Daniel Gomes

John Gomes

Blake Simmons

CIS 490: Sectional Project II Report

**1) X and Y, Regression or Classification Trees**

**X and Y**

In the Adults dataset, we had the “If income is greater than or less than 50k” attribute serve as our Y, we renamed the column as “class”. All other 14 attributes served as our X variable.

**Regression Trees or Classification Trees**

We found that it was appropriate to use a classification tree on the adult’s dataset. The reason for this is because the dataset is trying to predict whether income >= 50K or < 50K based on all other 14 features. In the end, our Y is a discrete value. Our terminal nodes would either be >= 50K or < 50K. This would not apply to regression trees as they look for continuous outcomes.

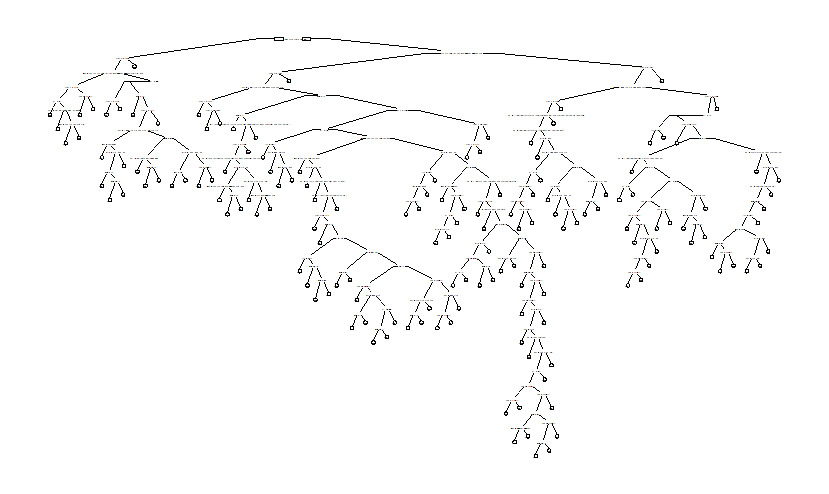
**2) Code and Output - at end of report**

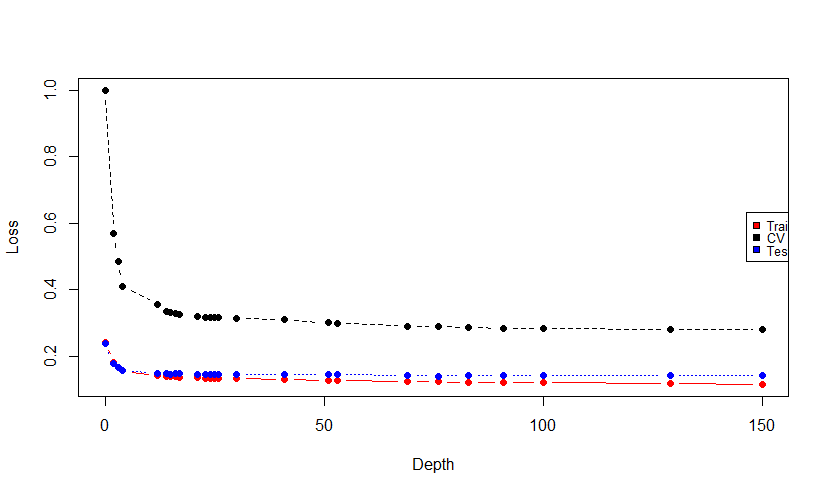
**3) Our Findings: Interpretation and Discussion**

**Single Tree**

When performing the rpart function on our dataset, we found that it was already pruning our tree down to a depth of 4. What we decided to do was make our tree much more complex by setting the cp parameter to 0.0003. From there we then pruned our tree and found that a tree of depth 14 was the best model for our single classification tree. On the next page contains images of how we went about that process:

