Machine Learning Final Project

Eric Tsibertzopoulos

4/7/2017

Project Background

Using devices such as Jawbone Up, Nike FuelBand, and Fitbit it is now possible to collect a large amount of data about personal activity relatively inexpensively. One thing that people using such devices regularly do is quantify how much of a particular activity they do, but they rarely quantify how well they do it.

Project Objective

In this project, our goal is to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants and try to predict 5 distinct activities using 160 or less predictor variables. This will be a classification problem since the response variable is a factor containing 5 levels (A, B, C, D, E) that corresponse to five unique activities.

Data Source

The training and test data sets for this project come from: http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har).

R Packages for the Project

The following R packages were utilized in this project.

```
library(caret)
library(rpart)
library(randomForest)
library(gbm)
library(reshape2)
library(corrplot)
library(rpart.plot)
```

Data Sets

```
#training<-read.csv(url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"))
#The above command takes too long to download the data, we will use local data files.
training<-read.csv("~/Downloads/ML/Project/pml-training.csv")
testing<-read.csv("~/Downloads/ML/Project/pml-testing.csv")
set.seed(123)
```

The dimensions of the training and testing sets are: 19622, 160 and 20, 160 respectivelly. A training data set with 160 variables and 19K observations could present some computing challenges for regular PCs. Typical corss-validation of Ensemble algorithms like Random Forests and Boosting algorithms, require a lot of resampling, build large numbers of trees and run over many iterations. We will need to do some work pre-processing the training and testing data sets in order to reduce dimensionality.

Exploratory Data Analysis

Before even attempting any Principal Component Analysis (PCA) to reduce the dimensionality of the training data set, we can attmept simple variable exlusions. Numeric variables with large ratios of NAs will not add any predictive value to our models. Caret also contains a function that can identify Near-Zero-Value variables that need to be excluded. A simple str() command on the training set reveals some other variables (indexes and dates) that will have no effect in modeling.

Removal of obvious, useless variables

Variables like X, user_name, raw_timestamp_part_1, raw_timestamp_part_2, cvtd_timestamp will add no value to a predictive model.

```
removeTrainCols<-which(colnames(training) %in% c("X", "user_name", "raw_timestamp_par
t_1", "raw_timestamp_part_2", "cvtd_timestamp", "num_window"))
training<-training[,-removeTrainCols]
dim(training)
```

```
## [1] 19622   154
```

```
removeTestCols<-which(colnames(testing) %in% c("X", "user_name", "raw_timestamp_part_
1", "raw_timestamp_part_2", "cvtd_timestamp", "num_window"))
testing<-testing[,-removeTestCols]
dim(testing)
```

```
## [1]  20 154
```

Removal of Near-Zero-Value variables

Let's use Caret to detect near-zero-value variables.

```
insignificantVariables<-nearZeroVar(training)
insignificantVariables
```

```
##  [1]   1   6   7   8   9  10  11  14  17  20  45  46  47  48  49  50  51
## [18]  52  53  63  64  65  66  67  68  69  72  73  75  76  81  82  83  84
## [35]  85  86  89  92  95 119 120 121 122 123 124 125 127 128 130 131 133
## [52] 136 137 138 139 140 141 142 143 144
```

```
training<-training[,-insignificantVariables]
dim(training)
```

```
## [1] 19622    94
```

```
testing<-testing[,-insignificantVariables]
dim(testing)
```

```
## [1] 20 94
```

