Predicting Income Report

Landon Kleinbrodt and Andrew de la Cruz

12/08/2017

In this report we will attempt to answer the question: can a classification model built on census data predict whether or not an individul's yearly income exceeds $50,000? Specifically, we will investigated the following classification methods: single tree, bagged forest, and random forest. The performance of these three models will be analyzed and compared to determine the best possible model for the data.

The code behind this report can be seen by examining the .Rmd version of this file, or by inspecting the Appendix.Rmd file included in this folder.

Data

The data set used in this project is the Census Income Data Set from the UCI Machine Learning Repository. This data set was donated by Ronny Kohavi and Barry Becker in 2001 and has been cited in over a dozen papers. The set contains ~49k observations of 14 attributes detailing different pieces of census information. Some are continuous, such as capital-gains and hours-per-week worked, while most are categorical, like education level, race, and relationship status. This data has already been split into a training (2/3) and validation set (1/3). A small subsection of the data is shown below.

```
##   Age  Education         Occupation   Relationship   Race     Sex Salary
## 1  39  Bachelors       Adm-clerical  Not-in-family  White    Male  <=50K
## 2  50  Bachelors    Exec-managerial        Husband  White    Male  <=50K
## 3  38    HS-grad  Handlers-cleaners  Not-in-family  White    Male  <=50K
## 4  53       11th  Handlers-cleaners        Husband  Black    Male  <=50K
## 5  28  Bachelors     Prof-specialty           Wife  Black  Female  <=50K
## 6  37    Masters    Exec-managerial           Wife  White  Female  <=50K
```

Overview

We will begin with a brief discussion of the data loading and pre processing, followed by an exploratory data analysis. Once finished, we move on to fitting and tuning the three types of models. Finally, we will compare the results of these models - specifically using confusion matrices, accuracy rates, and ROC curves. Finally we will choose our best model and validate it on the included test data, as well as compare its predictive test power to the other models.

Exploratory Data Analysis

Loading and Preprocessing

Loading in the data from the UCI repository was straightforward, but once loaded it required some pre-procesing. For example, all the categorical variable values began with an empty space, and the `Salary` column of the test data contained an extra `.` in all its entries. Also, the data contained ~6500 total missing values, denoted by the character `?`. The data are supplied as two files (train and test), which we concatanated so that all preprocessing steps were applied equally, and splitting them up again before model training.

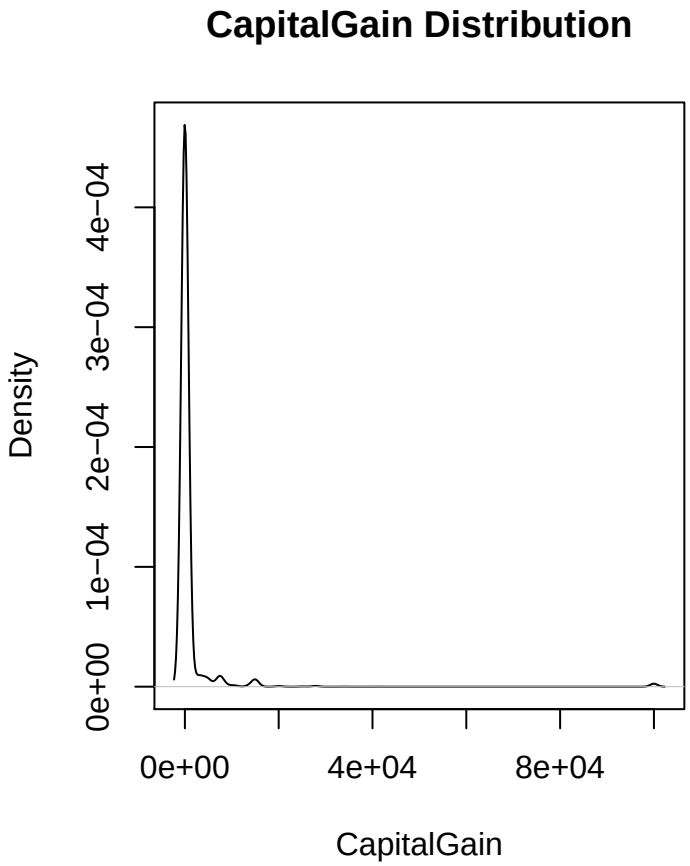
Missing Values

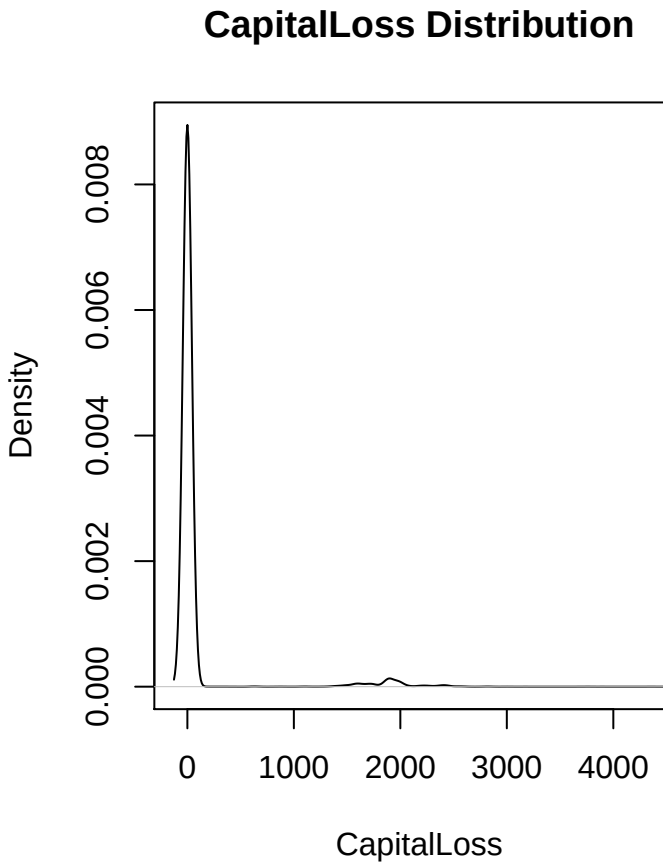
The majority of the missing values come from the `Occupation` and `WorkClass` columns. In fact, there is almost a complete overlap betwen those without information on occupation and those without information on working class, which makes sense as the two categories are quite related. For the most part, the distribution of the other statistics did not vary greatly between those with missing values and those without. Since there were relatively few individuals with missing values, we were first tempted to simply drop those observations with any missing data. However, analysis shows that 90% of people with missing values make less than $50k as ompared to the base rate of 75% in the general population. It would be misrepresentative to rely on imputation to fill in these missing values, especially when we see such a sharp difference in salary distribution. Therefore, we chose to encode missing `Occupation` values as 'Unknown', this allows us to keep the information gained from an unknown occupation (which we see relates to salary) without misrepresenting the individual's status.

The only other variable with missing values is `NativeCountry`, with less than a thousand unknowns. Inspecting this subset reveals that they follow the same distribution as the general population, so it does not appear that excluding this data will bias our models against any specific subset. Thus, since these individuals seem to represent a very small, random subset of the general population we choose to exclude them from the model building process.

Outliers and Binning

There were two considerations to make in regards to binning. First, the `CapitalGain` and `CapitalLoss` columns are extremely skewed.