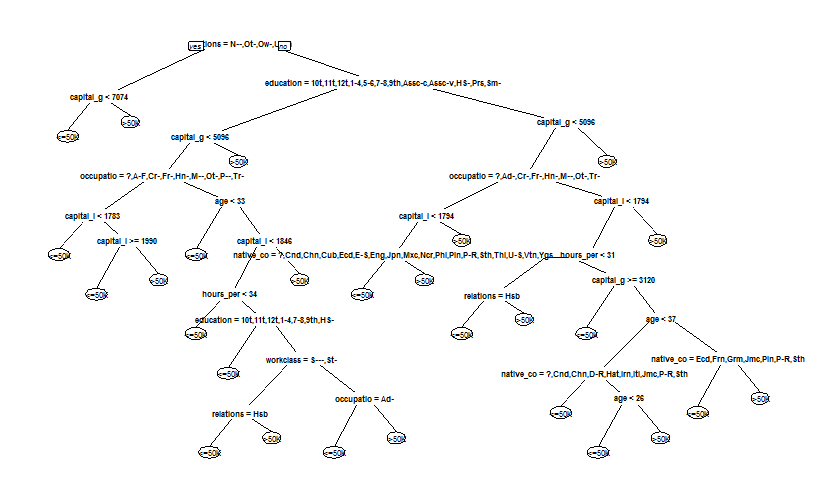
1. Original Tree:





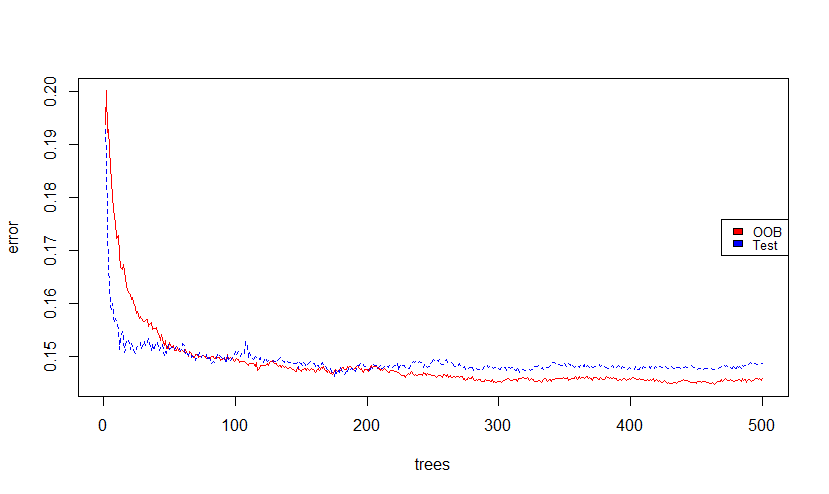
2. Graph of Best Depth (~14):