Removal of features with large NA ratios

A simple str() still reveals a lot of variables with NAs.

```
Cols_withNAs<-apply(training, 2, function(c){any(is.na(c))})
toRemove<-which(colnames(training) %in% colnames(training[,Cols_withNAs[Cols_withNAs=
T]]))
toRemove
```

```
##  [1]  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 34 44 45 46 47 51 52
## [24] 53 54 55 56 58 59 60 61 62 63 64 65 66 67 80 81 82 84
```

```
NA_Subset<-training[,toRemove]
NARatioForNAColumn<-function(x){table(is.na(x))[2]/length(x)}
NARatios<-apply(NA_Subset,2,function(e){NARatioForNAColumn(e)})
NARatios
```

```
##            max_roll_belt           max_picth_belt            min_roll_belt
##                0.9793089                0.9793089                0.9793089
##           min_pitch_belt      amplitude_roll_belt     amplitude_pitch_belt
##                0.9793089                0.9793089                0.9793089
##     var_total_accel_belt            avg_roll_belt         stddev_roll_belt
##                0.9793089                0.9793089                0.9793089
##            var_roll_belt           avg_pitch_belt        stddev_pitch_belt
##                0.9793089                0.9793089                0.9793089
##           var_pitch_belt             avg_yaw_belt          stddev_yaw_belt
##                0.9793089                0.9793089                0.9793089
##             var_yaw_belt            var_accel_arm            max_picth_arm
##                0.9793089                0.9793089                0.9793089
##              max_yaw_arm              min_yaw_arm        amplitude_yaw_arm
##                0.9793089                0.9793089                0.9793089
##        max_roll_dumbbell       max_picth_dumbbell        min_roll_dumbbell
##                0.9793089                0.9793089                0.9793089
##       min_pitch_dumbbell  amplitude_roll_dumbbell amplitude_pitch_dumbbell
##                0.9793089                0.9793089                0.9793089
##       var_accel_dumbbell        avg_roll_dumbbell     stddev_roll_dumbbell
##                0.9793089                0.9793089                0.9793089
##        var_roll_dumbbell       avg_pitch_dumbbell    stddev_pitch_dumbbell
##                0.9793089                0.9793089                0.9793089
##       var_pitch_dumbbell         avg_yaw_dumbbell      stddev_yaw_dumbbell
##                0.9793089                0.9793089                0.9793089
##         var_yaw_dumbbell        max_picth_forearm        min_pitch_forearm
##                0.9793089                0.9793089                0.9793089
##  amplitude_pitch_forearm        var_accel_forearm
##                0.9793089                0.9793089
```

Since the NA ratios of the detected columns exceeds 95% they can be removed from both training and testing sets.

```
training<-training[,-toRemove]
testing<-testing[,-toRemove]
```

The dimensions of the training and testing sets are now: 19622, 53 and 20, 53 respectivelly. The testing data set does not contian the classe response variable. However it contains an index variable prblem_id that does not exist in training. We need to remove problem_id from tresting.

```
testing<-testing[,-which(colnames(testing)%in%c("problem_id"))]
colnames(testing) %in% colnames(training)
```

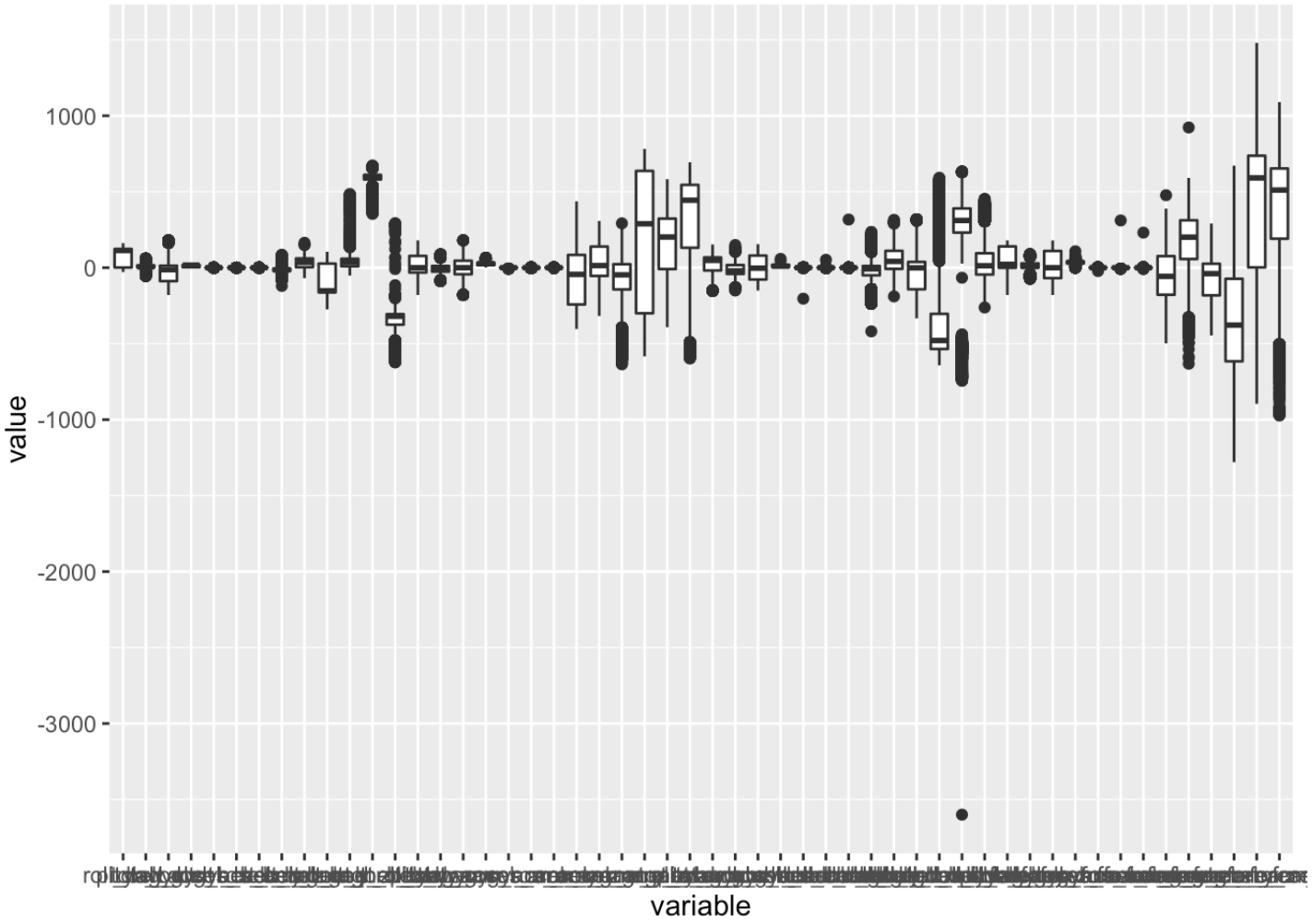
```
##  [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [15] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [29] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [43] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

