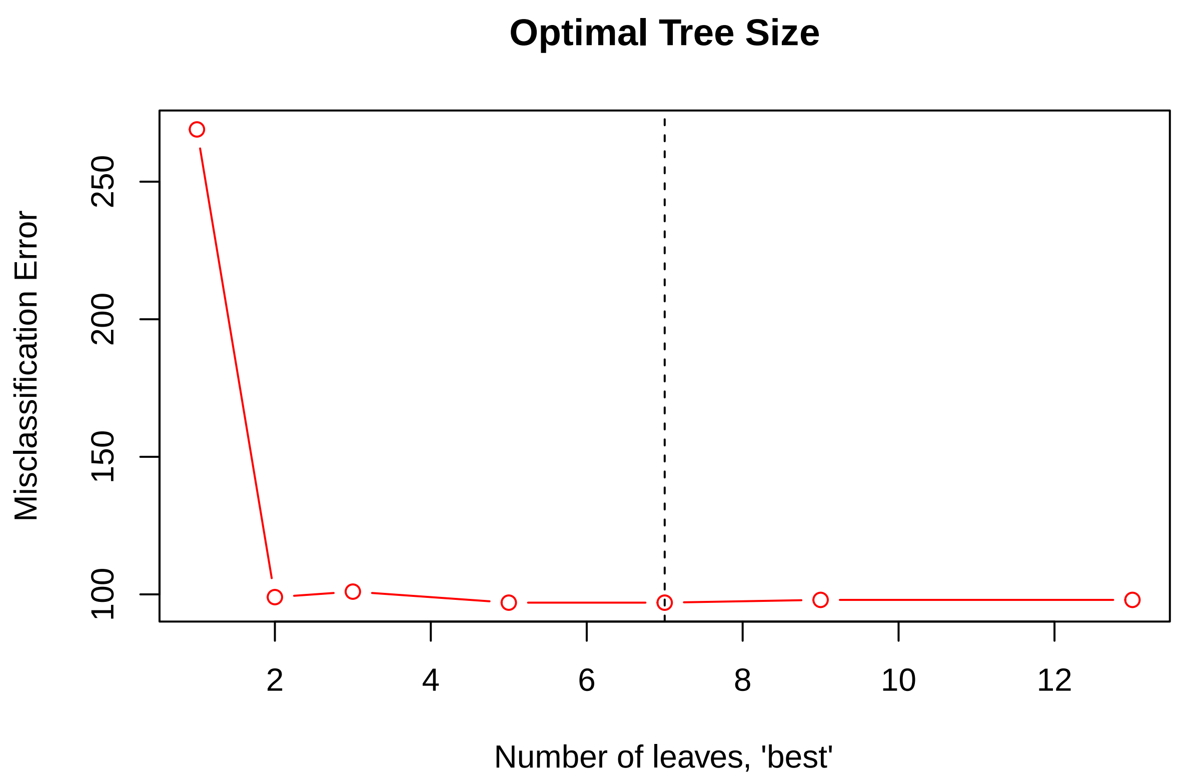


```
## [1] 0.9232558
```

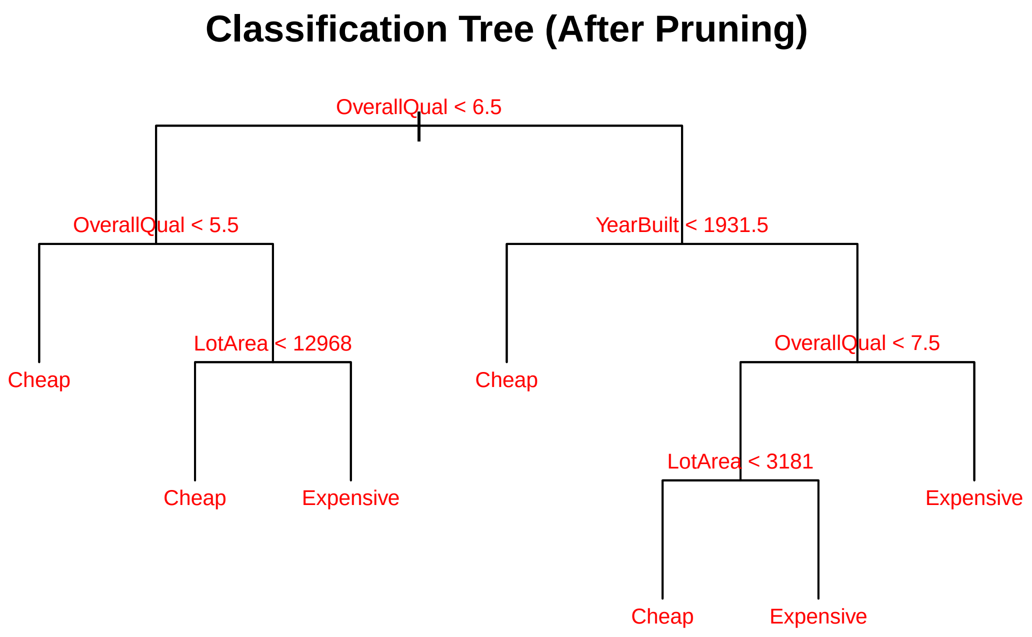
We get an AUC of 0.9233.