First, we noticed that `CapitalGain` and `CapitalLoss` are two sides of the same coin (if CapitalGain>0 then CapitalLoss = 0 and vice versa), and can be combined. This creates a variable `CapitalChange` which is again very skewed: 5% have less than 0, 87% of people have exactly 0, 8% experienced some gain, and .5% experienced the max value. This skewed data seemed ideal for binning, as it was clear those individuals with max `CapitalGain` behaved very differently than those with 0. However, the entirety of this report was run by binning into a single column (into those above specified categories) and by leaving the two columns as they are, and better performance was achieved without binning.

Furthermore, we can see that `NativeCountry` has over 40 levels, which exceeds the maximum number of many classification models. Thus, we binned this column into continents/regions: Asia, North America, South America/Caribbean, and Europe

Changing Scales

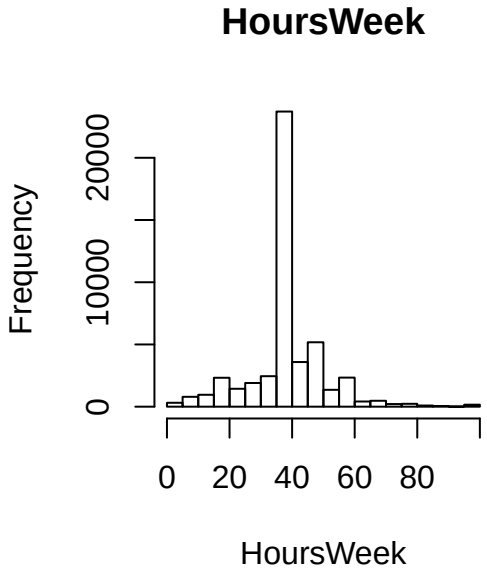
While most of the data is categorical, the various numerical variables are on widely different scales. `Age` varies between 17 and 99, `EducationNum` varies between 1 and 16, `HoursWeek` between 1 and 99, and `Capital` gains and losses can be anywhere from 0 to 99,999. This report focuses on tree-based classifers, which are scale invariant, but we chose to center and scale anyways so that other methods of regression/classification could be easily integrated if need be.

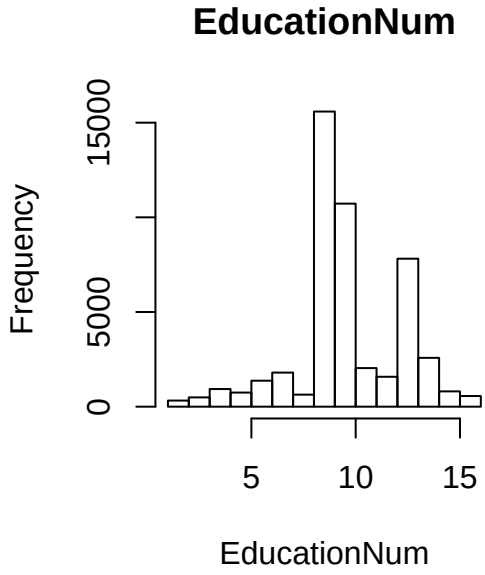
Dummy Indicators

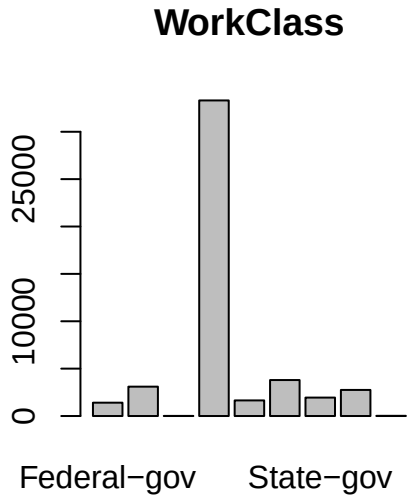
Tree based classification methods can easily handle categorical variables, and so one-hot-encoding is unecessary. Furthermore, since there are so many categorical variables with so many levels, dummifying them one would greatly increase the number of predictor columns in our data matrix.

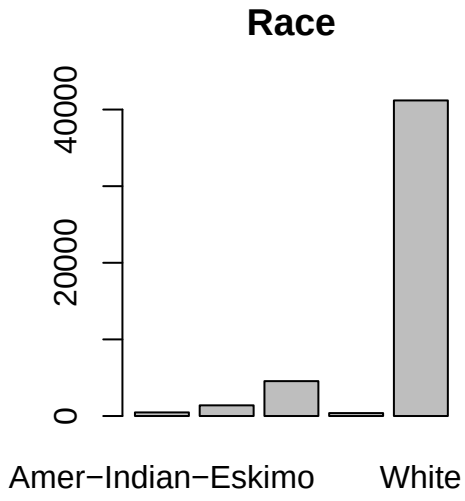
Summary Statistics and Distributions

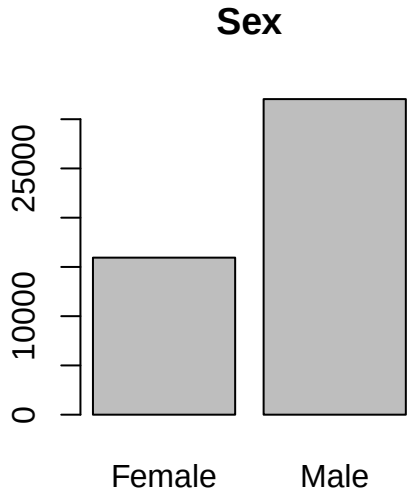
The summary statistics before and after transformation, as well as the various correlations between predictors were explored and analyzed (figures and code can be found in the appendix). A few points are of note. First, many of the categorical are skewed around one mode value. Often this phenomenon likely reflects the trends of the general population: the vast majority of people are Private `WorkClass` and most have either highschool or some college education. Other skews seem to be more results of sampling bias, for example over 85% of participants are white, and over two thirds are male. While it is important to keep these skews in mind, they did not appear to overly interfere with the predictive power of our models.











Note that there are two variables representing an individuals education: the categorical `Education` and the numerical `EducationNum`. These variables are interchangeable, thus we ran this entire analysis using one and then repeated the process using the other. Superior results were obtained by using `EducationNum` while excluding `Education`

2) Model Building

After pre-processing and initial exploration, the data was split back into the training and validation sets as provided initially (a 2/3-1/3 split). Furthermore, since we have enough observations, we split the training

data again into train (80%) and test (20%) groups. This way, we can use the first test group as a means of comparing the predictive power of our models out of bag, and then we can use the holdout validation set to get a truly indicative performance measure of our final model.

We also specify that those with >$50k are classified as "Yes" and those with less are classified as "No".

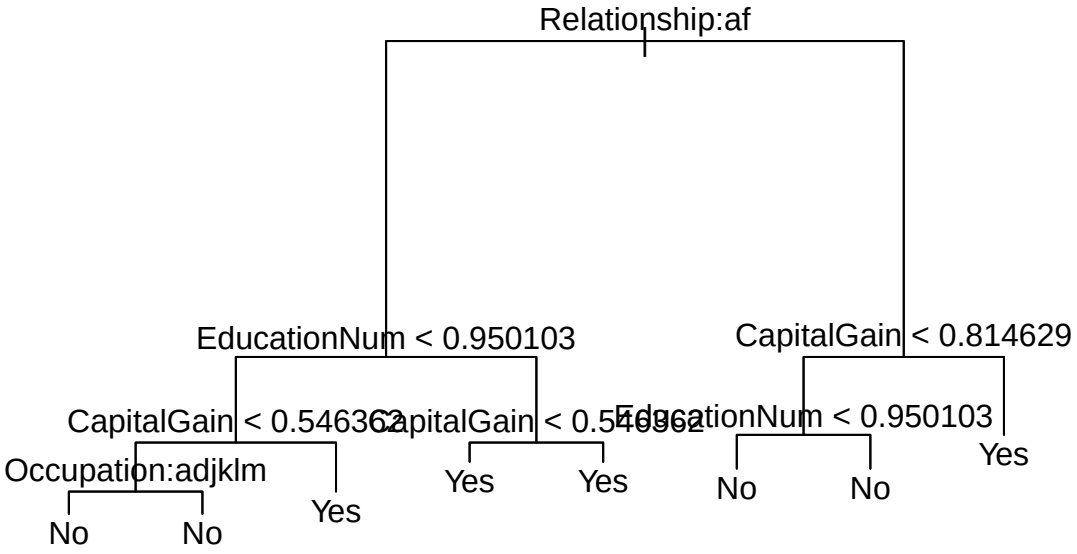
For each of the performed methods, you can include a brief description of what each method does (e.g. basic working principles, key ideas, goal), the functions and packages used to carry out the model building process, the main results (tables, summaries, graphs) and their corresponding descriptions and interpretations.