```
dim(testing)
```

```
## [1] 20 52
```

Classification and regression algorithms could be affected by large variance in predictor variables. Perdictor variables with very large values could bias the performance of the model. Let's investigate if we need to center and scale our numeric variables in the training data set.

```
numerics<-training[,-53]
meltedNumercis<-melt(numerics)
ggplot(meltedNumercis, aes(x=variable, y=value))+geom_boxplot()
```



Based on the above box-plot, I recommend we transform, center and scale the numeric predictors. Additionally one can look at the Skewness of such variables and apply the necessary tranforms. In this project we will only center and scale the predictors.

Data Preparation for Modeling

We are going to use the testing set from the original data source, to Evaluate predictions for the 20 cases once we select a classification model that has the best accuracy. We will use the training data set for model training (10-fold cross-validation with at least 3 repetitions). We will make a 70/30 split of the training set for training and testing purposes.

```
inTrain<-createDataPartition(training$classe, p=0.7, list=F)
mTrainSet<-training[inTrain,]
mTestSet<-training[-inTrain,]
```

Now that we have the training/testing split we will apply the centering and scaling transformations on both sets, using caret.

```
preProcessed<-preProcess(mTrainSet, method = c("center", "scale"))
mTrainSet<-predict(preProcessed, mTrainSet[,-53])

mTrainSet$classe<-training[inTrain,]$classe  #re-assign the response variable to the transformed training set.
dim(mTrainSet)
```

```
## [1] 13737    53
```

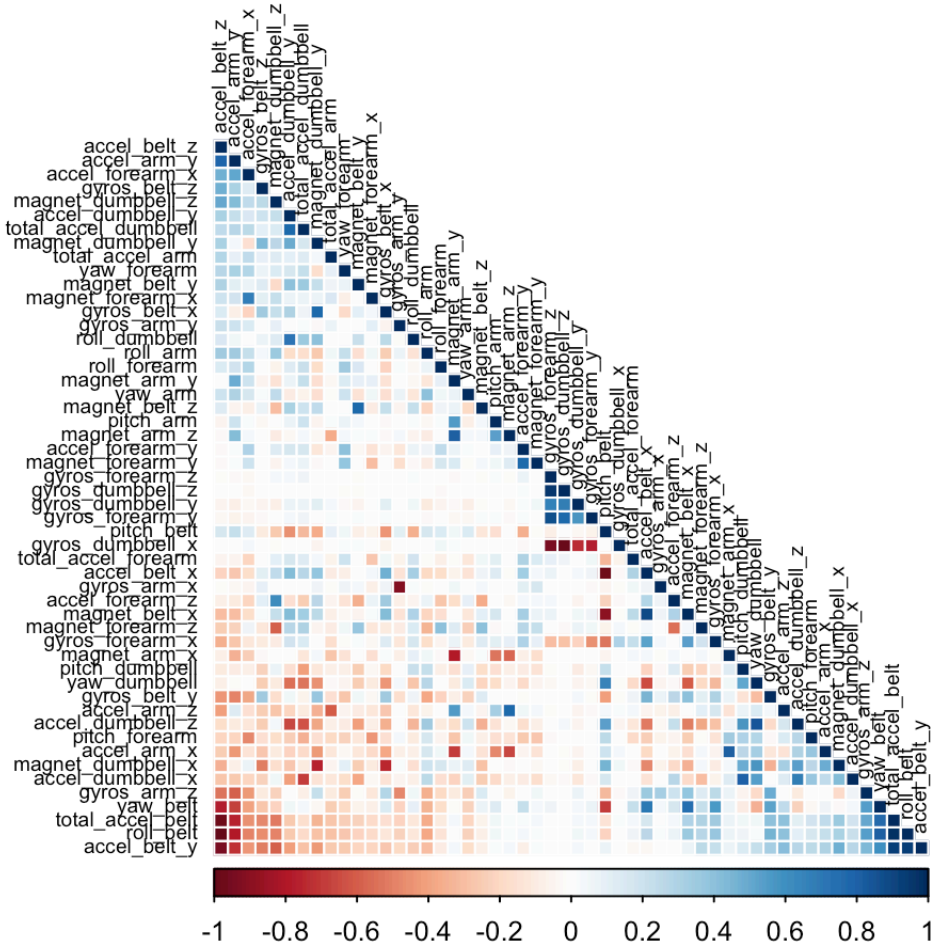
Apply transforms on the testing data set as well.