k-fold Cross Validation for Decision Tree

We can use k-fold cross-validation, which randomly partitions the dataset into folds of similar size, to see if the tree requires any pruning which can improve the model's accuracy as well as make it more interpretable for us.In k-fold cross validation, we divide the sample into k sub samples, then train the model on k -1 samples, leaving one as a holdout sample. We compute validation error on each of these samples, then average the validation error of all of them. The idea of cross-validation is that it will sample multiple times from the training set, with different separations. Ultimately, this creates a more robust model i.e. the tree will not be overfit.



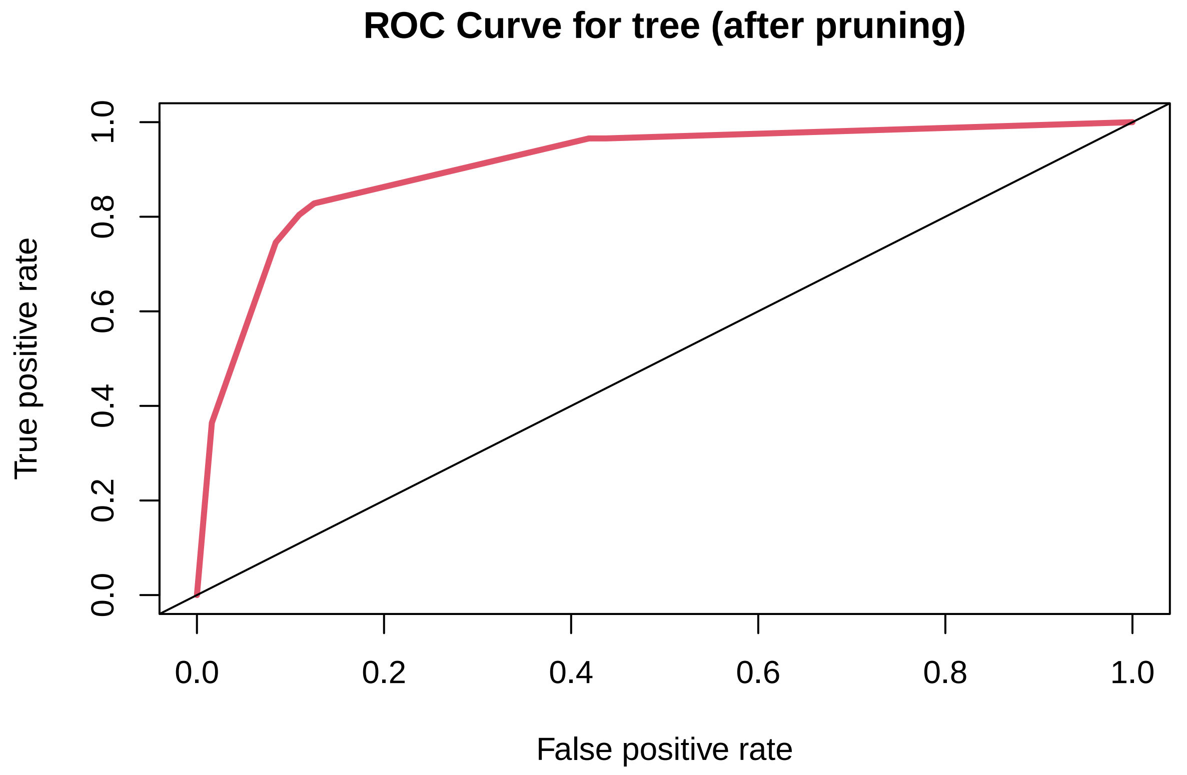
So we see, after running cross-validation, we see that we should prune the tree so that it has only 7 nodes. With this knowledge we can prune the tree and run the same diagnostics on it that we did on the unpruned model to see if any improvements are apparent.



```
##              true
## pred          Cheap Expensive
##   Cheap         391        57
##   Expensive      48       234
```

```
## [1] 0.8561644
## [1] 0.1438356
```