Fit a classification tree (see examples in ISL chapter 8, and APM chapter 14). Make plots and describe the steps you took to justify choosing optimal tuning parameters. Report your 5 (or 6 or 7) important features (could be either just 5, or 6 or 7), with their variable importance statistics. Report the training accuracy rate. Plot the ROC curve, and report its area under the curve (AUC) statistic.

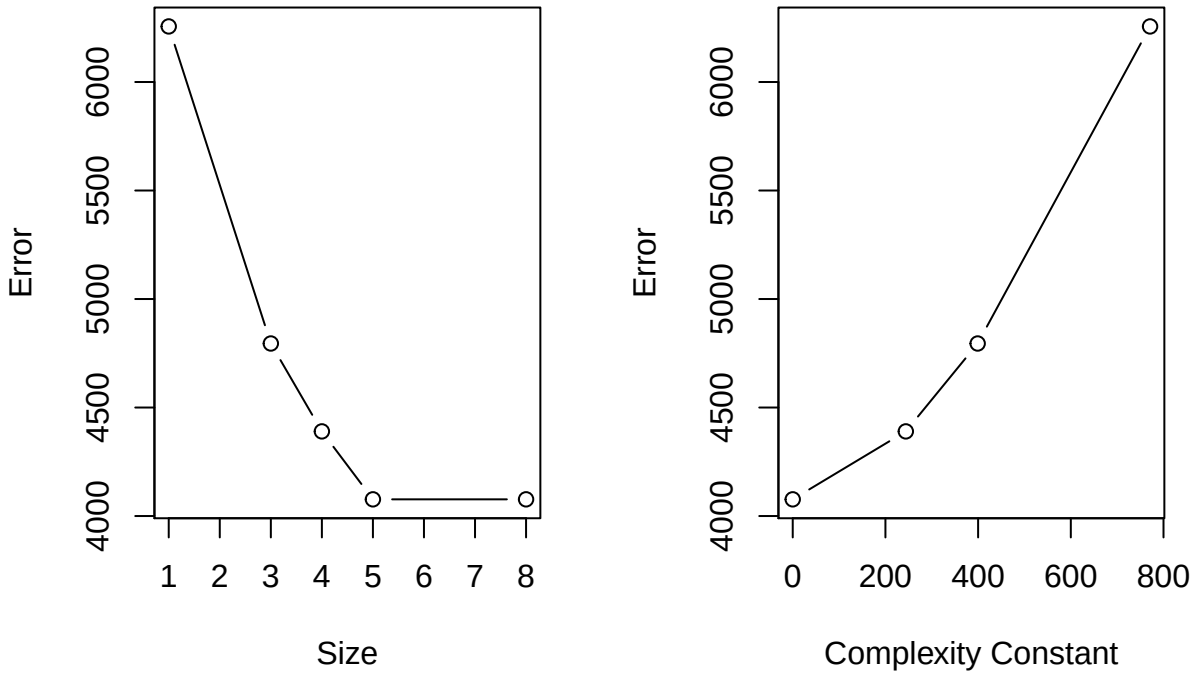
Classification Tree

The classification tree is one of the most intuitive and easily readable forms of classification. A tree is a combination of nodes, where each node has a splitting criterion that results in dividing its input group into two output groups based off their values in some variable. At the bottom of the tree are the terminal nodes, the proportions of classes in those terminal nodes decides what the overall classification is for a new individual that ends up in that terminal node. While there are many methods to determine which split is best, the overall goal is to reduce the entropy with each split - to have each new group be more homogenous than before the split.

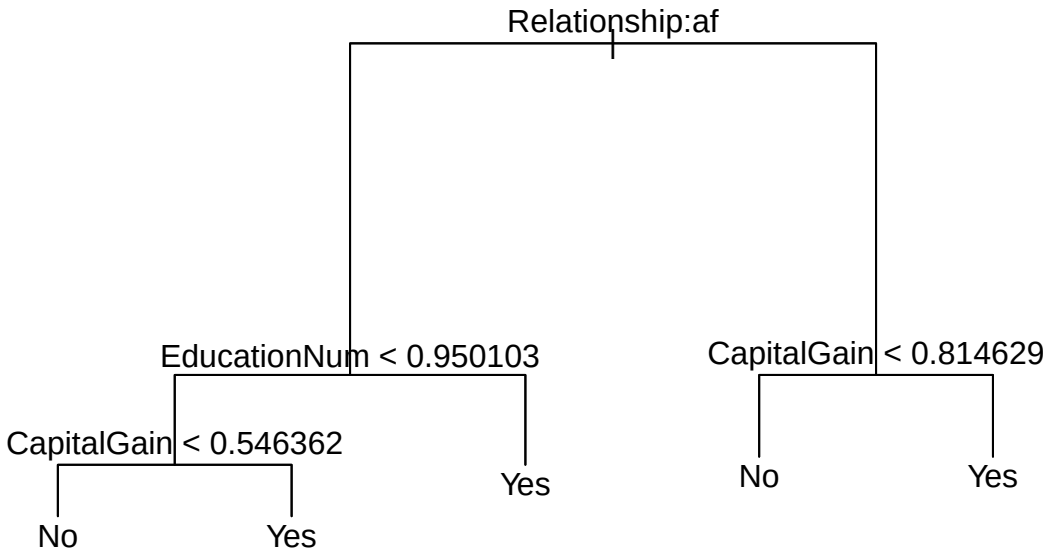
There are many variations and approaches to building a classification tree, this report explored two of them. First, we used the `tree()` function from the package `tree` to build a base tree (pictured below)



Notice how some of the splits in this tree are redundant, the final split on the far left and far right both result in the same classification. This encourages us to use cross validation to find more optimal parameters. Specifically, we used `cv.tree()` to determine the optimal parameters for tree size and for tree complexity.