3. Pruned Tree:



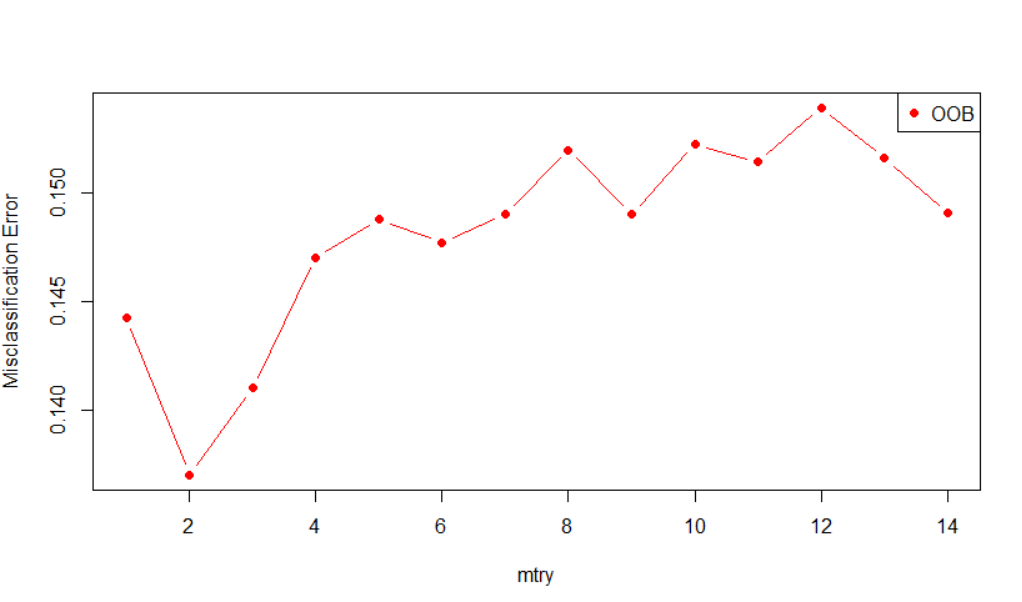
**Bagging**

Using bagging resulted in a model that had a low error rate after about 100 trees are generated:

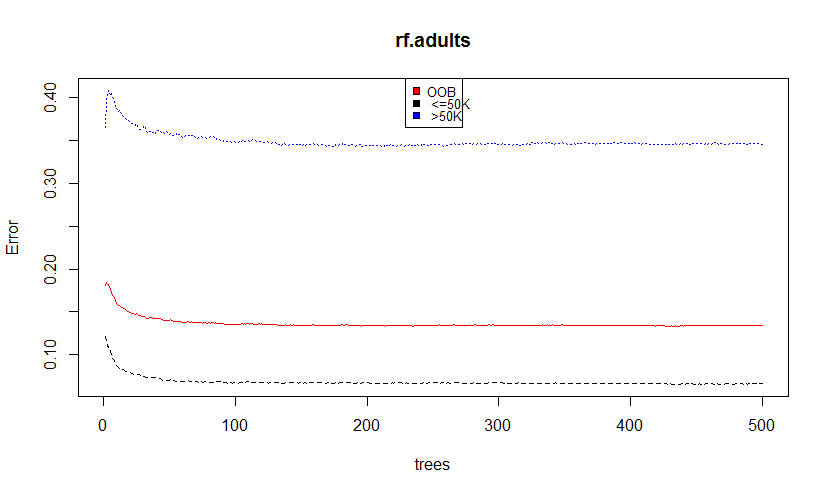
****

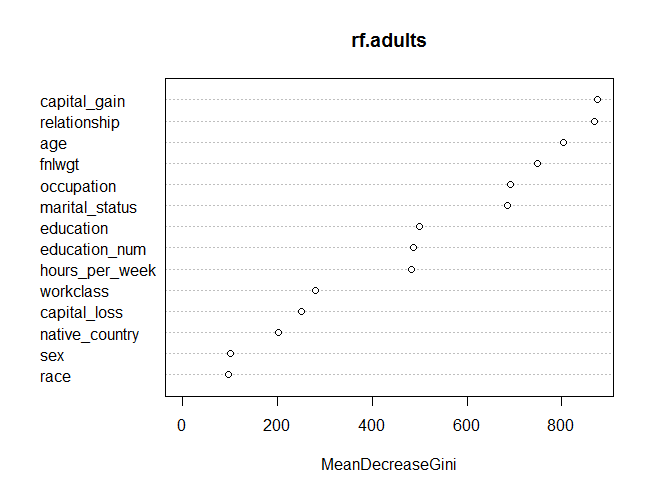
**Random Forests**

After testing 14 different values for mtry (the number of observations per branch) it was determined that the random forest was most accurate at an mtry value of 2:



The random forests model resulted in better accuracy with fewer trees, level out at around 50 trees:

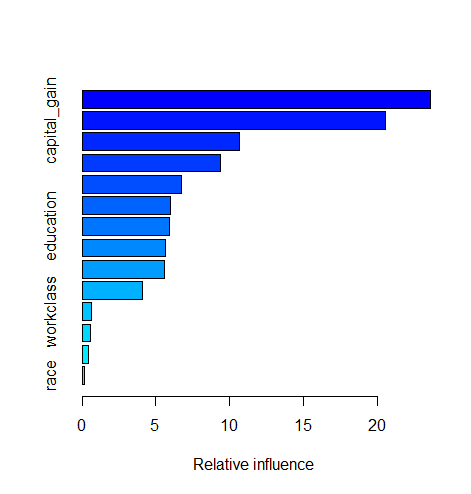
Using gini measurement to find the most effective observations, capital gain had the best measurement, followed by relationship:



**Boosting**

From boosting, it was determined that the attributes with the highest influence was relationship and capital gain.

The following is a graph of variable relevance:

var rel.inf

relationship 23.5437315

capital\_gain 20.6991121

marital\_status 10.7892803

occupation 9.3765051

native\_country 6.6073437

education\_num 6.0787238

education 5.7255636

age 5.7209692

capital\_loss 5.6142352

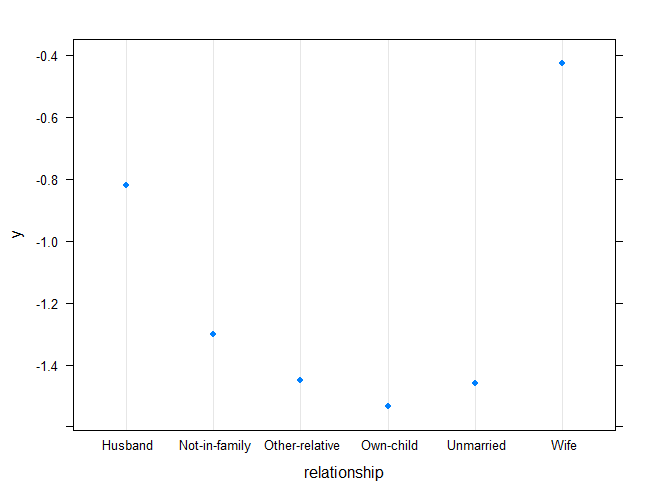
hours\_per\_week 4.1372454

workclass 0.5991952

fnlwgt 0.5251134

sex 0.4353243

race 0.1476572



Relationship Boosting:

**4) Evaluation Methods and Results**

**Single Tree**

- *Confusion Matrix*

We created a confusion matrix to see how our single classification tree was performing:

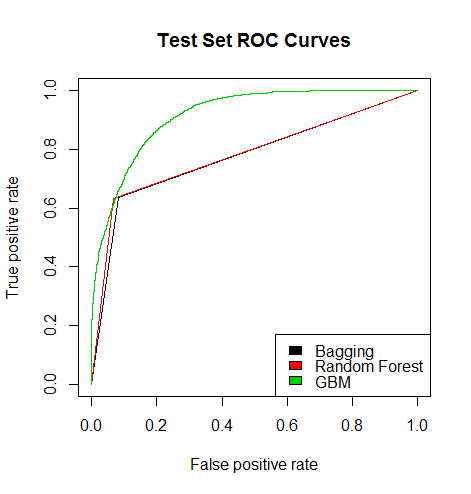
<=50K >50K From our confusion matrix we can see that our tree predicted

<=50K 7095 352 7095 cases of income being <= 50K correctly and 1272 cases of

>50K 1050 1272 income being >50K. It incorrectly predicted a total of 1050 cases of

income being <= 50K and 352 cases of income being >50K. We can also infer our *accuracy* from this, which was 85.65%**.**

**Bagging, Random Forests and Boosting**

In order to evaluate our methods such as bagging, random forests and boosting, we plotted their roc curves on one individual plot. The objective of this was to see which method’s curve was closer to 1. Also, we looked at the area under the curve. This would tell us which method was producing better predictions for our tree.