We see that pruning the tree didn't actually improve the accuracy rate of the model, although it did condense the number of relevant variables. Initially, seeing that accuracy actually decreased give the impression that pruning was not meaningful, but to the contrary, the fact that we were able to prune the tree without losing much accuracy shows that the sole variable we have remaining (overall quality) is almost just as good as classifying when using a decision tree as when using all 6 predictors. The original model being rather complex runs the risk of over-fitting, which is to say that the data follows the training data too closely and cannot be well generalized to new data. This is why we could be inclined to favor a simpler model such as that we found after pruning with cross-validation.



```
## [1] 0.9073887
```

KNN

K-nearest neighbors is a non parametric method of classification. It classifies each datapoint by plotting the test set in the same dimensional space as the training set, then classifies it based on the "k nearest neighbor(s)", i.e. if k = 5, then the classification of the 5 nearest neighbors in the training data to the test data observation will be applied to that observation. One problem with using KNN for our dataset is that it assumes all predictors have the same effect on the response variable. We know this is not generally true; for example, lot size may impact sale price more than house condition. However, we can address this somewhat by removing correlated predictors and scaling numeric predictors.

Confusion matrix for k=38

```
##            pred
## obs          Cheap Expensive
##   Cheap        386        53
##   Expensive     66       225
```

```
## [1] 0.8369863
## [1] 0.1630137
```

Confusion matrix for k=39

```
##              pred
## obs           Cheap Expensive
##   Cheap         386        53
##   Expensive      67       224
```

```
## [1] 0.8356164
```

```
## [1] 0.1643836
```

To find the optimal K value, we took the square root of the number of observations in the training data set. Since the number of observations is 1460, the square root is 38.2 so we fit 2 models with k=38 and k=39. We then created a confusion matrix for both models using the training and test data set and found that k=38 yields a lower misclassification rate in the training set. Therefore, we append our results for K=38 into our table.

We use k-fold cross-validation to select the best number of neighbors. Since our data isn't significantly large, we use 5 folds for cross validation so that 20% of our data can be used for testing.

For each fold and each neighbor, we want the type of error (training/test) and the corresponding value.

```
# Best number of neighbors
# if there is a tie, pick larger number of neighbors for simpler model
numneighbor = max(val.error.means$neighbors)
numneighbor
```

```
## [1] 9
```

The best number of neighbors is found to be 9.

```
##              true
## predicted     Cheap Expensive
##   Cheap         405        59
##   Expensive      34       232
```

