Project Two

Kelly Goles

May 16, 2018

Executive Summary

The following report outlines five prediction models wich use data colloected from the *Human Activity Recognition Using Smartphones Dataset*, to predict the activity being performed by the participants. The obtained dataset has been randomly partitioned into two sets, where 70% of the volunteers was selected for generating the training data(training.csv) and 30% the test data(testing.csv). Our models were trained based on parsed data from the training data, and will be tested against the training data. The models used include: randomForest, Classification and Regression Trees (CART), and Multinomial Logistic Regression with Cross Validation.

Cleaning the Code

Make sure you are in the working directory which contains both the training.csv and the testing.csv.

Install the following packages: install.packages(“caret”) install.packages(“confusionMatrix”) install.packages(“rpart”) install.packages(“tree”) install.packages(“randomForest”)

Import Data

```
Data<- "training.csv"
Table <- read.csv(Data)
Test<- "testing.csv"
Testt <- read.csv(Test)
```

Now we begin the cleaning process We deceided we are going to clean the Test Data first since it contains only 20 rows instead a few thousand and we can also understand which columns will not be needed since it contains columns that are all NAs.

Delete Columns that contain only NAs

```
Testc <- Testt[, colSums(is.na(Testt)) != nrow(Testt)]
```

Now we will delete the columns we think are not important -We believe the three timestamp columns should be deleted since all they tell us in the time and not activity the person is doing - We believe the new window column is not important since in the test data they all are equal to no - We get rid of the column labeled X since it only tells us what observation it is and this is already shown by the corresponding row number for that observation

Delete unwanted Columns

```
Testc2 <- Testc[, -3:-6]
Testc2 <- Testc2[, -1]
```

Our next intention is to match the columns of the Test Data with the columns of the Training Data. However, first we must save the classe column in the training data since it contains the solution to the Training Data -We renamed the column Answer

Save Classe

```
Answer <- Table[, 160]
```

We find which columnns have the same names in both the new Test Data (cleaned) and the Training Data

Find Column Names' Intersection

```
Colc <- intersect(colnames(Table), colnames(Testc2))
```

We will now use these intersection of the column names to get rid of the columns in both Test Data and Training Data that are not the same -This also helps in regard to when we predict using our models becasue we know we have the same columns in both the Test Data and Training Data

Delete non equivalent column names

```
Tablec <- Table[, Colc]
Testc3 <- Testc2[, Colc]
```

Finally, We need to add the Answer column back into Training Data so that when we run the models, the solutions are within the data frame.

Add Answer Column Back

```
Tablec2 <- cbind(Tablec, Answer)
```

Quick Removal of unneeded data frames

Remove data frames

```
rm(Testt)
rm(Table)
rm(Testc2)
rm(Testc)
```

randomForest Model: Matt Tizik

I decided to do a random forest for my modeling method

First, I am going to partition the data 80% Training and 20% Testing in order to see how well my random forest model will work

Partition

```
Training1 <- sample(1:nrow(Tablec2), size= .8* nrow(Tablec2))

TrainingData <- Tablec2[Training1, ]
TestData <- Tablec2[-Training1, ]
```

Second, I am going to run my random forest model on my training set with the solutions being the column titled Answer -I am using 200 trees instead of the default 500 since it is faster, less stressful on the computer, and more than enough diversifiction for the data -I am using 10 variables at each split to accomidate the reduction in number of trees -Using 10 out of 53 variables at each split should not over teach my model or complicate it too much

Random Forest

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.4.4
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
TrainD1 <- randomForest(Answer ~ . , data= TrainingData, importance= T, ntree=200, mtry = 10)
```

CMatrix and Statistics

```
##
## Call:
##  randomForest(formula = Answer ~ ., data = TrainingData, importance = T,      ntree = 200,
mtry = 10)
##                Type of random forest: classification
##                      Number of trees: 200
## No. of variables tried at each split: 10
##
##         OOB estimate of  error rate: 0.19%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 4470    0    0    0    0 0.0000000000
## B    3 3033    0    0    0 0.0009881423
## C    0   10 2699    0    0 0.0036913990
## D    0    0   10 2569    2 0.0046493607
## E    0    0    0    5 2896 0.0017235436
```

We see that we had an error rate of .19% and we miss classified 30 of the 15697 observations -Solid and as stated in our most recent class we don't want to low of an in sample error becasue it may hurt our out of sample error

Prediction on my Test data

Now we will see how my Random Forest did in regard to the testing data which I had partitioned -20% of original Training Data