```
mTestSet<-predict(preProcessed, mTestSet)
dim(mTestSet)
```

```
## [1] 5885   53
```

Finally, before start modeling, let's take a look at possible correlations among the numeric predictors.

```
corMat <- cor(mTrainSet[, -53])
corrplot(corMat, order = "FPC", method = "color", type = "lower", tl.cex = 0.6, tl.col = rgb(0, 0, 0))
```



Although we took action to remove near-zero-value variables, we still see some strongly (postively and negatively) correlated predictors. We anticipate that some of the 52 predictors will not score highly on importance once the models are trained.

Models

Before we start exploring Decision Tree, Random Forest and Boosting models, let's enable parallel processing on 3 CPU cores.

```
library(doMC)
registerDoMC(cores=3)
modelEval<-list() #structure for storing model evaluation metrics
```

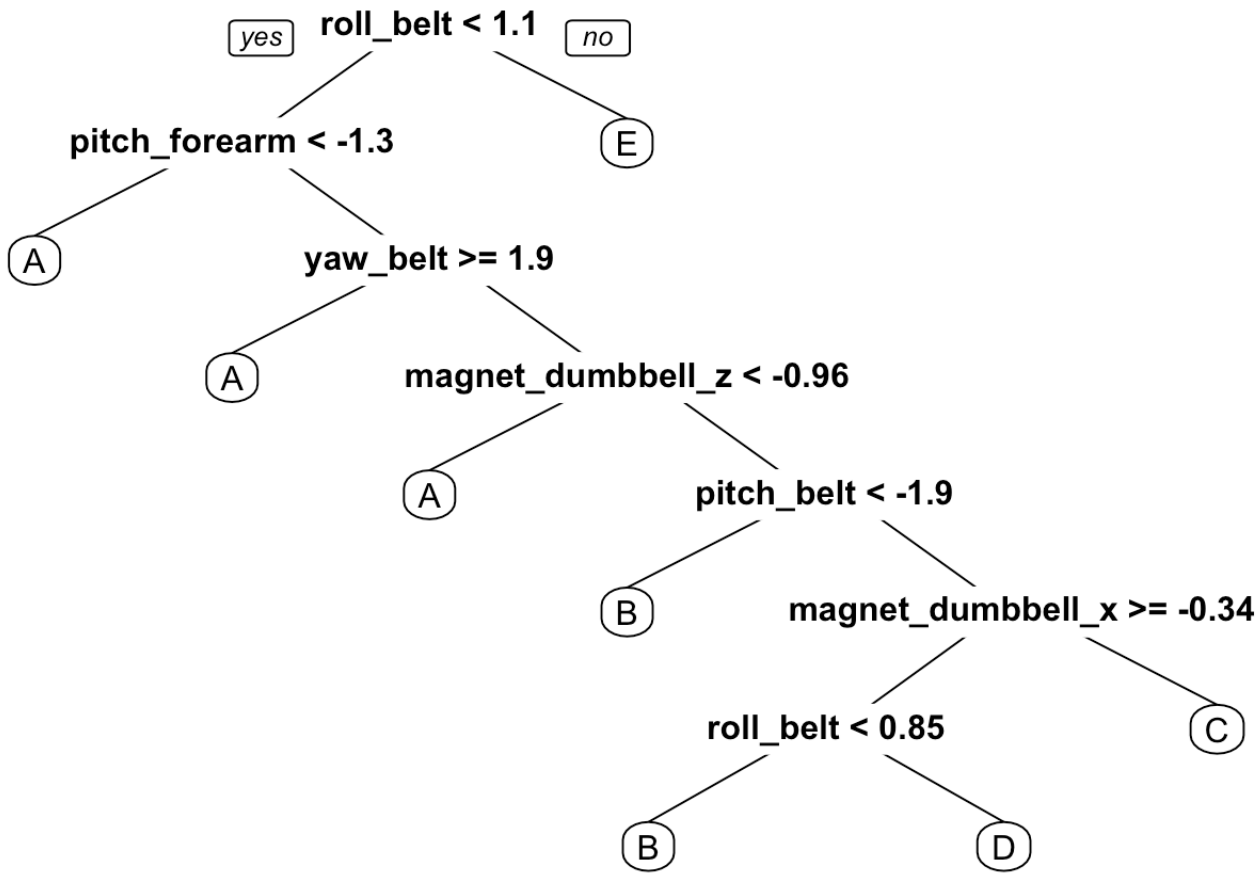
We will attemtp 3 tpyes of medels: 1) RPART Decision Tree, 2) Random Forest (Ensemble), and 3) GBM Boosting. For each of the modeling attempts we will use 10-Fold cross validation and we will attempt at least 3 iterations. The cross-validation parameter can be set up in the caret TrainControl object.