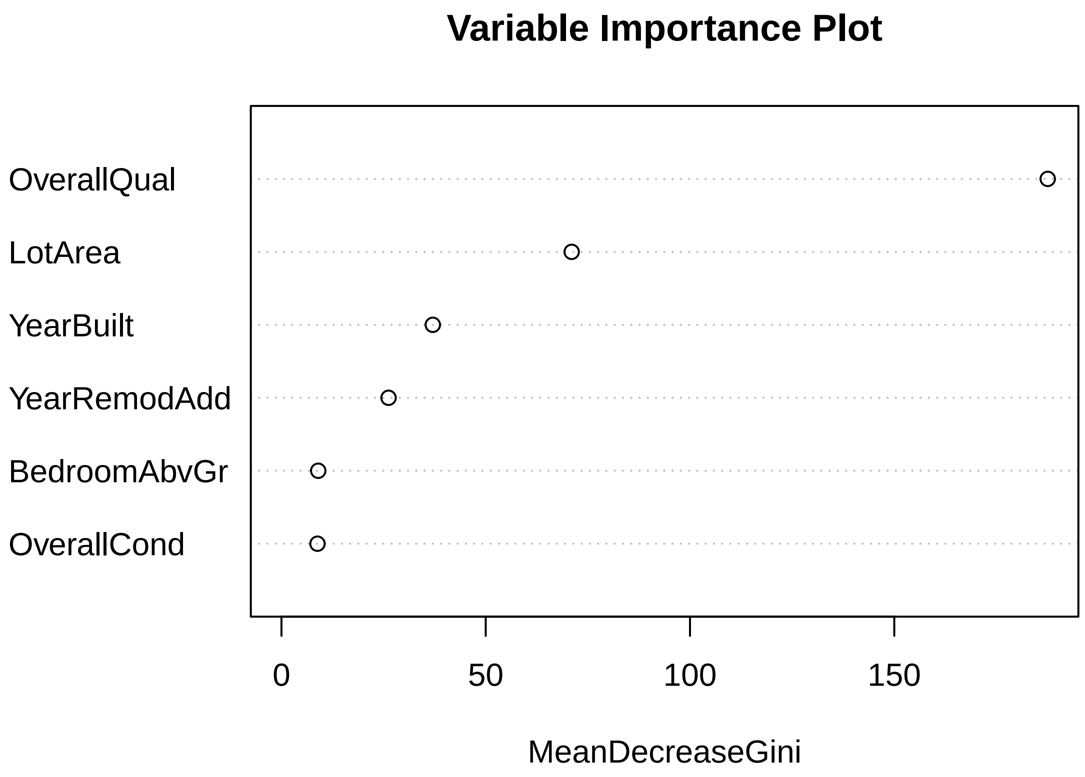
```
## [1] 0.8726027
```

```
## [1] 0.1273973
```

We then train a 9-NN classifier and calculate the test error rate to be 0.1273973, which is relatively low.

Random Forest

RandomForest is similar to the decision tree method in that it builds trees, hence the name 'random Forest'. This is a learning method which creates a lot of decision trees, and outputting the class that occurs most often. The advantage that randomForest has over decision trees is the element of randomness which guards against the pitfall of overfitting that decision trees run into on their own.



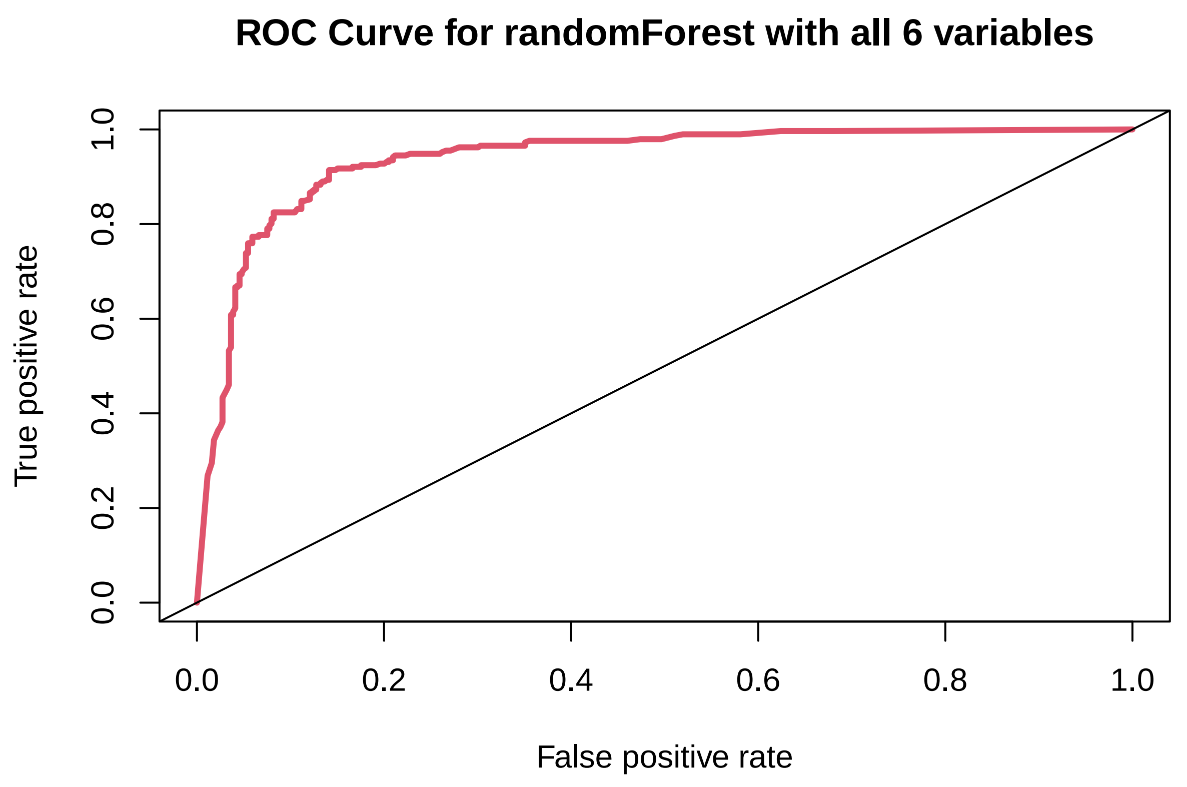
Impurity is a way that the optimal condition of a tree is determined and this plot shows how each variable individually affects the weighted impurity of the tree itself. It's consistent with our findings from the pruned decision tree method. OverallQual is by far the most important predictor, followed by LotArea and YearBuilt.

```
##            pred
## true        Cheap Expensive
##   Cheap       403        36
##   Expensive    53       238
```

```
## [1] 0.8780822
```

```
## [1] 0.1219178
```