```
Predict1 <- predict(TrainD1, TestData)
```

Frequency Table

Here is a quick table showing how well my predictions did in regard to the actual Answer to Test Data (20% Original Training Data)

```
CMatrix<- table(Predict1, TestData$Answer)
```

We missed 6 of the 3925 observations -Nice

Prediction on Real Test Data

Now we will use the random Forest model to predict the Class of the Test Data

```
Solution <- predict(TrainD1, Testc3)
```

Solution

Here is a table of the answers

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  B  A  B  A  A  E  D  B  A  A  B  C  B  A  E  E  A  B  B  B
## Levels: A B C D E
```

Final Table of Test Data

We will combine it with the Test data as the last column so that it is comparable to the Cleaned Training Data

Test_Solution_Table <- cbind(Testc3, Solution) View(Test_Solution_Table)

Classification and Regression Trees Model: Kelly Goles

Set Working Directory and load test&training data

```
setwd("C:/Users/Student/Desktop")
TrainD <-read.csv("training.csv")
TestD <-read.csv("testing.csv")
```

Delete Columns that only contain NA

```
Testc <- TestD[, colSums(is.na(TestD)) != nrow(TestD)]
Testc2 <- Testc[, -2:-7]
```

Value that contains answers to training data set

```
Answer <- TrainD[, 160]
```

Find columns that are in both TrainD and Testc

```
Colc <- intersect(colnames(TrainD), colnames(Testc2))
```

Omit columns that contain:

```
Tablec <- TrainD[, Colc]
Testc3 <- Testc2[, Colc]
```

Add Answers back into data

```
Tablec2 <- cbind(Tablec, Answer)
```

Remove “X” Column

```
Tablec4 <- Tablec2[,-c(1)]
```

Set Seed

```
set.seed(123)
```

Partition training data

```
alpha     <- 0.8
inTrain   <- sample(1:nrow(Tablec2), alpha * nrow(Tablec2))
train.set <- Tablec2[inTrain,]
test.set  <- Tablec2[-inTrain,]
```

Install Tree Packages

```
library(tree)
```

```
## Warning: package 'tree' was built under R version 3.4.4
```

Tree Model to Regress Trainig Data against classification

```
tree.model <-tree(Answer ~ . , data=train.set)
tree.model
```

```
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
##  1) root 15697 49820 A ( 0.2855 0.1948 0.1723 0.1635 0.1840 )
##    2) X < 9377.5 7538 10180 A ( 0.5945 0.4055 0.0000 0.0000 0.0000 )
##      4) X < 5581.5 4481     0 A ( 1.0000 0.0000 0.0000 0.0000 0.0000 ) *
##      5) X > 5581.5 3057     0 B ( 0.0000 1.0000 0.0000 0.0000 0.0000 ) *
##    3) X > 9377.5 8159 17910 E ( 0.0000 0.0000 0.3314 0.3146 0.3540 )
##      6) X < 16015.5 5271  7304 C ( 0.0000 0.0000 0.5130 0.4870 0.0000 )
##       12) X < 12799.5 2704     0 C ( 0.0000 0.0000 1.0000 0.0000 0.0000 ) *
##       13) X > 12799.5 2567     0 D ( 0.0000 0.0000 0.0000 1.0000 0.0000 ) *
##      7) X > 16015.5 2888     0 E ( 0.0000 0.0000 0.0000 0.0000 1.0000 ) *
```

```
summary(tree.model)
```

```
##
## Classification tree:
## tree(formula = Answer ~ ., data = train.set)
## Variables actually used in tree construction:
## [1] "X"
## Number of terminal nodes:  5
## Residual mean deviance:  0 = 0 / 15690
## Misclassification error rate: 0 = 0 / 15697
```

Distributional Prediction

```
my.prediction <- predict(tree.model, test.set)
```

Point Prediction- the probability output to categorical output

```
maxidx <- function(arr) {
  return(which(arr == max(arr)))
}
idx <- apply(my.prediction, c(1), maxidx)
prediction <- c('A', 'B', 'C', 'D', 'E')[idx]
table(prediction, test.set$Answer)
```

```
##
## prediction    A    B    C    D    E
##          A 1099    1    0    0    0
##          B    0  739    0    0    0
##          C    0    0  718    0    0
##          D    0    0    0  649    0
##          E    0    0    0    0  719
```

