ML Assignment

Omid Sedighi

6 mars 2017

#LOAD NEEDED PACKAGES  
library(caret)

## Warning: package 'caret' was built under R version 3.3.2

## Loading required package: lattice

## Loading required package: ggplot2

library(randomForest)

## randomForest 4.6-12

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

##   
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':  
##   
## margin

library(rpart)  
library(RColorBrewer)  
library(gbm)

## Loading required package: survival

##   
## Attaching package: 'survival'

## The following object is masked from 'package:caret':  
##   
## cluster

## Loading required package: splines

## Loading required package: parallel

## Loaded gbm 2.1.1

###LOAD DATA   
#Remove errors and blanks with NA  
trainData <- read.csv("pml-training.csv", head=TRUE, sep=",", na.strings=c("NA","#DIV/0!",""))   
testData <- read.csv("pml-testing.csv", head=TRUE, sep=",", na.strings=c("NA","#DIV/0!",""))   
  
  
#Take a look at the data before pre-processing  
dim(trainData)

## [1] 19622 160

dim(testData)

## [1] 20 160

levels(trainData$classe)

## [1] "A" "B" "C" "D" "E"

summary(trainData$classe)

## A B C D E   
## 5580 3797 3422 3216 3607

###Data pre-processing  
  
#Dectecting missing vlaues  
NApercentTrain <- sapply(trainData, function(df) {sum(is.na(df)==TRUE)/length(df)})  
NApercentTest <- sapply(testData, function(df) {sum(is.na(df)==TRUE)/length(df)})  
  
#The results from above show that a lot of variables have more than 97% missing values  
#The below table shows this  
table(NApercentTrain > .97)

##   
## FALSE TRUE   
## 60 100

table(NApercentTest > .97)

##   
## FALSE TRUE   
## 60 100

#Remove exceeding variables  
removetrain <- names(which(NApercentTrain < 0.97))  
trainData <- trainData[, removetrain]  
removetest <- names(which(NApercentTest < 0.97))  
testData <- testData[, removetest]  
  
sum(is.na(trainData) == TRUE)

## [1] 0

sum(is.na(testData) == TRUE)

## [1] 0

#And if we look at the dimensions now  
dim(trainData)

## [1] 19622 60

dim(testData)

## [1] 20 60

#Now we can see that the first 7 variables are metadata and we which we can exklude  
  
trainData <- trainData[,-c(1:7)]  
testData <- testData[,-c(1:7)]  
dim(trainData)

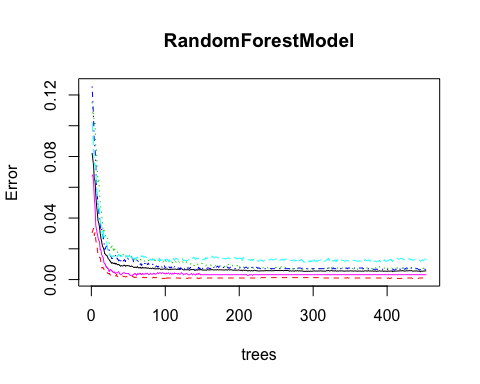
## [1] 19622 53

dim(testData)

## [1] 20 53

###Data splitting for resampling  
  
set.seed(123)  
inTrain <- createDataPartition (trainData$classe, p=0.7, list=FALSE)  
training <- trainData [inTrain ,]  
testing <- trainData [- inTrain,]  
  
  
###Model fitting  
  
####Random Forest####  
RandomForestModel <- randomForest(classe ~ ., data=training, ntree=453, mtry = 15)

#Check error ratez§  
plot(RandomForestModel)



#Find optimal number of trees and variables with smallest error rate  
#which.min(RandomForestModel$err.rate[,1])  
#tuneRF(training[,-53], training[,53], stepFactor=1.5)