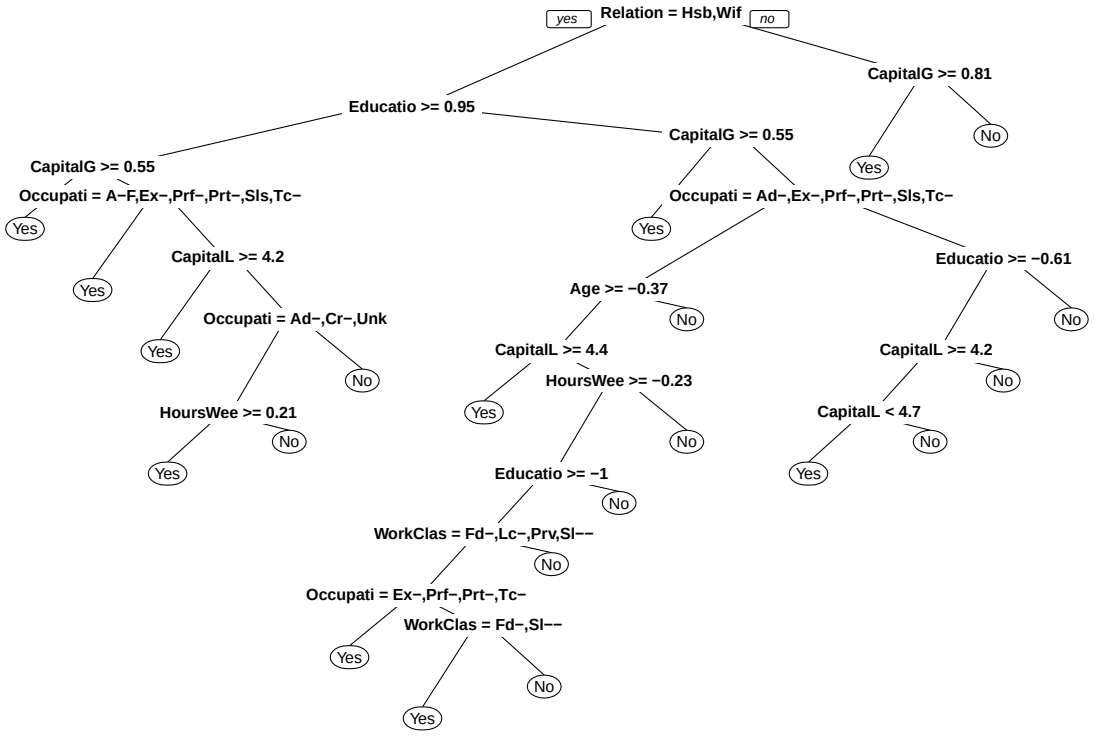
From these plots we can see that error stops decreasing with a tree size of 5, and that error is minimized for small complexity. To be thorough, we pruned our original tree using `prune.misclass`, once with a desired size of 5 and once with a size of 8. Comparing these two reveals that the tree of size 5 performs equally well and is considerably simpler. Thus we see the final result from our `tree()` construction below:



We can see that this tree splits on 3 variables. First it splits based on relationship status, and then by education or capital gain, and then again by capital gain if necessary. Immediately we can see that a low `CapitalGain` is associated with low income as we would expect.

`tree()` is a relatively simple function, and we can obtain more complex results and interpretation using the `rpart` package, which we will do now.

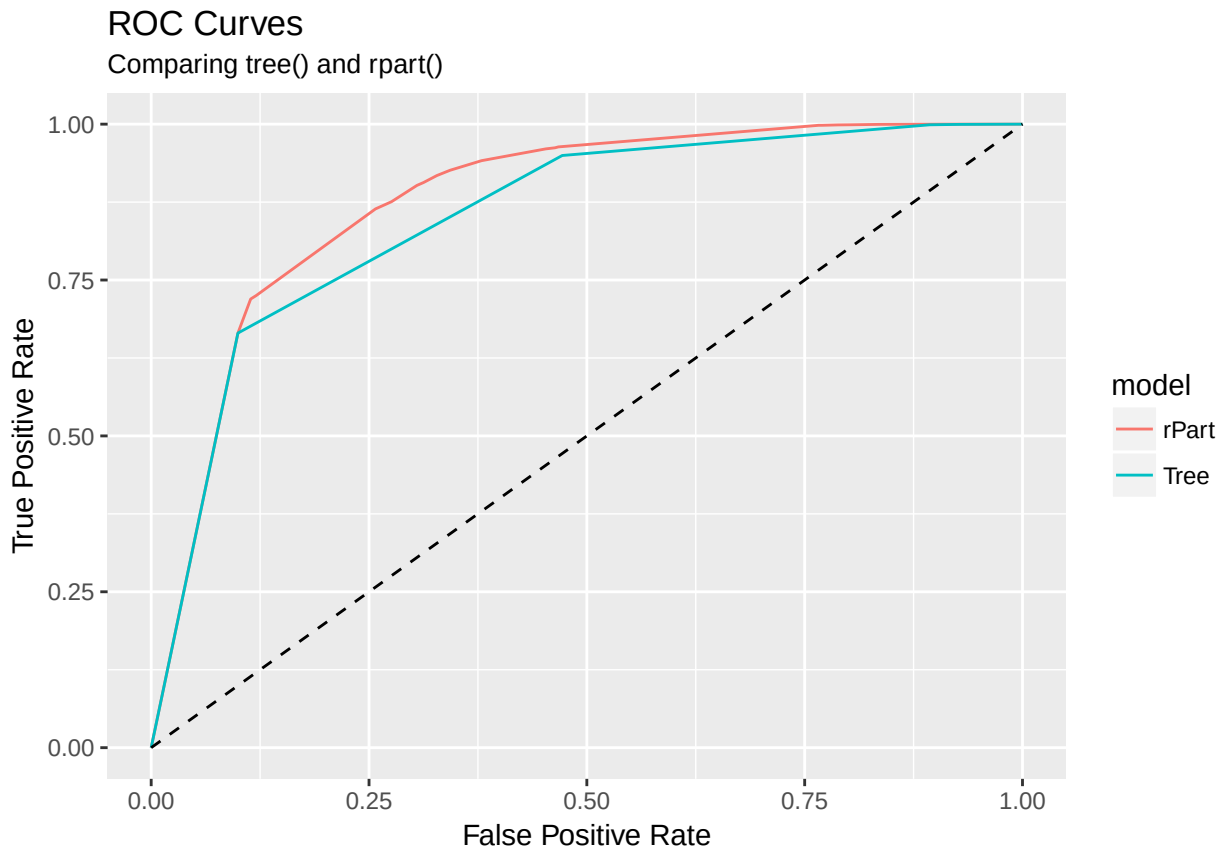
the `rpart` package allows us to train our model using the `train()` function from the `caret` package, by setting `method = 'rpart'`. Using cross validation we tuned the complexity parameter and depth of the tree to result in a depth of 23 and a complexity of ~.002. We then used `rpart` with the specified tuning parameters to arrive at the final tree:



Immediately it is apparant how much more complex this tree is than the basic one produced by `tree()`. We also see that it includes significantly more variables than previously. We can use `varImp()` of the `caret` package to determine the relative importance of each variable based upon the reduction in the loss function attributed to each variable.

```
##             var  importance
## 1    CapitalGain 2776.317117
## 2     Occupation 2339.376170
## 3   EducationNum 2317.692838
## 4   Relationship 1898.109623
## 5  MaritalStatus 1867.991922
## 6    CapitalLoss  723.725550
## 7      HoursWeek  504.685870
## 8            Age  470.070349
## 9      WorkClass  126.308089
## 10        FNLWGT    7.726556
## 11          Race    0.000000
## 12           Sex    0.000000
## 13 NativeCountry    0.000000
```

Now we will examine the differences in performance:




```
##                  Tree     rPart
## Accuracy    0.8476892 0.8638108
## Sensitivity 0.5282181 0.6207990
## Specificity 0.9497569 0.9414506
## AUC         0.8509806 0.8778790
```

Despite the vast differences in complexity, the trees created by `rpart()` and by `tree()` have very similar performance metrics. We can see that their ROC curves are essentially the same, with rPart performing slightly better, resulting in a ~.03 higher AUC. Similarly, rpart produces a slightly higher accuracy rating of .87 compared to .85. The most important difference though, is that rpart produces a significantly higher sensitivity while only perforing very slightly worse in specificity. Thus, these metrics lead us to conclude that the tree generated by rpart has a higher predictive power. It is important to note that the tree created by `tree()` is **much** more readable and easily interpretable than rpart's, but this reports focuses on predictive power and thus rpart produces the superior model.