```
ctr<-trainControl(method="cv", number=10, repeats=3, allowParallel = T)
```

RPART Decision Tree

Single, cross validated RPART decision tree.

```
model_rpart<-train(classe~., method="rpart", data=mTrainSet, trControl=ctr)
rpart.plot(model_rpart$finalModel)
```



Generate predictions for the use cases in the testing set (split from the original Training set).

```
pred_rpart<-predict(model_rpart, mTestSet)
cm<-confusionMatrix(pred_rpart,mTestSet$classe)
cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1061  235   27   64   13
##          B  163  631   42  133  281
##          C  341  230  819  509  247
##          D  102   43  138  258   60
##          E    7    0    0    0  481
##
## Overall Statistics
##
##                Accuracy : 0.5523
##                  95% CI : (0.5394, 0.565)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4373
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.6338   0.5540   0.7982  0.26763  0.44455
## Specificity            0.9195   0.8696   0.7269  0.93030  0.99854
## Pos Pred Value         0.7579   0.5048   0.3816  0.42928  0.98566
## Neg Pred Value         0.8633   0.8904   0.9446  0.86639  0.88864
## Prevalence             0.2845   0.1935   0.1743  0.16381  0.18386
## Detection Rate         0.1803   0.1072   0.1392  0.04384  0.08173
## Detection Prevalence   0.2379   0.2124   0.3647  0.10212  0.08292
## Balanced Accuracy      0.7767   0.7118   0.7626  0.59897  0.72154
```