With an accuracy of 0.8753, this randomForest model is our best classification method so far.



```
## [1] 0.9369936
```

After graphing the ROC curve, we find the area under the curve is 0.9363, which is the strongest AUC compared the previous classification methods.Therefore, we can conclude that randomForest is the most likely method to correctly classify sales based off the attributes.

Conclusion

```
compare
```

```
##                 Accuracy Rate       Error Rate          AUC
## tree            "0.854794520547945" "0.145205479452055" "0.923255759340582"
## pruned.tree     "0.856164383561644" "0.143835616438356" "0.907388707543699"
## k=38 kNN        "0.836986301369863" "0.163013698630137" "NA"
## k=15 kNN        "0.872602739726027" "0.127397260273973" "NA"
## randomForest "0.878082191780822" "0.121917808219178" "0.936993635958011"
```

Looking at our findings, we see that randomForest is the method that gets us both the highest accuracy rate and the highest AUC value (when compared to the decision tree method since we didn't calculate AUC for K-Nearest Neighbors). RandomForest is also not as prone to overfitting as decision trees since it is just an aggregate of many decision trees, and it is a more robust method than KNN for large datasets where predictors may be correlated. Therefore, we can conclude that RandomForest is the best classification algorithm.

There isn't a huge difference in accuracy rates between any of the methods, so each method may have it's own uses. For example, we have already stated that decision trees are prone to overfitting, but it is also the most easily interpreted method. We were able to prune the tree through K-fold cross validation which allowed us to narrow the model down to the most important variables. We could have also used the variables from the pruned tree to build a simpler randomForest model using only the "important" variables, since a simpler model gives reduces bias and complexity. It would be interesting see if making the randomForest model less complex would be worthwhile by comparing metrics. If we could apply this subset of variables to the randomForest algorithm and come out with a strong model that only utilizes a few independent variables in order to classify at a high success rate, it would lend strength to the argument that OverallQual, LotArea, YearBuilt are the most relevant predictors when it comes to determining whether a house is expensive or not.

As far as further questions we still have, we would want to know how best to balance having a high level of accuracy while also balancing variance and bias so we don't over/underfit. For example, is it necessary to drop variables from randomForest when the method already inherently accounts for overfitting? However if we just compare for models with the same number of variables, we conclude randomForest is the best for binary classification and our most important predictors are OverallQual, YearBuilt, and LotArea.

References

- https://www.kaggle.com/lespin/house-prices-dataset#test.csv