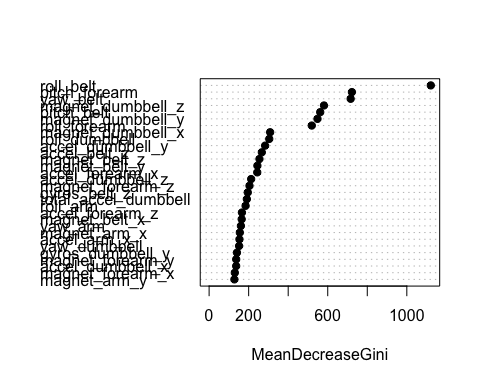
#Print and plot importance plot  
print(RandomForestModel)

##   
## Call:  
## randomForest(formula = classe ~ ., data = training, ntree = 453, mtry = 15)   
## Type of random forest: classification  
## Number of trees: 453  
## No. of variables tried at each split: 15  
##   
## OOB estimate of error rate: 0.56%  
## Confusion matrix:  
## A B C D E class.error  
## A 3901 3 0 1 1 0.001280082  
## B 14 2639 5 0 0 0.007148232  
## C 0 14 2380 2 0 0.006677796  
## D 0 0 27 2223 2 0.012877442  
## E 0 0 4 4 2517 0.003168317

importance(RandomForestModel)

## MeanDecreaseGini  
## roll\_belt 1121.36952  
## pitch\_belt 561.93574  
## yaw\_belt 715.93928  
## total\_accel\_belt 111.72935  
## gyros\_belt\_x 49.59962  
## gyros\_belt\_y 64.37679  
## gyros\_belt\_z 195.58081  
## accel\_belt\_x 52.12572  
## accel\_belt\_y 61.01920  
## accel\_belt\_z 266.62794  
## magnet\_belt\_x 164.81673  
## magnet\_belt\_y 244.22280  
## magnet\_belt\_z 254.65628  
## roll\_arm 184.44907  
## pitch\_arm 108.83106  
## yaw\_arm 160.82118  
## total\_accel\_arm 53.23966  
## gyros\_arm\_x 68.76570  
## gyros\_arm\_y 79.35140  
## gyros\_arm\_z 30.64861  
## accel\_arm\_x 153.38277  
## accel\_arm\_y 88.55875  
## accel\_arm\_z 73.21579  
## magnet\_arm\_x 155.37967  
## magnet\_arm\_y 127.88010  
## magnet\_arm\_z 111.80035  
## roll\_dumbbell 304.17431  
## pitch\_dumbbell 95.76448  
## yaw\_dumbbell 151.62074  
## total\_accel\_dumbbell 190.86393  
## gyros\_dumbbell\_x 77.17390  
## gyros\_dumbbell\_y 140.82492  
## gyros\_dumbbell\_z 46.57544  
## accel\_dumbbell\_x 136.86265  
## accel\_dumbbell\_y 283.32916  
## accel\_dumbbell\_z 212.65910  
## magnet\_dumbbell\_x 308.66984  
## magnet\_dumbbell\_y 548.13644  
## magnet\_dumbbell\_z 581.07284  
## roll\_forearm 519.34934  
## pitch\_forearm 722.26631  
## yaw\_forearm 107.80918  
## total\_accel\_forearm 66.38686  
## gyros\_forearm\_x 36.30188  
## gyros\_forearm\_y 64.61596  
## gyros\_forearm\_z 44.95215  
## accel\_forearm\_x 243.43882  
## accel\_forearm\_y 78.99094  
## accel\_forearm\_z 166.08706  
## magnet\_forearm\_x 130.01177  
## magnet\_forearm\_y 137.00833  
## magnet\_forearm\_z 204.30156

plot.new()  
varImpPlot(RandomForestModel, pch=19, col=1, cex=1.0, main="")  
abline(v=10000, col="blue")



#Predictions  
RFpred <- predict(RandomForestModel, testing, type="response", decision.values = TRUE, probability = TRUE)  
  
