Practical Machine Learning - Prediction Assignment

Author: Karthik Muthuveeramani

Executive Summary

As part of this assignment, we use the Human Activity Recognition(HAR) dataset, from http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har, that describes the exercise activities of 6 participants from accelerometers on the belt, forearm, arm and dumbbell. The classe variable is the outcome variable that shows how well they did the exercise by classifying them as A, B, C, D and E

Class A - exactly according to the specification

Class B - throwing the elbows to the front

Class C - lifting the dumbbell only halfway

Class D - lowering the dumbbell only halfway

Class E - throwing the hips to the front

We have been given the training and testing data sets. We used the training dataset to train the model and use the testing set to predict the outcomes. As part of this activity, we have read the training and testing data into R, then preprocessed the data and removed the variable that are not required or are incomplete, then we have split the training data into train and validation set to perform the validation and get the accuracy and out of sample error. Finally, we have used the train data and built 6 models - Decision Trees, Random Forests, Gradient Boosting, Support Vector Machines, Naive Bayes and LDA. We then decide on the best model based on accuracy and use it to predict the classe for the test data set.

Reading required Data and libraries

```r
library(caret)
library(rattle)
library(naivebayes)
set.seed(123)
train <- read.csv("./pml-training.csv")
test <- read.csv("./pml-testing.csv")
```

```r
dim(train)
```

```
## [1] 19622   160
```

```r
dim(test)
```

```
## [1]  20 160
```

Cleaning up the training data

We are removing those variables which have greater than 90% NA values in them. Also, we remove the 1st 7 columns which are more of user information, timestamp details. We then remove the variables with near zero variance.

```r
train_na_removed <- train
train_na_removed <- train[, which(colMeans(!is.na(train)) > 0.9)]
# Removing the 1st 7 columns which are more of user information, timestamp
train_na_removed <- train_na_removed[, -c(1:7)]
# Removing the variables with near zero variance.
nsv <- nearZeroVar(train_na_removed)
train_na_removed <- train_na_removed[,-nsv]
dim(train_na_removed)
```

```
## [1] 19622    53
```

We now have 53 variables in the train set.

Split into train and validation

```r
inTrain <- createDataPartition(y= train_na_removed$classe,
                               p=0.7,list=FALSE)
train_data <- train_na_removed[inTrain,]
validation_data <- train_na_removed[-inTrain,]
dim(train_data)
```

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

```r
dim(validation_data)
```