Prune the tree to prevent overfitting/ Fits model with optimal number of leaves

```
pruned.tree <- prune.tree(tree.model, best=5)
```

Give the predicted class

```
pruned.prediction <- predict(pruned.tree, test.set, type="class")
table(pruned.prediction, test.set$Answer)
```

```
##
## pruned.prediction    A    B    C    D    E
##                 A 1099    1    0    0    0
##                 B    0  739    0    0    0
##                 C    0    0  718    0    0
##                 D    0    0    0  649    0
##                 E    0    0    0    0  719
```

Confusion Matrix

```
getAnywhere(confusionMatrix(data = pruned.prediction, test.set$Answer))
```

```
## Warning in find(x, numeric = TRUE): elements of 'what' after the first will
## be ignored
```

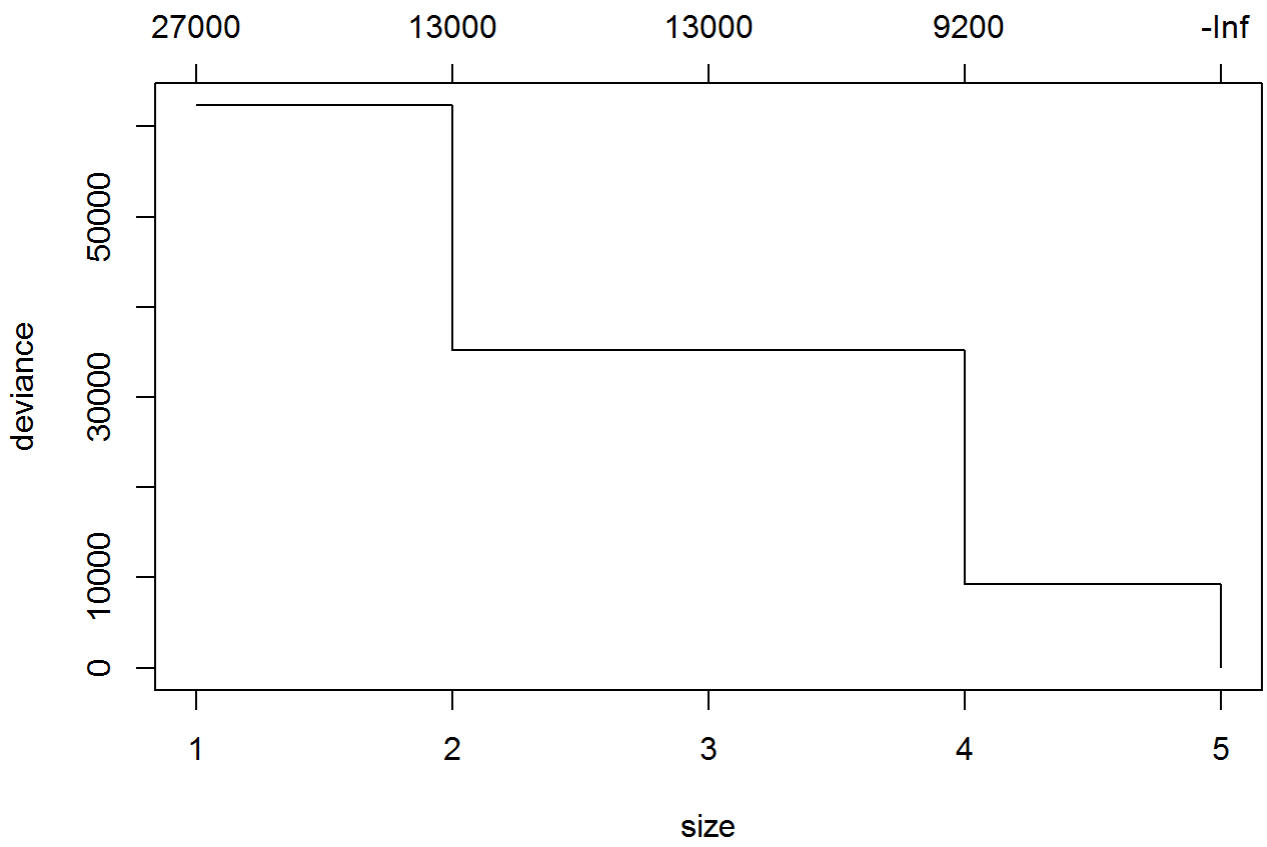
```
## no object named 'confusionMatrix' 'pruned.prediction' 'test.set$Answer' was found
```