- *ROC*

From the plot, boosting was the best in predicting for our classification tree.

- *AUC*

Boosting was also determined the best through AUC.

Boosting: 0.9166

Bagging: 0.7749

Random Forest: 0.7833

**5) Random Forests Pseudo-code**

The following is our pseudo-code for our random forests model:

# random forests algorithm

Load random forest library

Set seed for reuse

# create random forest model using training data

# use the square root of the number of variables in the dataset for the number of variables randomly sampled as candidates at each split.

rf<- randomForest(Y~., data = training data, mtry=round(sqrt(ncol(training data) - 1)), importance=TRUE)

Plot mean decrease accuracy and gini of variable to see importance of variables.

Plot model to find optimal mtry (number of variables randomly sampled as candidates at each split)

Plot oob error to find other candidates of mtry

**6) References**

**Works Cited**

Data Science by Arpan Gupta IIT, Roorkee. “Decision Tree in R.” *YouTube*, YouTube, 23 Dec. 2016, [www.youtube.com/watch?v=9ED\_cw1VjrE](http://www.youtube.com/watch?v=9ED_cw1VjrE).

“Gbm.” *Function | R Documentation*,

[www.rdocumentation.org/packages/gbm/versions/2.1.5/topics/gbm](http://www.rdocumentation.org/packages/gbm/versions/2.1.5/topics/gbm).

Kohavi , Ronny, and Barry Becker . “Adult Data Set .” *UCI Machine Learning Repository: Adult Data Set*, <https://archive.ics.uci.edu/ml/datasets/Adult>

Learning, Statistical. “StatsLearning Lect10 R Trees B 111213.” *YouTube*, YouTube, 2 Dec. 2013, [www.youtube.com/watch?v=IY7oWGXb77o&t=272s](http://www.youtube.com/watch?v=IY7oWGXb77o&t=272s).

“Plot & Compare ROC Curves.” *R*, campus.datacamp.com/courses/machine-learning-with-tree-based-models-in-r/boosted-trees?ex=12.

“RandomForest.” *Function | R Documentation*, [www.rdocumentation.org/packages/randomForest/versions/4.6-14/topics/randomForest](http://www.rdocumentation.org/packages/randomForest/versions/4.6-14/topics/randomForest).