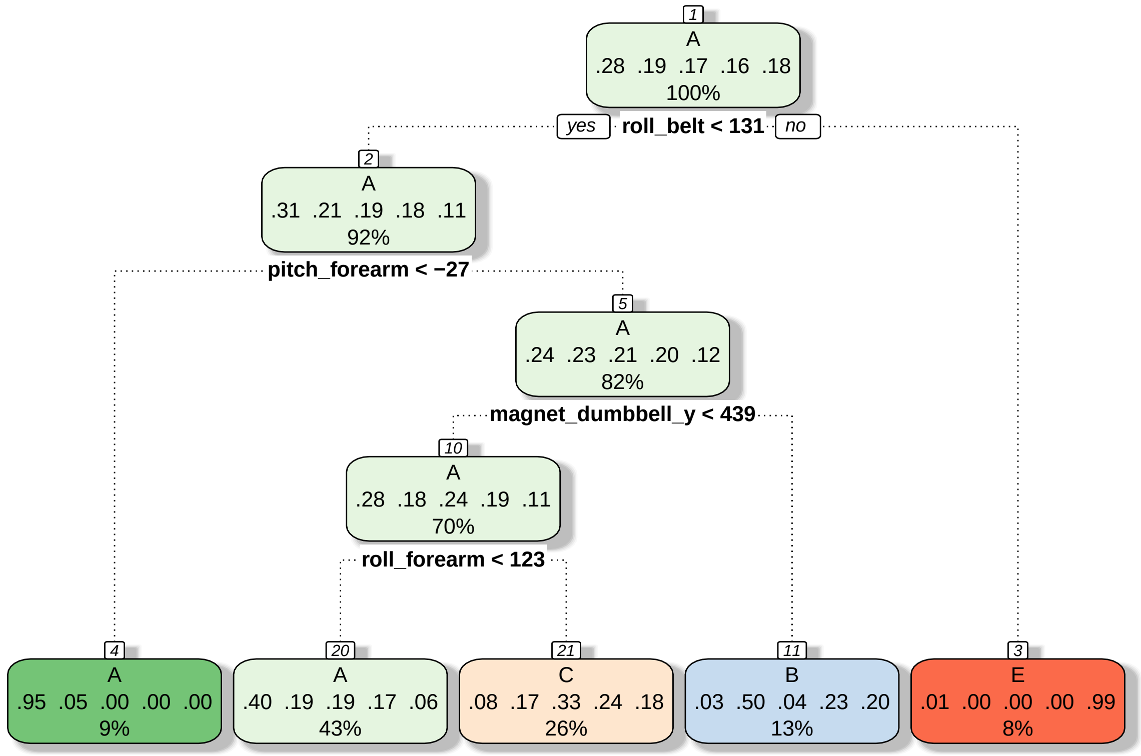
```
## [1] 5885   53
```

Fitting the Models

We apply the cross validation method to the trainControl parameters of all the models except the LDA. The resampling method used is cv and the number of iterations is 3

Decision Trees

```r
modFit_dtree <- train(classe ~ ., method = "rpart",
                      data = train_data,trControl = trainControl(method="cv",
                                                                 number=3))
fancyRpartPlot(modFit_dtree$finalModel)
```



Rattle 2021−May−21 23:12:27 Dell

```
#Predicting using decision trees
pred_dtree <- predict(modFit_dtree,validation_data)
conf_mat_dtree <- confusionMatrix(pred_dtree,factor(validation_data$classe))
conf_mat_dtree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1530  464  469  440  144
##          B   28  397   30  169  145
##          C  114  278  527  355  306
##          D    0    0    0    0    0
##          E    2    0    0    0  487
##
## Overall Statistics
##
##                Accuracy : 0.4997
##                  95% CI : (0.4869, 0.5126)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3464
##
##  Mcnemar's Test P-Value : NA
##
```

```
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9140  0.34855  0.51365   0.0000  0.45009
## Specificity            0.6398  0.92162  0.78329   1.0000  0.99958
## Pos Pred Value         0.5021  0.51625  0.33354      NaN  0.99591
## Neg Pred Value         0.9493  0.85496  0.88409   0.8362  0.88973
## Prevalence             0.2845  0.19354  0.17434   0.1638  0.18386
## Detection Rate         0.2600  0.06746  0.08955   0.0000  0.08275
## Detection Prevalence   0.5178  0.13067  0.26848   0.0000  0.08309
## Balanced Accuracy      0.7769  0.63508  0.64847   0.5000  0.72484
```

The accuracy of the decision tree model is 0.4997451

The out of sample error of the decision tree model is 0.5002549

Random Forest

```r
modFit_rf <- train(classe ~ ., method = "rf", data = train_data,trControl =
                       trainControl(method="cv", number=3))
#Predicting using random forests
pred_rf <- predict(modFit_rf,validation_data)
conf_mat_rf <- confusionMatrix(pred_rf,factor(validation_data$classe))
conf_mat_rf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    7    0    0    0
##          B    1 1124    5    0    0
##          C    0    8 1018   10    4
##          D    0    0    3  954    4
##          E    0    0    0    0 1074
##
## Overall Statistics
##
##                Accuracy : 0.9929
##                  95% CI : (0.9904, 0.9949)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.991
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   0.9868   0.9922   0.9896   0.9926
## Specificity            0.9983   0.9987   0.9955   0.9986   1.0000
## Pos Pred Value         0.9958   0.9947   0.9788   0.9927   1.0000
## Neg Pred Value         0.9998   0.9968   0.9983   0.9980   0.9983
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2843   0.1910   0.1730   0.1621   0.1825
## Detection Prevalence   0.2855   0.1920   0.1767   0.1633   0.1825
## Balanced Accuracy      0.9989   0.9928   0.9938   0.9941   0.9963
```