K-fold cross-validation to find the best tree

```
tree.model <- tree(Answer ~ ., data=Tablec2)
summary(tree.model)
```

```
##
## Classification tree:
## tree(formula = Answer ~ ., data = Tablec2)
## Variables actually used in tree construction:
## [1] "X"
## Number of terminal nodes:  5
## Residual mean deviance:  0 = 0 / 19620
## Misclassification error rate: 0 = 0 / 19622
```

```
cv.model <- cv.tree(tree.model)
plot(cv.model)
```



Gives the deviance for each K

```
cv.model$dev
```

```
## [1]    55.26204  9238.89298 35166.66929 35166.66929 62322.98840
```

Gives which size is better

```
best.size <- cv.model$size[which(cv.model$dev==min(cv.model$dev))]
best.size
```

```
## [1] 5
```

Refit the tree model (the number of leafs will be no more than best.size)

```
cv.model.pruned <- prune.misclass(tree.model, best=best.size)
summary(cv.model.pruned)
```

```
##
## Classification tree:
## tree(formula = Answer ~ ., data = Tablec2)
## Variables actually used in tree construction:
## [1] "X"
## Number of terminal nodes:  5
## Residual mean deviance:  0 = 0 / 19620
## Misclassification error rate: 0 = 0 / 19622
```

Classification and Regression Trees: Trinity Manning-Pickett

Partition

```
Training1 <- sample(1:nrow(Tablec2), size= .8* nrow(Tablec2))
TrainingData <- Tablec2[Training1, ]
TestData <- Tablec2[-Training1, ]
```

Create CART Model

```
set.seed(2224)
library(rpart)
```

```
## Warning: package 'rpart' was built under R version 3.4.4
```

```
rpart <- rpart(TrainingData$Answer~ . , data = TrainingData, method  = "class")
```

Create and View Estimated Values of Model

```
getAnywhere(predict)
```

```
## A single object matching 'predict' was found
## It was found in the following places
##   package:stats
##   namespace:stats
## with value
##
## function (object, ...)
## UseMethod("predict")
## <bytecode: 0x00000000066f47c0>
## <environment: namespace:stats>
```

```
Prediction <- predict(rpart, TestData, type = "vector")
View(Prediction)
```

Substitute Numerical Values with Classifications

```
Prediction1 <- gsub(1, "A", Prediction)
Prediction2 <- gsub(2, "B", Prediction1)
Prediction3 <- gsub(3, "C", Prediction2)
Prediction4 <- gsub(4, "D", Prediction3)
Prediction5 <- gsub(5, "E", Prediction4)
```

Check Classes of Test Data and Prediction Outputs

```
class(TestData$Answer)
```

```
## [1] "factor"
```

```
class(Prediction5)
```

```
## [1] "character"
```

Make Both Outputs Factors

Test Data class is “factor”, while Prediction5 is “character”. Reformate Prediction from “character” to “factor”.

```
Prediction6 <- as.factor(Prediction5)
```

Test CART Model

Using a Confusion Matrix

```
CM <- getAnywhere(confusionMatrix(Prediction6, TestData$Answer))
```

```
## Warning in find(x, numeric = TRUE): elements of 'what' after the first will
## be ignored
```

```
print(CM)
```

```
## no object named 'confusionMatrix' 'Prediction6' 'TestData$Answer' was found
```

randomForest Model: Ursula Eisinger

Set working directory.

```
getwd()
```

```
## [1] "C:/Users/Student/Desktop"
```

```
setwd("C:/Users/Student/Desktop")
```

Import training data and remove unnecessary columns.

```
original_train <- read.csv("training.csv")
new_train <- original_train[,c(2, 7:11, 37:49, 60:68, 84, 85, 86, 113:124, 140, 151:160)]
```