Bagged Forest

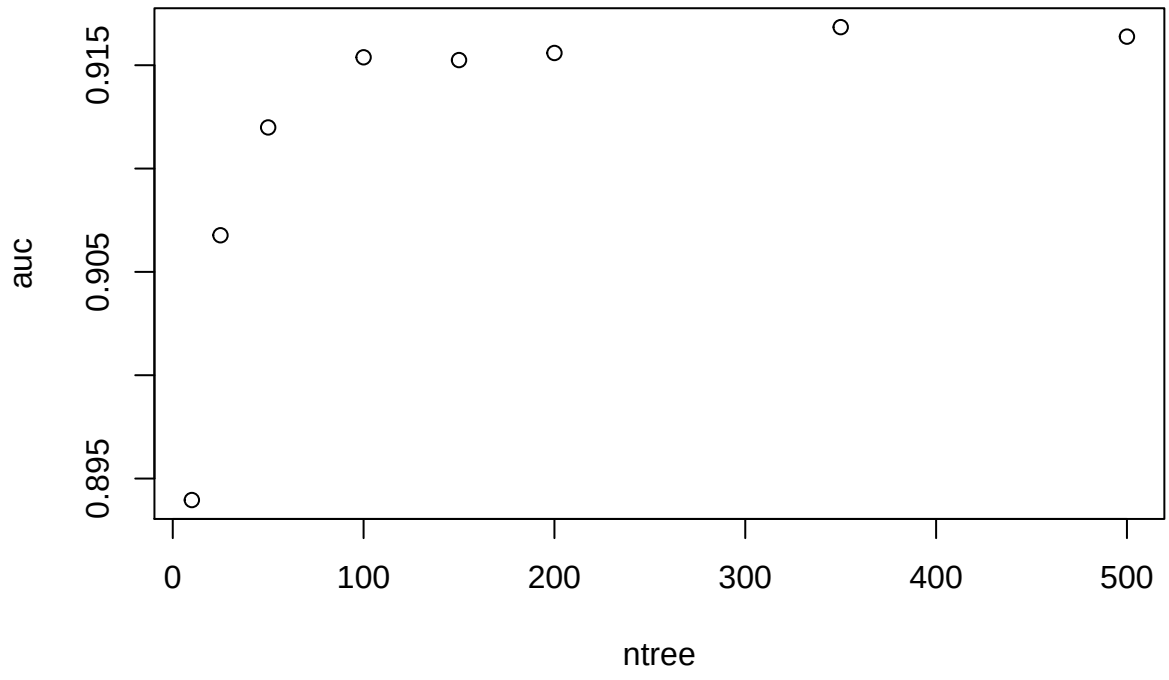
Bagging is an ensemble extension of classification trees. First, we take many random sub-samples of the data (with-replacement). For each of those sub-samples we then train a classification tree. Then, to classify a new observation, we input that observation into each tree in the forest, and each tree votes for the class that it predicts that observation to be in. Those votes are tallied over the entire forest, and the observation is assigned whichever class has the most votes.

Bagging is used to reduce variance, which is one of the weaknesses of decision trees. Trees are especially sensitive to the data they are trained on, and so training hundreds of trees on different subsets of data helps to lower the bias.

To do this we will use the `randomForest()` function from the `randomForest` package. As we will see soon, bagged trees are a certain generalization of random forests and vice versa.

The primary parameter to tune with bagging is the number of trees to include. We will now investigate how adding trees affects the performance (AUC) of the model.

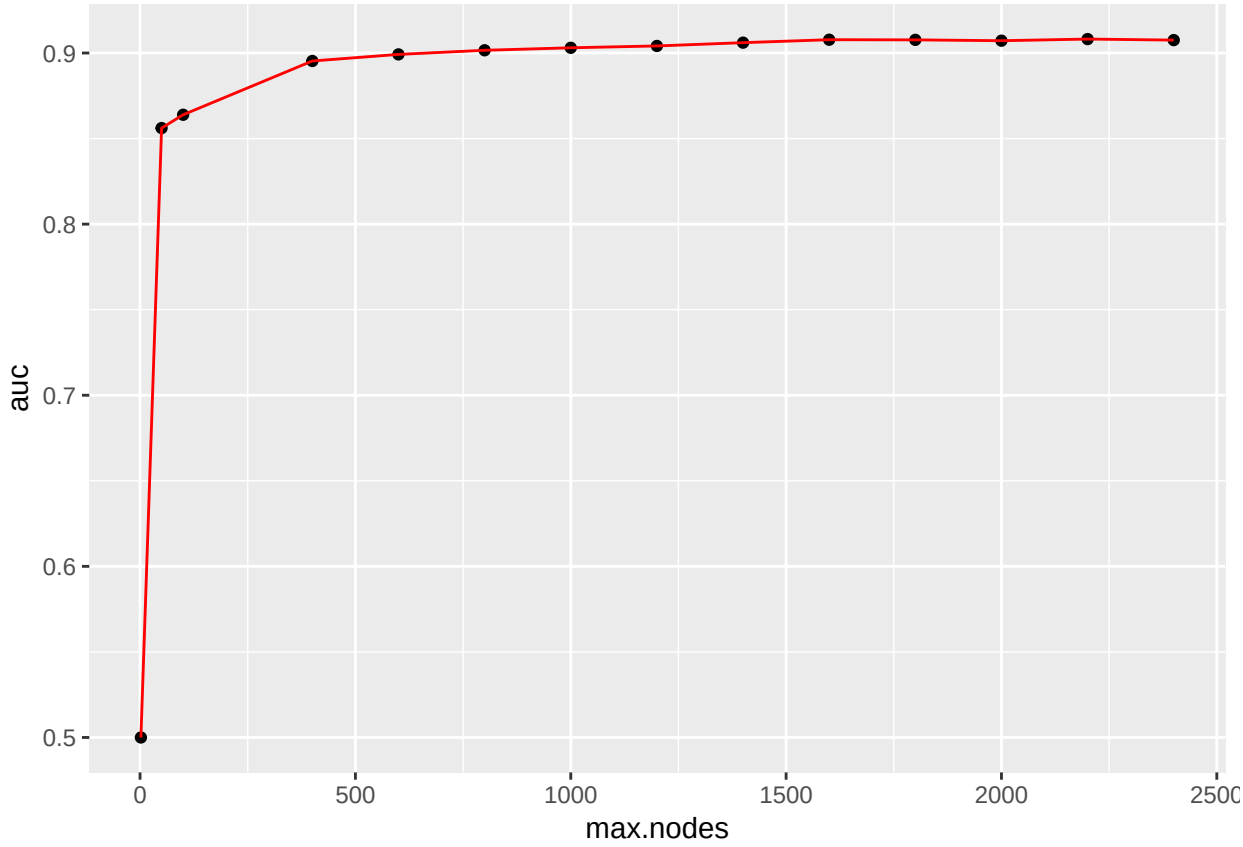
```
all.trees = c(10,25,50,100,150, 200, 350, 500)
results = data.frame(ntree = all.trees, auc = 0)
for (i in 1:length(all.trees)){
  n = all.trees[i]
  forest = randomForest(Salary~.,
                        data = Train,
                        importance = F,
                        ntree = n)

  posteriors = predict(forest, newdata = test, type = 'prob')
  predic = predict(forest, newdata = test, type = 'class')
  df = cbind(posteriors[,2], test$Salary)
  preds = prediction(df[,1], df[,2])
  ROC = performance(preds, measure = 'tpr', x.measure = 'fpr')
  auc <-performance(preds, measure = 'auc')
  results[i,'auc'] = auc@y.values[[1]]
}
plot(results)
```



It is clear that there is diminishing returns for adding new trees to our model. Adding more trees helps us to lower the variance of our model, but it there is a point in which the gains we get from decreased variance are

outweighed by complexity and computing time. From this plot we can see that a 100 tree forest will give us essentially the same results as a 500 tree forest. So, for simplicity and to reduce the complexity of our model we will use 100 trees for the following analysis.