```
#model_rpart$finalModel$variable.importance
```

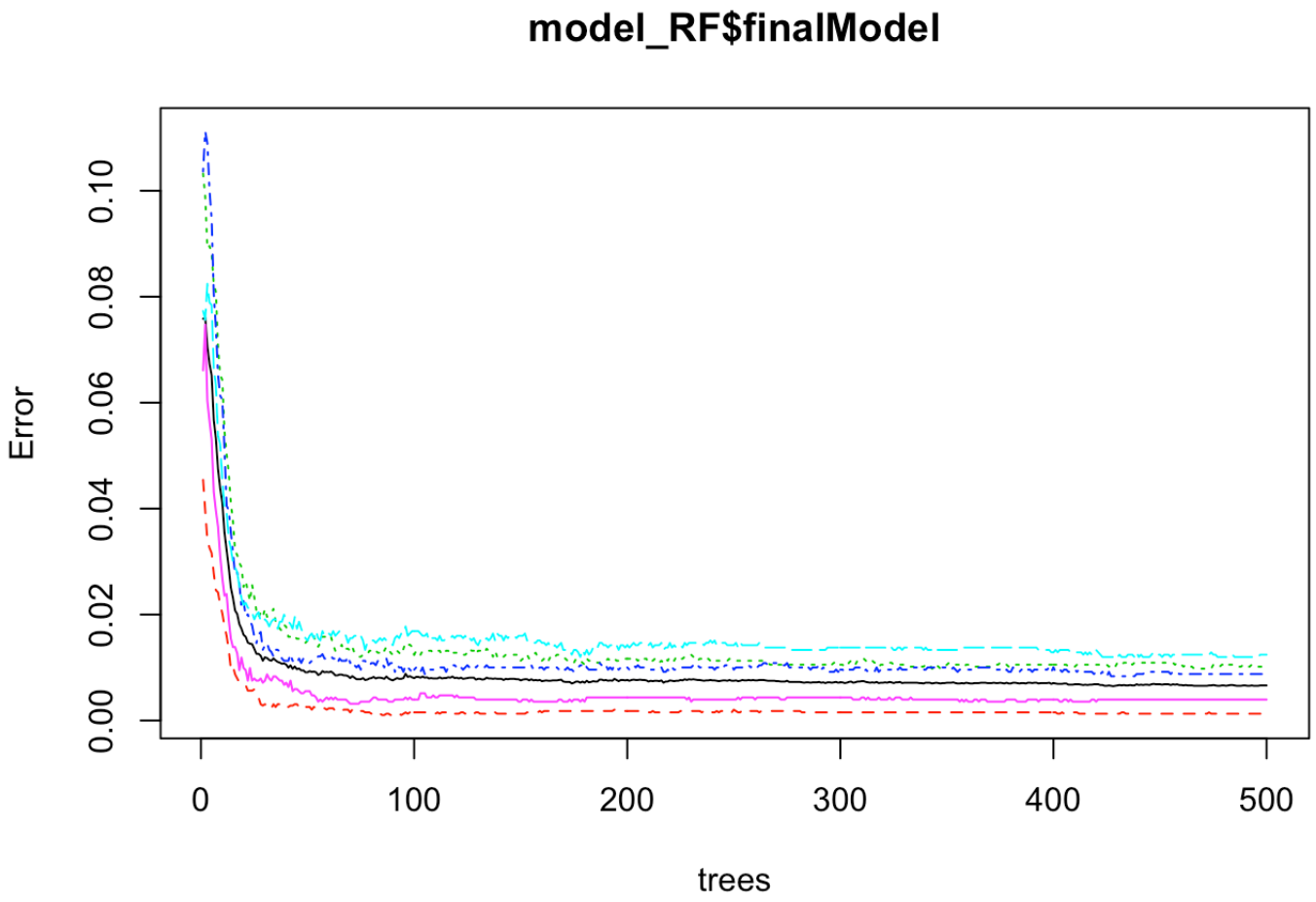
We see that the RPART classification accuracy is slightly over 55%. Add the RPART model in our result structure.

```
li<-list(Model="RPART Decision Tree, 10-fold Cross Validation", Accuracy=cm$overall[[
1]], Out_of_Sample_Error=1-cm$overall[[1]])
modelEval[length(modelEval)+1]<-list(li)
```

Random Forest, Ensemble Modeling

Random Forest model, 10-fold cross validated, building up to 500 decision trees across 3 iterations.