Create answer column.

```
answer <- new_train[, 54]
```

Import testing data and remove unnecessary columns.

```
original_test <- read.csv("testing.csv")
new_test <- original_test[,c(2, 7:11, 37:49, 60:68, 84, 85, 86, 113:124, 140, 151:159)]
```

Ensures that the training and testing column names are the same.

```
column_c <- intersect(colnames(new_train), colnames(new_test))
Train0 <- new_train[, column_c]
Test1 <- new_test[, column_c]
Train1 <- cbind(Train0, answer)
```

Partition the training data.

```
set.seed(123)
samplea <- floor(0.7 *nrow(Train1))
ind <- sample(seq_len(nrow(Train1)), size = samplea)
train_1 <- Train1[ind, ]
test_1 <- Train1[-ind, ]
```

Build the Classification Model using the Random Forest package.

```
library(randomForest)
set.seed(222)
train_1$answer = as.factor(train_1$answer)
rf <-randomForest(answer~., data=train_1)
print(rf)
```

```
##
## Call:
##  randomForest(formula = answer ~ ., data = train_1)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.28%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3922    0    0    0    1 0.000254907
## B    3 2680    3    0    0 0.002233805
## C    0   12 2354    0    0 0.005071851
```

```
## D    0    0   15 2237    2 0.007542147
## E    0    0    0    3 2503 0.001197127
```

```
attributes(rf)
```

```
## $names
##  [1] "call"            "type"            "predicted"
##  [4] "err.rate"        "confusion"       "votes"
##  [7] "oob.times"       "classes"         "importance"
## [10] "importanceSD"    "localImportance" "proximity"
## [13] "ntree"           "mtry"            "forest"
## [16] "y"               "test"            "inbag"
## [19] "terms"
##
## $class
## [1] "randomForest.formula" "randomForest"
```

Prediction and Confusion Matrix with the training data.

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.4.4
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
##
## Attaching package: 'ggplot2'
```

```
## The following object is masked from 'package:randomForest':
##
##     margin
```

```
p1 <- predict(rf, train_1)
head(train_1$answer) ##check predictions on actual data
```

```
## [1] B D B E E A
## Levels: A B C D E
```

```
confusionMatrix(p1, train_1$answer)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
```

```
##          A 3923    0    0    0    0
##          B    0 2686    0    0    0
##          C    0    0 2366    0    0
##          D    0    0    0 2254    0
##          E    0    0    0    0 2506
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9997, 1)
##     No Information Rate : 0.2856
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2856   0.1956   0.1723   0.1641   0.1825
## Detection Rate         0.2856   0.1956   0.1723   0.1641   0.1825
## Detection Prevalence   0.2856   0.1956   0.1723   0.1641   0.1825
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

Prediction and Confusion Matrix with the testing data (partitioned from training).