Typically, forests such as these are grown as deeply as possible. Since we are aggregating many trees, we are less concerned with any individual tree overfitting the data, and thus there is generally no pruning of such forest methods. Now, in order to confirm this decision we will tune our model by cross validation at several different `maxnode` values. This will limit the size of each tree in our forest, and we will plot our results.




```
##   max.nodes       auc
## 1      2200 0.9081374
## 2      1600 0.9077686
## 3      1800 0.9076805
## 4      2400 0.9075305
## 5      2000 0.9071799
## 6      1400 0.9060444
```

From this graph we can see that at first, increasing the size of our trees *greatly* improves accuracy (AUC), but that quickly begins to level off. Still though, we can see that allowing our trees to grow larger continues to improve model accuracy, which confirms the methodology behind not pruning forests. However, in this case we achieved optimal AUC at a maxnode size of 2200 so we will use this max depth for our final bagged tree.

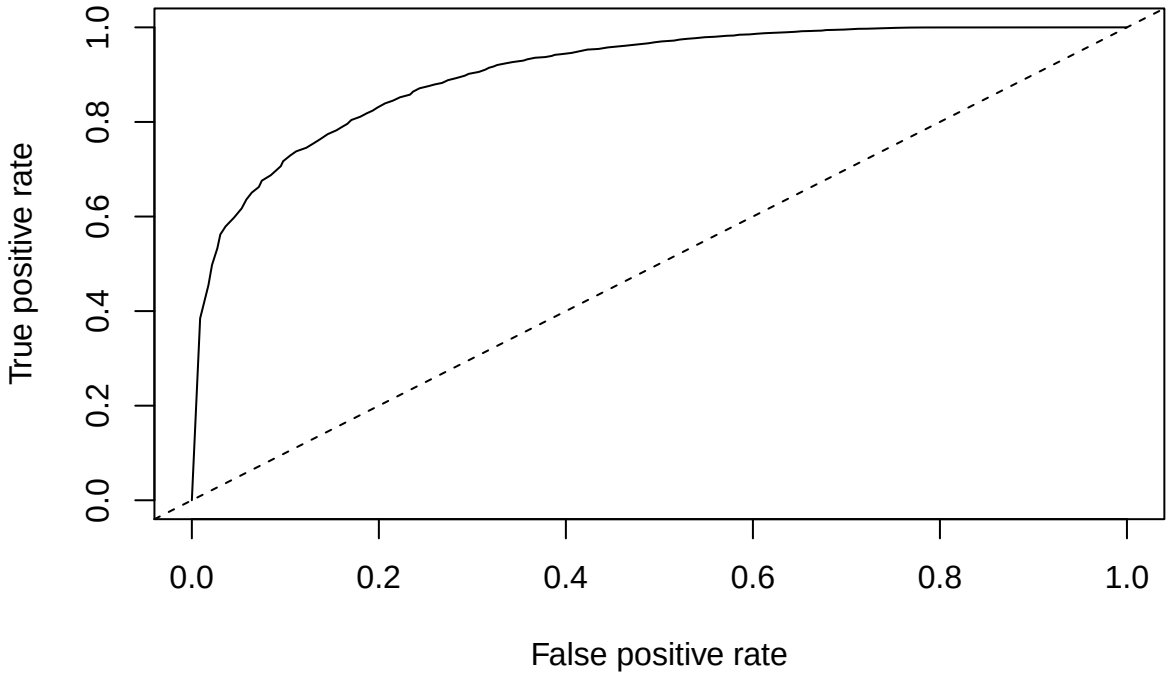
Confusion Matrix and Accuracy

```
##            Reference
## Prediction  Yes   No
##        Yes 1027  550
```

```
##        No   355 4581
```

```
##    Accuracy Sensitivity Specificity
##   0.8610471   0.7431259   0.8928084
```

ROC



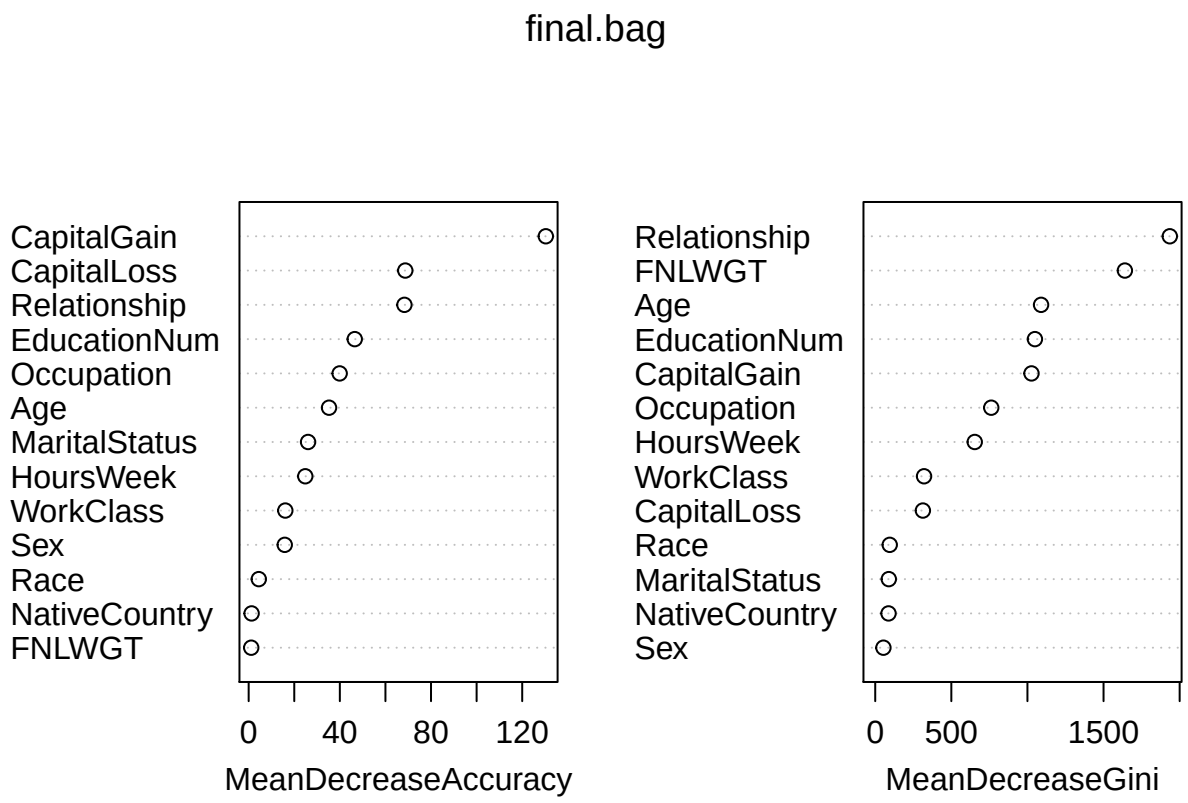

```
##       AUC
## 0.9070605
```

Immediately we can see that our bagged approach has performed better than a single classification tree. The new model has higher accuracy, higher sensititivty, much higher AUC, and a slightly lower specificity. While specific model goals may change the decisionmaking process, a bagged approach seems to have created a more powerful predictive model than a simple classification tree.