#Look at results from RF-model  
confusionMatrix(data = RFpred, testing$classe)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction A B C D E  
## A 1673 4 0 0 0  
## B 1 1133 7 0 0  
## C 0 2 1018 12 0  
## D 0 0 1 951 0  
## E 0 0 0 1 1082  
##   
## Overall Statistics  
##   
## Accuracy : 0.9952   
## 95% CI : (0.9931, 0.9968)  
## No Information Rate : 0.2845   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.994   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: A Class: B Class: C Class: D Class: E  
## Sensitivity 0.9994 0.9947 0.9922 0.9865 1.0000  
## Specificity 0.9991 0.9983 0.9971 0.9998 0.9998  
## Pos Pred Value 0.9976 0.9930 0.9864 0.9989 0.9991  
## Neg Pred Value 0.9998 0.9987 0.9984 0.9974 1.0000  
## Prevalence 0.2845 0.1935 0.1743 0.1638 0.1839  
## Detection Rate 0.2843 0.1925 0.1730 0.1616 0.1839  
## Detection Prevalence 0.2850 0.1939 0.1754 0.1618 0.1840  
## Balanced Accuracy 0.9992 0.9965 0.9947 0.9932 0.9999

#####Gradient boosting classification#####  
  
control = trainControl(method = "CV", number = 10)  
  
#Only do ones to determine parameters  
#gbmtrain <- train(classe ~ ., data = training, method = "gbm", trControl=control)  
  
#Set and train model  
gbm <- gbm(classe ~ ., data = training, n.trees = 150, interaction.depth = 3, shrinkage = 0.1)

## Distribution not specified, assuming multinomial ...

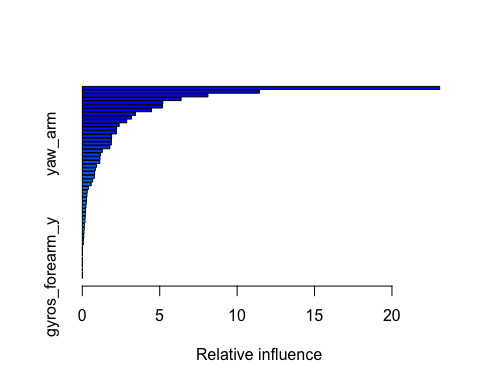
gbm

## gbm(formula = classe ~ ., data = training, n.trees = 150, interaction.depth = 3,   
## shrinkage = 0.1)  
## A gradient boosted model with multinomial loss function.  
## 150 iterations were performed.  
## There were 52 predictors of which 43 had non-zero influence.

gbmtest = predict(gbm, newdata = testing, type = "response", n.trees = 150)  
  
#Classify the final results from given probabilities of every outcome  
gbmres <- apply(gbmtest, 1, which.max)  
  
#Classify above to letters according to classe variable  
gbmres2 <- ifelse(gbmres ==1, "A", ifelse(gbmres ==2, "B", ifelse(gbmres ==3, "C", ifelse(gbmres ==4, "D", "E"))))  
  
  
#Print results  
confusionMatrix(data = gbmres2, testing$classe)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction A B C D E  
## A 1645 52 0 0 6  
## B 17 1066 37 1 9  
## C 7 21 977 43 7  
## D 4 0 12 914 16  
## E 1 0 0 6 1044  
##   
## Overall Statistics  
##   
## Accuracy : 0.9594   
## 95% CI : (0.954, 0.9643)  
## No Information Rate : 0.2845   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.9486   
## Mcnemar's Test P-Value : 3.388e-12   
##   
## Statistics by Class:  
##   
## Class: A Class: B Class: C Class: D Class: E  
## Sensitivity 0.9827 0.9359 0.9522 0.9481 0.9649  
## Specificity 0.9862 0.9865 0.9839 0.9935 0.9985  
## Pos Pred Value 0.9659 0.9434 0.9261 0.9662 0.9933  
## Neg Pred Value 0.9931 0.9846 0.9899 0.9899 0.9921  
## Prevalence 0.2845 0.1935 0.1743 0.1638 0.1839  
## Detection Rate 0.2795 0.1811 0.1660 0.1553 0.1774  
## Detection Prevalence 0.2894 0.1920 0.1793 0.1607 0.1786  
## Balanced Accuracy 0.9845 0.9612 0.9681 0.9708 0.9817

summary(gbm)