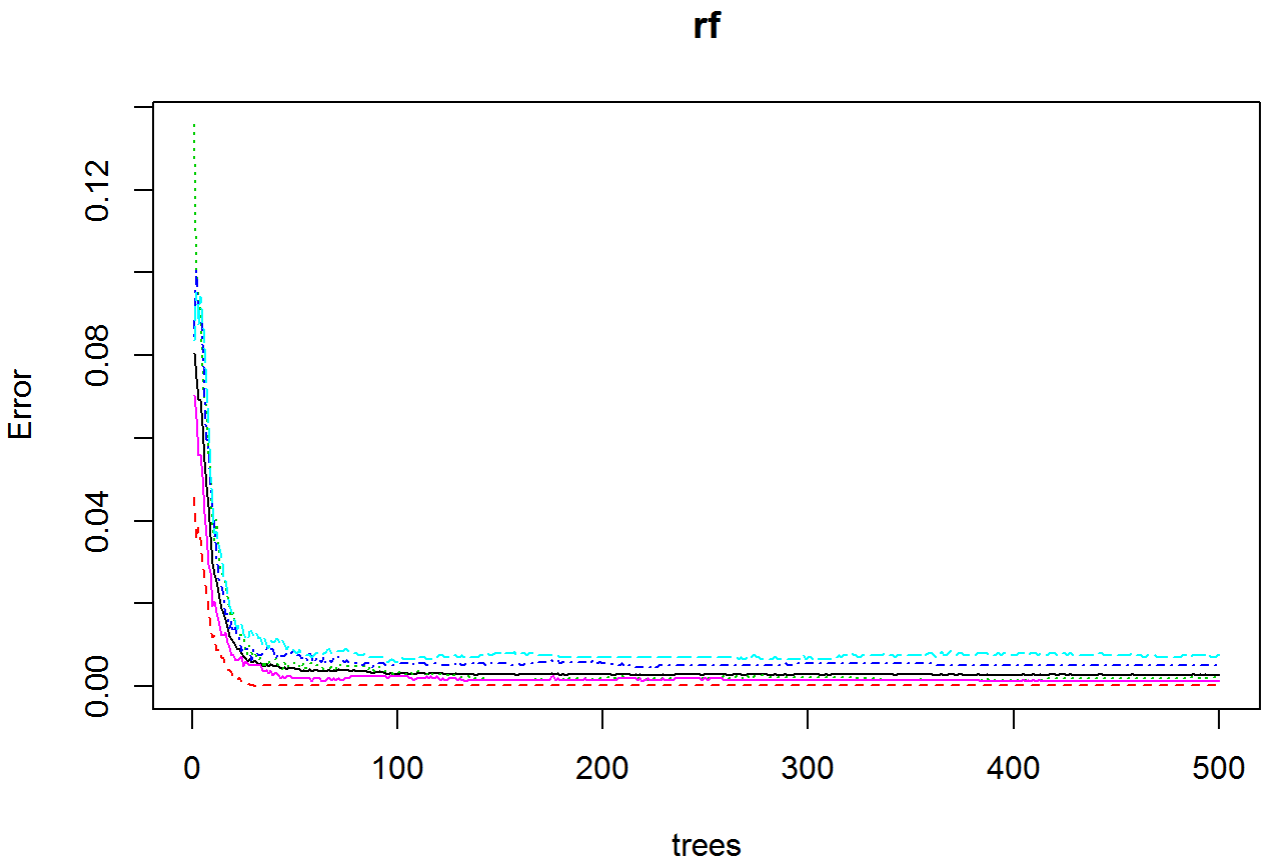
```
library(caret)
test_1 <- rbind(train_1[1, ] , test_1)  ##equalize train and test to remove non-matching predi
ctor error
test_1 <- test_1[-1,]
p2 <- predict(rf, test_1)
confusionMatrix(p2, test_1$answer)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1657    1    0    0    0
##          B    0 1110    9    0    0
##          C    0    0 1047    6    0
##          D    0    0    0  956    2
##          E    0    0    0    0 1099
##
## Overall Statistics
##
##                Accuracy : 0.9969
##                  95% CI : (0.9952, 0.9982)
##     No Information Rate : 0.2815
##     P-Value [Acc > NIR] : < 2.2e-16
```