```
ctr<-trainControl(method="cv", number=10, repeats=3, allowParallel = T)
model_RF<-train(classe~., method="rf", data=mTrainSet, trControl=ctr, do.trace=F, ntr
ee=500)
plot(model_RF$finalModel)
```

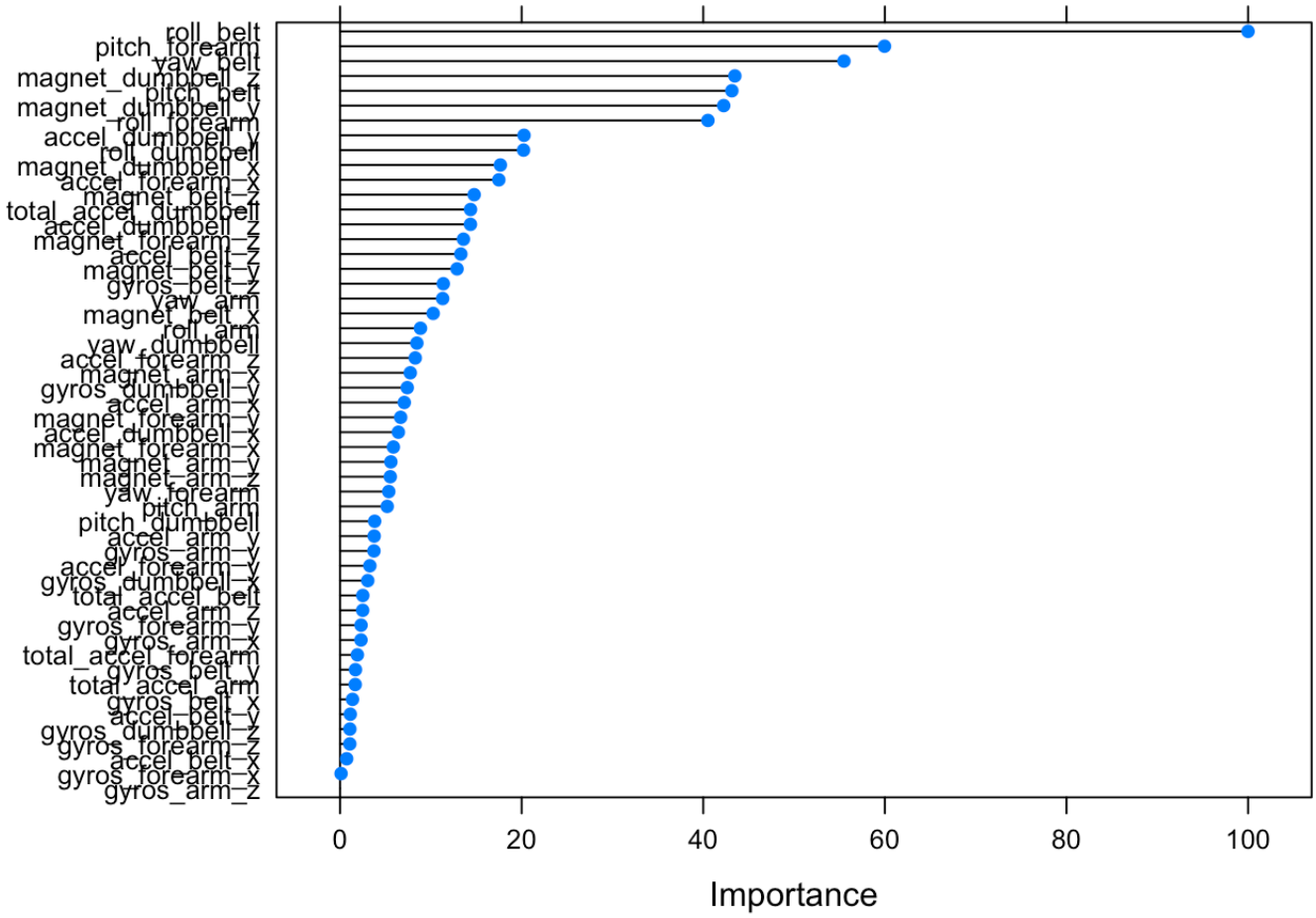


Generate predictions for the use cases in the testing set (split from the original Training set).

```
pred_RF<-predict(model_RF, mTestSet)
cm<-confusionMatrix(pred_RF,mTestSet$classe)
cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    8    0    0    0
##          B    1 1129    8    0    0
##          C    0    2 1015   18    2
##          D    0    0    3  945    1
##          E    0    0    0    1 1079
##
## Overall Statistics
##
##                Accuracy : 0.9925
##                  95% CI : (0.99, 0.9946)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9905
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   0.9912   0.9893   0.9803   0.9972
## Specificity            0.9981   0.9981   0.9955   0.9992   0.9998
## Pos Pred Value         0.9952   0.9921   0.9788   0.9958   0.9991
## Neg Pred Value         0.9998   0.9979   0.9977   0.9962   0.9994
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2843   0.1918   0.1725   0.1606   0.1833
## Detection Prevalence   0.2856   0.1934   0.1762   0.1613   0.1835
## Balanced Accuracy      0.9988   0.9947   0.9924   0.9897   0.9985
```

```
#t<-getTree(model_RF$finalModel, 27, labelVar = T)
variableImportance<-varImp(model_RF)
plot(variableImportance)
```




```
##Book keeping
li<-list(Model="Random Forest, 10-fold Cross Validation", Accuracy=cm$overall[[1]], O
ut_of_Sample_Error=1-cm$overall[[1]])
modelEval[length(modelEval)+1]<-list(li)
```

Gradient Boosting, Ensemble Modeling

Gradient Boosting model, 10-fold crossvalidated, over 3 repetitions.