**2) Code & Output**

**> # Daniel Gomes, John Gomes, Blake Simmons**

**> # CIS490 Project 2**

**> # Adults dataset**

**>**

**> # libraries**

**> library(rpart)**

**> library(rpart.plot)**

**> library(randomForest)**

**> library(pROC)**

**>**

**> adults <- read.csv('adult.csv', header=FALSE)**

**>**

**> #name columns**

**> names(adults) <- c("age", "workclass", "fnlwgt", "education",**

**+ "education\_num", "marital\_status", "occupation", "relationship",**

**+ "race", "sex", "capital\_gain", "capital\_loss", "hours\_per\_week", "native\_country",**

**+ "class")**

**>**

**> #set seed**

**> set.seed(490)**

**>**

**> #split data**

**> adults.split <- sample(1:nrow(adults), size = nrow(adults) \* 0.7)**

**> adults.train <- adults[adults.split,]**

**> adults.test <- adults[-adults.split,]**

**>**

**>**

**> ###cart model (no pruning) WITH cp = 0.001 in rpart function**

**> ###this will make a tree using the whole dataset - class variable**

**>**

**> adults.cart <- rpart(formula = class ~ ., data= adults.train, method= "class",**

**+ control= rpart.control(cp = 0.0003))**

**> prp(adults.cart, roundint = FALSE, varlen = 3)**

**Warning message:**

**labs do not fit even at cex 0.15, there may be some overplotting**

**> #plotcp(adults.cart)**

**>**

**>**

**> #pruning**

**> cp.adults.param <- adults.cart$cptable**

**>**

**>**

**> #create tables for cross validation**

**>**

**> ###second attempt after adding cp = 0.001 to rpart function**

**> train.acc <- double(25)**

**> cv.acc <- double(25)**

**> test.acc <- double(25)**

**>**

**>**

**> for (i in 1:nrow(cp.adults.param)) {**

**+ alpha <- cp.adults.param[i, 'CP']**

**+ train.cm <- table(adults.train$class, predict(prune(adults.cart, cp=alpha), newdata = adults.train, type='class'))**

**+ train.acc[i] <- 1-sum(diag(train.cm))/sum(train.cm)**

**+ cv.acc[i] <- cp.adults.param[i, 'xerror'] \* cp.adults.param[i, 'rel error']**

**+ test.cm <- table(adults.test$class, predict(prune(adults.cart, cp=alpha), newdata = adults.test, type='class'))**

**+ test.acc[i] <- 1-sum(diag(test.cm))/sum(test.cm)**

**+ }**

**>**

**>**

**> #plot cross validation**

**> matplot(cp.adults.param[,'nsplit'], cbind(train.acc, cv.acc, test.acc), pch=19, col=c("red", "black", "blue"), type="b", ylab="Loss", xlab="Depth")**

**> legend("right", c('Train', 'CV', 'Test'), col=seq\_len(3),cex=0.8,fill=c("red", "black", "blue"))**

**>**

**> #create model and plot new tree**

**> prune.adults.trees <- prune(adults.cart, cp=cp.adults.param[14 ,'CP'])**

**> prp(prune.adults.trees)**

**>**

**> conf.mat.adults <- table(adults.test$class, predict(prune.adults.trees, type='class', newdata = adults.test))**

**> conf.mat.adults**

**<=50K >50K**

**<=50K 7095 352**

**>50K 1050 1272**

**>**

**> acc <- sum(diag(conf.mat.adults))/sum(conf.mat.adults)**

**> acc**

**[1] 0.8564848**

**>**

**> #bagging**

**> set.seed(464)**

**>**

**> bag.adults <- randomForest(class~., data=adults.train, mtry=ncol(adults.train) - 1, importance=TRUE, xtest=adults.test[,-15], ytest=adults.test$class, keep.forest=TRUE)**

**> #bag.adults <- randomForest(class~., data=adults.test, mtry=ncol(adults.test) - 1, importance=TRUE, xtest=adults.test[,-15], ytest=adults.test$class, keep.forest=TRUE)**