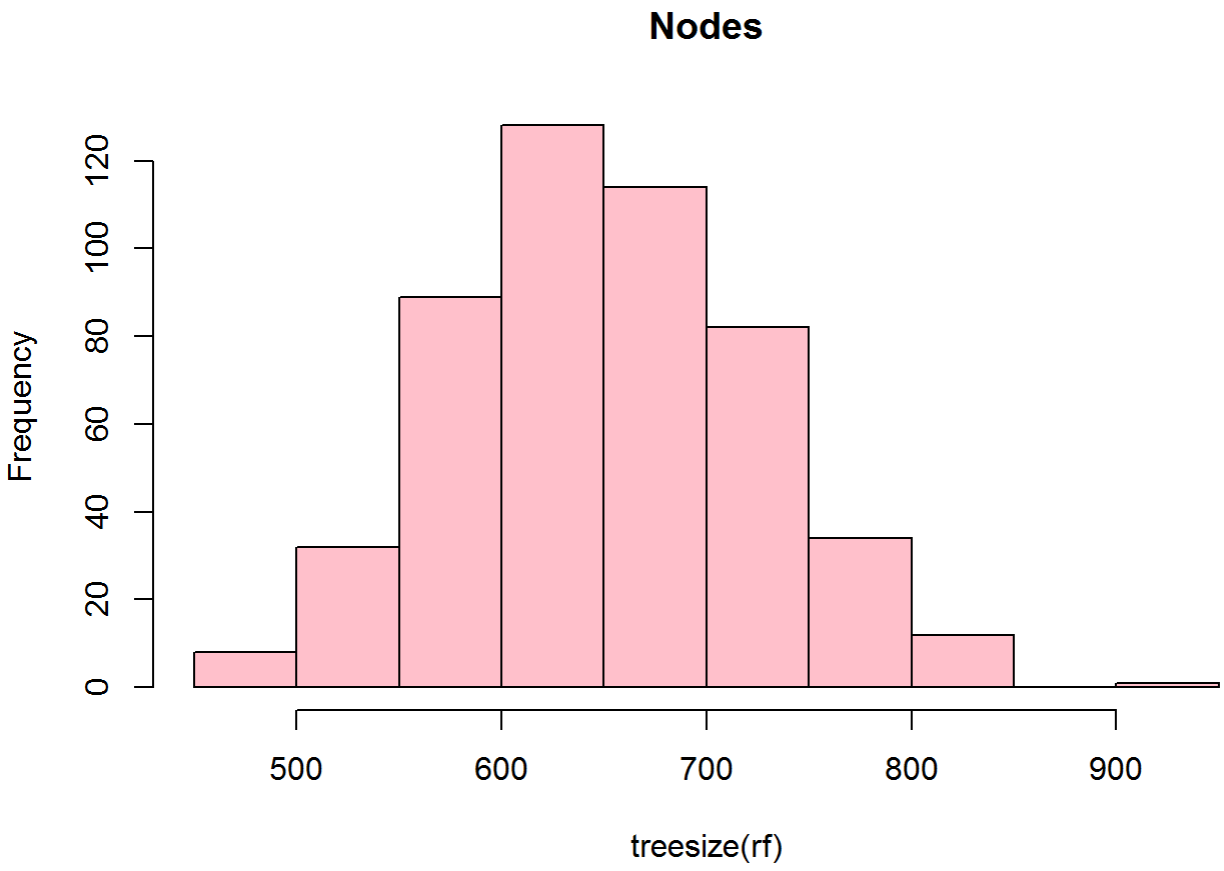
```
##
##                  Kappa : 0.9961
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9991   0.9915   0.9938   0.9982
## Specificity            0.9998   0.9981   0.9988   0.9996   1.0000
## Pos Pred Value         0.9994   0.9920   0.9943   0.9979   1.0000
## Neg Pred Value         1.0000   0.9998   0.9981   0.9988   0.9996
## Prevalence             0.2815   0.1887   0.1794   0.1634   0.1870
## Detection Rate         0.2815   0.1886   0.1778   0.1624   0.1867
## Detection Prevalence   0.2816   0.1901   0.1789   0.1627   0.1867
## Balanced Accuracy      0.9999   0.9986   0.9951   0.9967   0.9991
```

Create graphs to show optimal tree number, tree nodes, and variable importance.

```
plot(rf)
```




```
hist(treesize(rf), main = 'Nodes', col = "pink")
```




```
importance(rf)
```

```
##                       MeanDecreaseGini
## user_name                    124.30014
## num_window                   961.02240
## roll_belt                    818.60917
## pitch_belt                   463.42136
## yaw_belt                     569.81302
## total_accel_belt             157.29264
## gyros_belt_x                  60.37653
## gyros_belt_y                  74.02910
## gyros_belt_z                 185.82020
## accel_belt_x                  85.94703
## accel_belt_y                  87.20938
## accel_belt_z                 247.00768
## magnet_belt_x                169.99458
## magnet_belt_y                247.42886
## magnet_belt_z                252.35494
## roll_arm                     204.78923
## pitch_arm                    109.83857
## yaw_arm                      138.33722
## total_accel_arm               61.41624
## gyros_arm_x                   73.25021
## gyros_arm_y                   78.65563
## gyros_arm_z                   34.98541
```

```
## accel_arm_x               143.67865
## accel_arm_y                88.97521
## accel_arm_z                85.49726
## magnet_arm_x              159.98513
## magnet_arm_y              146.48149
## magnet_arm_z              107.25313
## roll_dumbbell             271.82242
## pitch_dumbbell            123.86529
## yaw_dumbbell              174.28306
## gyros_dumbbell_x           78.89039
## gyros_dumbbell_y          156.05368
## gyros_dumbbell_z           48.76746
## accel_dumbbell_x          186.98782
## accel_dumbbell_y          286.74222
## accel_dumbbell_z          236.39551
## magnet_dumbbell_x         322.33963
## magnet_dumbbell_y         461.10152
## magnet_dumbbell_z         480.90724
## roll_forearm              365.12656
## pitch_forearm             525.51396
## yaw_forearm               100.51724
## total_accel_forearm        64.09114
## gyros_forearm_x            45.18980
## gyros_forearm_y            74.73625
## gyros_forearm_z            51.06099
## accel_forearm_x           211.33546
## accel_forearm_y            78.64365
## accel_forearm_z           147.56917
## magnet_forearm_x          133.68042
## magnet_forearm_y          125.92566
## magnet_forearm_z          164.33449
```