```
ctr<-trainControl(method="repeatedcv", number=10, repeats=3, allowParallel = T)
model_GBM<-train(classe~., method="gbm", data=mTrainSet, trControl=ctr)
```

```
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.2371
##      2        1.4586             nan     0.1000    0.1636
##      3        1.3553             nan     0.1000    0.1244
##      4        1.2775             nan     0.1000    0.0999
##      5        1.2134             nan     0.1000    0.0921
##      6        1.1552             nan     0.1000    0.0769
##      7        1.1071             nan     0.1000    0.0758
##      8        1.0593             nan     0.1000    0.0601
##      9        1.0224             nan     0.1000    0.0586
##     10        0.9863             nan     0.1000    0.0442
##     20        0.7532             nan     0.1000    0.0210
##     40        0.5333             nan     0.1000    0.0113
##     60        0.4094             nan     0.1000    0.0098
##     80        0.3269             nan     0.1000    0.0048
##    100        0.2706             nan     0.1000    0.0026
##    120        0.2274             nan     0.1000    0.0031
##    140        0.1919             nan     0.1000    0.0021
##    150        0.1772             nan     0.1000    0.0015
```

```
model_GBM$results
```

```
##   shrinkage interaction.depth n.minobsinnode n.trees  Accuracy     Kappa
## 1       0.1                 1             10      50 0.7520307 0.6858127
## 4       0.1                 2             10      50 0.8531695 0.8139698
## 7       0.1                 3             10      50 0.8953420 0.8675268
## 2       0.1                 1             10     100 0.8186402 0.7704562
## 5       0.1                 2             10     100 0.9063589 0.8814913
## 8       0.1                 3             10     100 0.9416171 0.9261291
## 3       0.1                 1             10     150 0.8512047 0.8117143
## 6       0.1                 2             10     150 0.9304552 0.9119951
## 9       0.1                 3             10     150 0.9597193 0.9490417
##    AccuracySD     KappaSD
## 1 0.013336544 0.016943001
## 4 0.009546314 0.012141872
## 7 0.007982079 0.010119317
## 2 0.009890002 0.012543261
## 5 0.007746015 0.009804454
## 8 0.006357185 0.008044202
## 3 0.009716351 0.012326093
## 6 0.006599304 0.008351969
## 9 0.005759413 0.007287514
```

Generate predictions for the use cases in the testing set (split from the original Training set).

```
pred_GBM<-predict(model_GBM, mTestSet)
cm<-confusionMatrix(pred_GBM,mTestSet$classe)
cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1647   48    0    2    7
##          B   16 1063   37    2    7
##          C    9   26  976   43    5
##          D    1    2   11  912    9
##          E    1    0    2    5 1054
##
## Overall Statistics
##
##                Accuracy : 0.9604
##                  95% CI : (0.9551, 0.9652)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9499
##  Mcnemar's Test P-Value : 3.402e-09
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9839   0.9333   0.9513   0.9461   0.9741
## Specificity            0.9865   0.9869   0.9829   0.9953   0.9983
## Pos Pred Value         0.9665   0.9449   0.9216   0.9754   0.9925
## Neg Pred Value         0.9935   0.9840   0.9896   0.9895   0.9942
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2799   0.1806   0.1658   0.1550   0.1791
## Detection Prevalence   0.2895   0.1912   0.1799   0.1589   0.1805
## Balanced Accuracy      0.9852   0.9601   0.9671   0.9707   0.9862
```

```
#Book keeping
li<-list(Model="Boosting GBM, 10-fold Cross Validation", Accuracy=cm$overall[[1]], Ou
t_of_Sample_Error=1-cm$overall[[1]])
modelEval[length(modelEval)+1]<-list(li)
```

Winning Model and Case Predictions

Model	Accuracy	Out_of_Sample_Error
RPART Decision Tree, 10-fold Cross Validation	0.5522515	0.4477485

Random Forest, 10-fold Cross Validation	0.9925234	0.007476636
Boosting GBM, 10-fold Cross Validation	0.9604078	0.03959218

The Random Forest model has the best cross-validated performance. Its accuracy is: 0.9925234. The out-of-sample error is 1-0.9925234 = 0.0074766. We will select this model to predict (model Evaluation) the original 20 test cases in the Testing data set.

First we will have to apply the same transformations (center and scale) on the evaluation data set.

```
transformed_TestCases<-predict(preProcessed ,testing)
```

Now we can predict the evaluation cases.

```
pred_testCases<-predict(model_RF, transformed_TestCases)
pred_testCases
```

```
##  [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
```