**>**

**>**

**> #plot bag.adult**

**> err <- bag.adults$err.rate[,1]**

**> bag.err <- cbind(err, bag.adults$test$err.rate[,1])**

**> colnames(bag.err) <- c("OOB", "Test")**

**> matplot(1:bag.adults$ntree, bag.err, type = "l", xlab = "trees", ylab="error", col= c("red", "blue"))**

**> legend("right", c('OOB', 'Test'), col = seq\_len(2), cex = 0.8, fill=c("red", "blue"))**

**>**

**>**

**> #random forest**

**> set.seed(381)**

**> rf.adults <- randomForest(as.factor(class)~., data=adults.train)**

**> varImpPlot(rf.adults)**

**> rf.adults**

**Call:**

**randomForest(formula = as.factor(class) ~ ., data = adults.train)**

**Type of random forest: classification**

**Number of trees: 500**

**No. of variables tried at each split: 3**

**OOB estimate of error rate: 13.36%**

**Confusion matrix:**

**<=50K >50K class.error**

**<=50K 16130 1143 0.06617264**

**>50K 1903 3616 0.34480884**

**>**

**> plot(rf.adults, col=c("red", "black", "blue"))**

**> legend("top", colnames(rf.adults$err.rate), col=seq\_len(3), cex=0.8, fill=c("red", "black", "blue"))**

**>**

**> p <-ncol(adults.train) - 1**

**> oob.error.adults <- double(p)**

**> set.seed(262)**

**>**

**> for(m in 1:p) {**

**+ fit <- randomForest(class ~ ., data = adults.train, mtry = m, ntree = 400)**

**+ con.mat <- fit$err.rate[400]**

**+ oob.error.adults[m] <- fit$err.rate[50, 'OOB']**

**+ cat(m, " ")**

**+ }**

**1 2 3 4 5 6 7 8 9 10 11 12 13 14 >**

**> matplot(1:p, oob.error.adults, pch = 19, col = "red", type = "b", ylab = "Misclassification Error", xlab="mtry")**

**> legend("topright", legend = c("OOB"), pch = 19, col = c("red"))**

**>**

**> #boosting**

**> library(gbm)**

**> set.seed(158)**

**>**

**> class.0.1 <- ifelse(adults.train$class == " >50K", 1, 0)**

**> adults.boost = gbm(class.0.1 ~ . - class, data = adults.train, n.trees = 5000, distribution = "adaboost", shrinkage = 0.01)**

**> summary(adults.boost)**

**var rel.inf**

**relationship relationship 23.5437315**

**capital\_gain capital\_gain 20.6991121**

**marital\_status marital\_status 10.7892803**

**occupation occupation 9.3765051**

**native\_country native\_country 6.6073437**

**education\_num education\_num 6.0787238**

**education education 5.7255636**

**age age 5.7209692**

**capital\_loss capital\_loss 5.6142352**

**hours\_per\_week hours\_per\_week 4.1372454**

**workclass workclass 0.5991952**

**fnlwgt fnlwgt 0.5251134**

**sex sex 0.4353243**

**race race 0.1476572**

**>**

**> #par(mfrow=c(2, 2))**

**> plot(adults.boost, i="relationship")**

**>**

**>**

**> n.trees <- seq(from = 100, to = 5000, by = 100)**

**>**

**> #ROC plotting**

**> bag.pred <- predict(bag.adults, newdata = adults.test)**

**> #bag.pred.prob <- predict(bag.adults, newdata = adults.test, type="prob")**

**>**

**> rf.pred <- predict(rf.adults, newdata = adults.test, type = "class")**

**>**

**> class.0.1 <- ifelse(adults.test$class == " >50K", 1, 0)**

**> boost.pred <- predict(adults.boost, newdata = adults.test, n.trees = 5000)**

**>**

**> preds\_list <- list(bag.pred, rf.pred, boost.pred)**

**> m <- length(preds\_list)**

**> actuals\_list <- rep(list(adults.test$class), m)**

**>**

**> #str(actuals\_list)**

**> #str(preds\_list)**

**> #length(actuals\_list)**

**> #length(preds\_list)**

**>**

**> pred <- prediction(preds\_list, actuals\_list)**

**> rocs <- performance(pred, "tpr", "fpr")**

**> plot(rocs, col=as.list(1:m), main = "Test Set ROC Curves")**

**> legend(x = "bottomright", legend = c("Bagging", "Random Forest", "GBM"), fill = 1:m)**

**>**

**>**

**> summary(bag.pred)**

**<=50K >50K**

**7698 2071**

**>**

**> #bagging AUC**

**> bag.predictions <- prediction(as.numeric(bag.pred), as.numeric(adults.test$class))**

**> bag.perf <- performance(bag.predictions, measure = "auc")**

**> bag.perf@y.values**

**[[1]]**

**[1] 0.778163**

**>**

**> #random forest AUC**

**> rf.predictions <- prediction(as.numeric(rf.pred), as.numeric(adults.test$class))**

**> rf.perf <- performance(rf.predictions, measure = "auc")**

**> rf.perf@y.values**

**[[1]]**

**[1] 0.7832887**

**>**

**> #boosting AUC**

**> boost.predictions <- prediction(as.numeric(boost.pred), as.numeric(adults.test$class))**

**> boost.perf <- performance(boost.predictions, measure = "auc")**

**> boost.perf@y.values**

**[[1]]**

**[1] 0.9165098**