Prediction for the real test data.

```
library(caret)
p3 <- predict(rf, Test1)
print(p3)
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  B  A  B  A  A  E  D  B  A  A  B  C  B  A  E  E  A  B  B  B
## Levels: A B C D E
```

Multinomial Logistic Regression with Cross Validation: Katherine Stolin

Partition

Split training data into subset training and testing group for building model

```
set.seed(1234)
Training1 <- sample(1:nrow(Tablec2), size= .7* nrow(Tablec2))
```

```
TrainingData <- Tablec2[Training1, ]
TestData <- Tablec2[-Training1, ]
```

Build Multi Model

```
library(nnet)
```

```
## Warning: package 'nnet' was built under R version 3.4.4
```

```
multimodel <- multinom(Answer~ . , data = TrainingData, maxit = 500, trace = T)
```

```
## # weights:  275 (216 variable)
## initial  value 22105.629727
## iter  10 value 15459.265706
## iter  20 value 12459.735536
## iter  30 value 11832.138846
## iter  40 value 11274.380928
## iter  50 value 10945.747594
## iter  60 value 10600.678340
## iter  70 value 10313.618698
## iter  80 value 10166.870861
## iter  90 value 10116.300689
## iter 100 value 10081.909805
## iter 110 value 10055.215727
## iter 120 value 10019.057894
## iter 130 value 9982.696345
## iter 140 value 9901.426872
## iter 150 value 9858.052360
## iter 160 value 9754.407571
## iter 170 value 9415.139553
## iter 180 value 7560.470816
## iter 190 value 6166.828265
## iter 200 value 4650.103920
## iter 210 value 3290.872462
## iter 220 value 751.423424
## iter 230 value 345.278332
## iter 240 value 218.216369
## iter 250 value 149.858842
## iter 260 value 101.303786
## iter 270 value 75.754803
## iter 280 value 54.695935
## iter 290 value 39.957296
## iter 300 value 29.939951
## iter 310 value 22.872508
## iter 320 value 15.353986
## iter 330 value 9.531416
## iter 340 value 4.018790
## iter 350 value 0.957229
## iter 360 value 0.203735
## iter 370 value 0.054164
## iter 380 value 0.024292
```

```
## iter 390 value 0.005158
## iter 400 value 0.001477
## final  value 0.000005
## converged
```

Sort

Sort by most influencial variables

Predict

With testing data See what it looks like using the head function

```
multipred1 <- predict(multimodel, type = "probs", newdata = TestData)
multipred2 <- predict(multimodel, type = "class", newdata = TestData)

head(multipred2, 5)
```

```
## [1] A A A A A
## Levels: A B C D E
```

```
head(multipred1, 5)
```

```
##   A B C D E
## 1 1 0 0 0 0
## 2 1 0 0 0 0
## 3 1 0 0 0 0
## 4 1 0 0 0 0
## 5 1 0 0 0 0
```

Model Selction

Of the five models that we ran, we decided to select Ursula's Random Forest model. Trinity used a recursive partitioning model and had approximately 75% accuracy, Katherine used a Multinomial Logistic Regression with Cross Validation model had 78% accuracy, Kelly used a CART model and was 99.7% accurate, Ursula used a randomForest model and was 99.7% accurate, Matt used a randomForest model and was 99.8% accurate. Although Matt's model had slightly higher accuracy, they yeilded the same results when used on the test data. Ultimately we chose Ursula's model because it was less complex and we didn't want to overfit the results.

Prediction for the real test data.

```
library(caret)
p3 <- predict(rf, Test1)
print(p3)
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  B  A  B  A  A  E  D  B  A  A  B  C  B  A  E  E  A  B  B  B
## Levels: A B C D E
```