## var rel.inf  
## roll\_belt roll\_belt 23.08286924  
## pitch\_forearm pitch\_forearm 11.43906647  
## yaw\_belt yaw\_belt 8.11284334  
## magnet\_dumbbell\_z magnet\_dumbbell\_z 6.38299712  
## magnet\_dumbbell\_y magnet\_dumbbell\_y 5.20088722  
## roll\_forearm roll\_forearm 5.18358792  
## magnet\_belt\_z magnet\_belt\_z 4.46917324  
## pitch\_belt pitch\_belt 3.42903584  
## roll\_dumbbell roll\_dumbbell 3.16973862  
## gyros\_belt\_z gyros\_belt\_z 2.86796683  
## accel\_forearm\_x accel\_forearm\_x 2.38200491  
## accel\_dumbbell\_y accel\_dumbbell\_y 2.21468517  
## gyros\_dumbbell\_y gyros\_dumbbell\_y 2.19917122  
## magnet\_dumbbell\_x magnet\_dumbbell\_x 1.89192077  
## magnet\_forearm\_z magnet\_forearm\_z 1.88529885  
## yaw\_arm yaw\_arm 1.87360319  
## accel\_dumbbell\_x accel\_dumbbell\_x 1.77444253  
## accel\_forearm\_z accel\_forearm\_z 1.29018503  
## magnet\_belt\_y magnet\_belt\_y 1.16386298  
## magnet\_arm\_z magnet\_arm\_z 1.14394037  
## magnet\_forearm\_x magnet\_forearm\_x 1.13116620  
## magnet\_belt\_x magnet\_belt\_x 0.91961880  
## accel\_dumbbell\_z accel\_dumbbell\_z 0.84242950  
## magnet\_arm\_x magnet\_arm\_x 0.79329414  
## accel\_belt\_z accel\_belt\_z 0.77494236  
## roll\_arm roll\_arm 0.66648670  
## magnet\_arm\_y magnet\_arm\_y 0.57163788  
## gyros\_arm\_y gyros\_arm\_y 0.38910481  
## total\_accel\_dumbbell total\_accel\_dumbbell 0.31819483  
## gyros\_belt\_y gyros\_belt\_y 0.29864980  
## gyros\_dumbbell\_x gyros\_dumbbell\_x 0.27228520  
## pitch\_dumbbell pitch\_dumbbell 0.26058358  
## accel\_arm\_y accel\_arm\_y 0.23402228  
## accel\_belt\_y accel\_belt\_y 0.21398937  
## accel\_arm\_z accel\_arm\_z 0.19671268  
## gyros\_forearm\_z gyros\_forearm\_z 0.18593082  
## total\_accel\_forearm total\_accel\_forearm 0.17746179  
## gyros\_dumbbell\_z gyros\_dumbbell\_z 0.13236427  
## magnet\_forearm\_y magnet\_forearm\_y 0.12613434  
## accel\_arm\_x accel\_arm\_x 0.10939682  
## total\_accel\_arm total\_accel\_arm 0.09811384  
## gyros\_arm\_x gyros\_arm\_x 0.06985288  
## accel\_forearm\_y accel\_forearm\_y 0.06034628  
## total\_accel\_belt total\_accel\_belt 0.00000000  
## gyros\_belt\_x gyros\_belt\_x 0.00000000  
## accel\_belt\_x accel\_belt\_x 0.00000000  
## pitch\_arm pitch\_arm 0.00000000  
## gyros\_arm\_z gyros\_arm\_z 0.00000000  
## yaw\_dumbbell yaw\_dumbbell 0.00000000  
## yaw\_forearm yaw\_forearm 0.00000000  
## gyros\_forearm\_x gyros\_forearm\_x 0.00000000  
## gyros\_forearm\_y gyros\_forearm\_y 0.00000000

#Predictions on real data - Final Model  
testData$RFfinal <- predict(RandomForestModel, testData, type="response", decision.values = TRUE, probability = TRUE)  
testData$RFfinal

## [1] 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

###REPEAT SETTING MODELS WITH DIFFERENT SEEDS TO SEE IF THE OUTCOME IS THE SAME!!!