Variable Importance

```
##                     Yes        No MeanDecreaseAccuracy MeanDecreaseGini
## Age           48.8737068  1.767120            35.262378       1089.48881
## WorkClass      6.8059336 14.803438            16.070870        320.62619
## FNLWGT        -0.1970847  1.547255             1.129154       1640.09952
## EducationNum  46.2734220 20.718157            46.536000       1049.03035
## MaritalStatus -2.3971091 27.299432            26.064241         89.08687
## Occupation    40.7968158 12.320364            39.919653        762.19145
## Relationship  30.8715548 37.512454            68.330631       1935.34127
## Race           3.1998097  2.729762             4.445083         95.14410
```

```
## Sex            -0.9549325 17.291689          15.823321        53.74402
## CapitalGain   105.8075724 99.954496         130.359199      1026.65136
## CapitalLoss    67.2761950 36.091984          68.708139       312.92871
## HoursWeek      27.4684257  6.135537          24.840474       653.70304
## NativeCountry  -0.8454149  2.068074           1.283589        87.25684
```



The most important variables are `CapitalGain`, `Relationship`, `Education`, `Age`, and `Occupation`. These are the same top five variables as our rpart tree, except that rpart used `MaritalStutus` instead of `Age`. One thing to notice here is that `FNLWGT` has a high Mean Decrease in Gini, but almost no Mean Decrease in Accuracy. This is likely because `FNLWGT` represents an imputed number representing how well reprsented that individual is in terms of how similar their census data is to those of other individuals. So, `FNLWGT` can be thought of as a numerical indicator of how common/represented an individual is. Thus, splitting on `FNLWGT` would help us to separate individuals into more pure nodes (splitting up individuals in terms of their 'normalcy'), however it does not help us with prediction, since a person's representation/uniqueness in demographic data does not necessarily correlate with their income. Thus, `FNLWGT` helps us to separate our individuals, but not to predict their income. In the future, `FNLWGT` can probably be excluded from model building since it is not a true census data point but rather an imputed measure of similarity, and it does not help our predictive models.

Random Forest

Random forests are an extension of bagged bagging. The two follow almost exactly the same process, and have very similar goals (reducing variance). However, whereas bagging uses *all* predictors when creating each tree, a random forest takes a random subset of the predictors to be used in each tree. The number (or fraction) of predictors to use is a parameter unique to random forest: `mtry`. Again, while random forests are

generally not pruned, we will confirm that choice here by tuning for both tree depth (through max number of nodes) and for mtry (number of predictors used to build each tree)

```
##   max.nodes nvar       auc
## 1      2400    4 0.9130498
## 2      2400    6 0.9113606
## 3      2000    6 0.9112721
## 4      2000    4 0.9106136
## 5      2400    8 0.9104300
## 6      2000    8 0.9090342
```

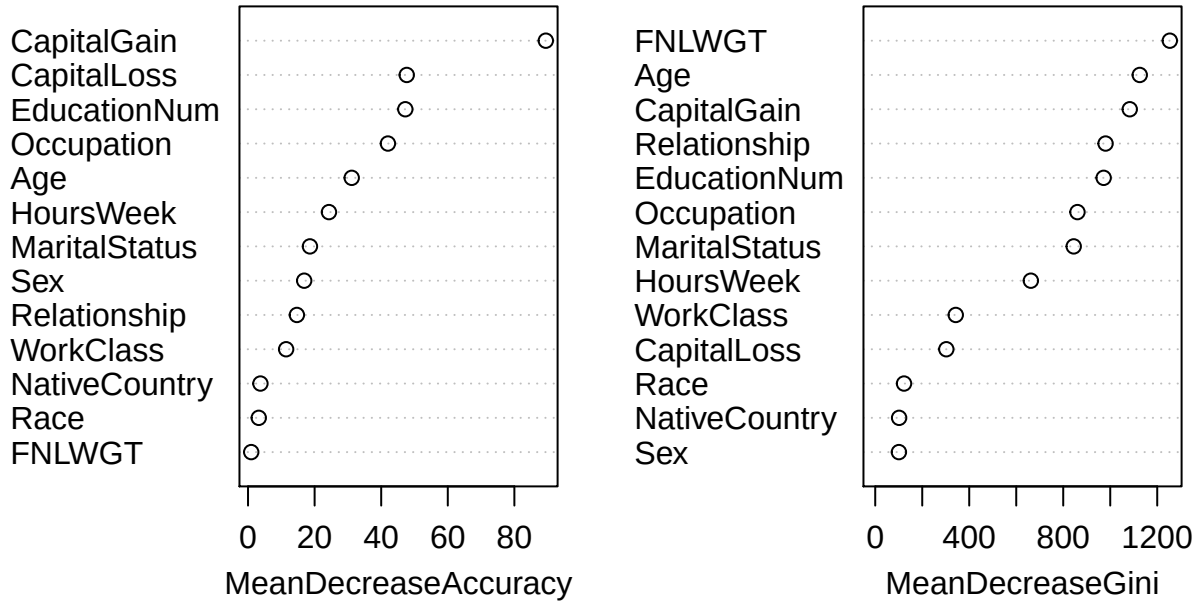
We see that we get our best results were obtained when using the highest maxnode size we tested for. So here we will follow the general convention in our final model and allow our forest's trees to grow without pruning. Furthermore, we see that our best results were obtained when each tree was built using only 4 predictors.

Now that we have obtained our optimal parameters, we can train our final Random Forest

```
##           Reference
## Prediction  Yes   No
##        Yes 1029  548
##        No   331 4605
```

```
##    Accuracy Sensitivity Specificity
##   0.8650392   0.7566176   0.8936542
```

```
##                      Yes          No MeanDecreaseAccuracy MeanDecreaseGini
## Age            43.843453 -3.62139283           31.1424180        1125.5993
## WorkClass       7.828699  8.31517826           11.4427747         342.6251
## FNLWGT         -1.201966  2.09611093            0.9631398        1253.5325
## EducationNum   47.474469 18.42679712           47.2393114         971.9595
## MaritalStatus   9.505562 22.82203509           18.5935545         844.5884
## Occupation     51.619253  5.14325330           42.0194363         860.5441
## Relationship   16.127184 10.82893281           14.6751155         980.1759
## Race            3.692252  1.37745561            3.2300181         122.5254
## Sex             2.847215 12.41167222           16.8667872         100.7605
## CapitalGain   106.719572 63.71128533           89.4436998        1083.1094
## CapitalLoss    44.368263 27.65509707           47.6759897         302.4795
## HoursWeek      34.610571  0.07208193           24.3086367         662.1521
## NativeCountry   3.552499  1.69412283            3.7210400         101.5968
```



We see many of the same important variables as before: `CapitalGain`, `CapitalLoss`, `Education`, `Relationship`, `Education`, `Occupation`, and `Age`

Model Selection

Training Results:

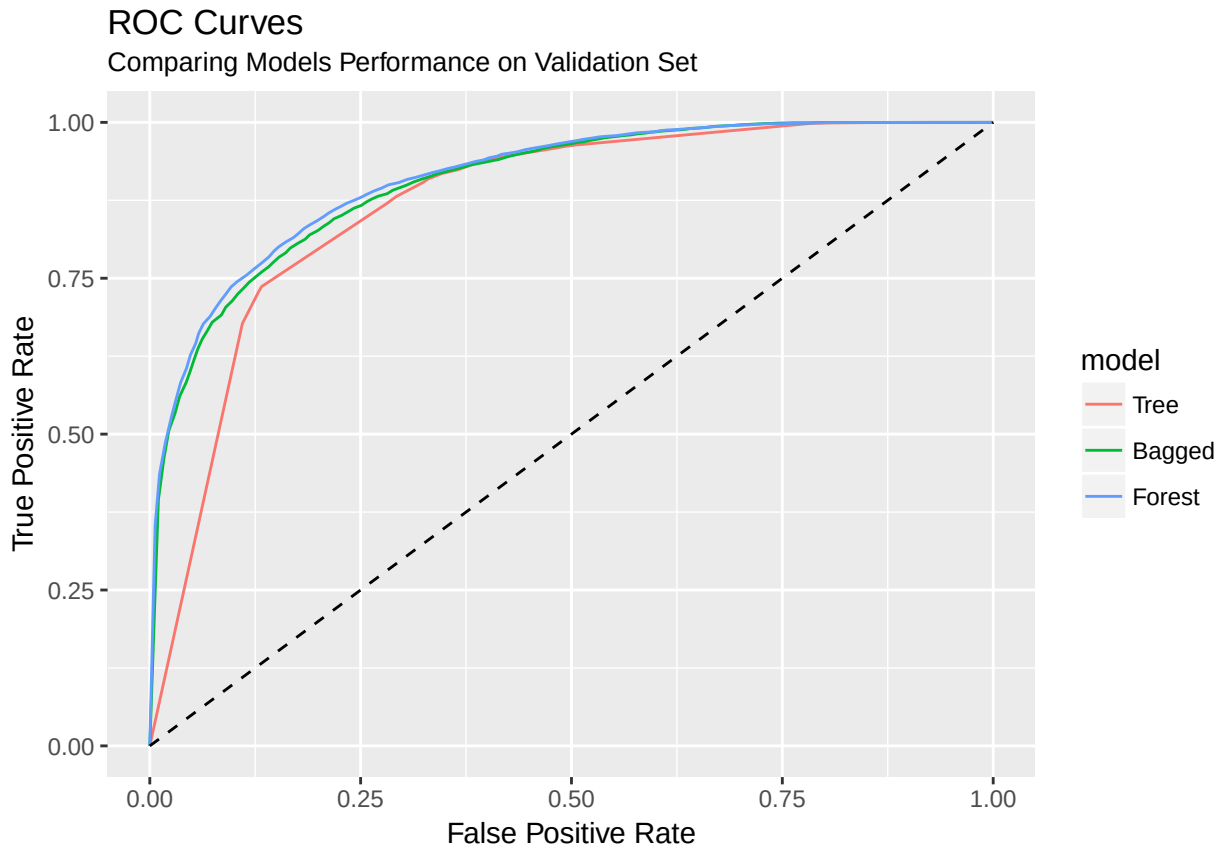
```
##                  Tree    Bagged    Forest
## Accuracy    0.8638108 0.8610471 0.8650392
## Sensitivity 0.6207990 0.7431259 0.7566176
## Specificity 0.9414506 0.8928084 0.8936542
## AUC         0.8778790 0.9070605 0.9075277
```

Validation Results:

```
##                  Tree    Bagged    Forest
## Accuracy    0.8603475 0.8584025 0.8609310
## Sensitivity 0.7571230 0.7341731 0.7479187
## Specificity 0.8840268 0.8896009 0.8882538
## AUC         0.8705177 0.9047038 0.9110398
```

From these summary results, we can see that while a randomForest is not always uniformly the best model (Tree sometimes outperforms in overall accuracy or in sensitivity) but Forest is the best model in terms of

AUC for both the training data and the validation data. Visualizing the ROC curves for the validation data we can see how the Random Forest edges out the competition:



We can see the bagged trees and random forest models performed very similarly. This is as expected, since the two use very similar methods, but the random forest model uses some predictors for each tree while the bagged approach uses all of them. Although their performance metrics are almost identical, the random forest model performs very slightly better in each category, and thus we select the random forest model as our best classification model.

Here are the performance metrics for that model:

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   Yes    No
##        Yes  2246  1388
##        No    757 11033
##
##                Accuracy : 0.8609
##                  95% CI : (0.8554, 0.8664)
##     No Information Rate : 0.8053
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5892
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.7479
##             Specificity : 0.8883
```

```
##          Pos Pred Value : 0.6181
##          Neg Pred Value : 0.9358
##              Prevalence : 0.1947
##          Detection Rate : 0.1456
##    Detection Prevalence : 0.2356
##       Balanced Accuracy : 0.8181
##
##        'Positive' Class : Yes
##
```

Treating the class "over $50k a year" as the positive class, we can calculate the True Positive Rate (Sensitivity) and True Negative Rate (Specificity) manually and see that they match up with the above confusion matrix:

```
## Sensitivity Specificity
##   0.7479187   0.8882538
```

Conclusions and Further Steps

We began this report with the goal of classifying whether or not an individual earns more than $50,000 a year. After constructing several classification models and comparing their performance statistics on both the training data and a witheld validation set, we determined that a random forest model with tuned parameters was the best model for this prediction task. We were able to attain an overall accuracy percentage of over 85%, and the AUC statistic for this model was above .9. These are relatively high performance metrics, and lead us to conclude that yes, the random forest model is able to accurately predict whether or not an individual earns more than 50,000 a year. Note that this is purely an exercise in model building, the decision of whether or not a model is "good enough" depends on the objective of the model. For example, there are discrepencies between teh sensitivity and specificity of the model, in part due to the uneven distribution of people above and below 50k. A different probability threshold might be set depending on which type of error is worse for a given objective.

It should be noted that the same 5 to 6 variables were the most important in every model that we created. Similarly, several variables (such as `Race`, `Sex`, and `NativeCountry`) were consistently ranked very low in importance, and thus did not contribute much to the performance of the model. Further work may wish to exclude these variables to reduce complexity. Similarly, we discovered that while `FNLWGT` consistently ranked high for Mean Decrease in Gini, it rarely affected accuracy. As discussed earlier, `FNLWGT` is more a measure of similarity than it is a true census predictor; further work may also wish to exlude this variable as well.

Next Steps

We were very surprised to find that `NativeCountry` was not particularly helpful in this process. Initially, we anticipated that individuals in certain regions (such as the United States) would likely earn more on average than those in others (such as South America), and thus country would be a helpful predictor. However this was not the case. It is possible that this discrepency is due to the fact that the vast majority of our individuals resided in North America, and thus our trees did not have enough variability in the data to create useful prediction splits. Further work may wish to sample international individuals at a higher rate to explore this relationship more.

Similarly, the data was very imbalanced in regards to `Race` and `Sex`. There were more than twice as many males in this data set as females, and over 85% of individuals were White. This skewed census data can interfere with our model building **and** can create more problems when we attempt to generalize this model to a population that has different demographics than the one it was trained on. A more balanced data set would allow our model to be more generally applicable and less biased.

After completing this report, our team was curious to see how our models would have changed were we to build them using data from a different time period. For example, we are very interested to see if the same variables

(`CapitalGain`, `Education`, etc) were equally important in the past, or if the factors affecting/correlated with one's income have changed. This data set was collected in 1994; we would like to perform a similar report (or perhaps a more inference-based investigation) on data from an earlier decade and data from a later decade. Comparing those results may give insight into the changing landscape of income.