The accuracy of the random forest model is 0.9928632

The out of sample error of the random forest model is 0.0071368

Gradient Boosting

```
modFit_gbm <- train(classe ~ ., method = "gbm", data = train_data,trControl =
                        trainControl(method="cv", number=3), verbose = FALSE)
#Predicting using Gradient Boosting
pred_gbm <- predict(modFit_gbm,validation_data)
conf_mat_gbm <- confusionMatrix(pred_gbm,factor(validation_data$classe))
conf_mat_gbm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1650   34    0    3    3
##          B   12 1073   27    5    7
##          C    7   31  979   23   24
##          D    3    0   18  923   17
##          E    2    1    2   10 1031
##
## Overall Statistics
##
##                Accuracy : 0.9611
##                  95% CI : (0.9558, 0.9659)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9508
##
##  Mcnemar's Test P-Value : 4.95e-07
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9857   0.9421   0.9542   0.9575   0.9529
## Specificity            0.9905   0.9893   0.9825   0.9923   0.9969
## Pos Pred Value         0.9763   0.9546   0.9201   0.9605   0.9857
## Neg Pred Value         0.9943   0.9861   0.9903   0.9917   0.9895
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2804   0.1823   0.1664   0.1568   0.1752
## Detection Prevalence   0.2872   0.1910   0.1808   0.1633   0.1777
## Balanced Accuracy      0.9881   0.9657   0.9683   0.9749   0.9749
```

The accuracy of the Gradient Boosting model is 0.9610875

The out of sample error of the Gradient Boosting model is 0.0389125

Support Vector Machine

```r
modFit_svm <- train(classe ~ ., method = "svmLinear", data = train_data,trControl =
                        trainControl(method="cv", number=3))
#Predicting using svm
pred_svm <- predict(modFit_svm,validation_data)
conf_mat_svm <- confusionMatrix(pred_svm,factor(validation_data$classe))
conf_mat_svm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1558  149  109   63   59
##          B   27  837   83   42  154
##          C   34   64  796  112   85
##          D   43   18   23  703   52
##          E   12   71   15   44  732
##
## Overall Statistics
##
##                Accuracy : 0.7861
##                  95% CI : (0.7754, 0.7965)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7277
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9307   0.7349   0.7758   0.7293   0.6765
## Specificity            0.9098   0.9355   0.9393   0.9724   0.9704
## Pos Pred Value         0.8039   0.7323   0.7296   0.8379   0.8375
## Neg Pred Value         0.9706   0.9363   0.9520   0.9483   0.9302
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2647   0.1422   0.1353   0.1195   0.1244
## Detection Prevalence   0.3293   0.1942   0.1854   0.1426   0.1485
## Balanced Accuracy      0.9202   0.8352   0.8576   0.8508   0.8235
```

The accuracy of the Support Vector Machine model is 0.7860663

The out of sample error of the Support Vector Machine model is 0.2139337

Naive Bayes

```
modFit_nb <- train(classe ~ ., method = "naive_bayes", data = train_data,trControl =
                       trainControl(method="cv", number=3))
#Predicting using naive bayes
pred_nb <- predict(modFit_nb,validation_data)
conf_mat_nb <- confusionMatrix(pred_nb,factor(validation_data$classe))
conf_mat_nb
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1478  236  264  190   72
##          B   41  787   70    3   77
##          C   50   69  635  111   49
##          D   94   43   57  609   38
##          E   11    4    0   51  846
##
## Overall Statistics
##
##                Accuracy : 0.74
##                  95% CI : (0.7286, 0.7512)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6669
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8829   0.6910   0.6189   0.6317   0.7819
## Specificity            0.8190   0.9598   0.9426   0.9529   0.9863
## Pos Pred Value         0.6598   0.8047   0.6947   0.7241   0.9276
## Neg Pred Value         0.9462   0.9283   0.9213   0.9296   0.9525
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2511   0.1337   0.1079   0.1035   0.1438
## Detection Prevalence   0.3806   0.1662   0.1553   0.1429   0.1550
## Balanced Accuracy      0.8510   0.8254   0.7807   0.7923   0.8841
```

The accuracy of the Naive Bayes model is 0.740017

The out of sample error of the Naive Bayes model is 0.259983

LDA

```
modlda <- train(classe ~ ., data = train_data, method = "lda")
#predicting using lda
```

```
pred_lda <- predict(modlda,validation_data)
conf_mat_lda <- confusionMatrix(pred_lda,factor(validation_data$classe))
conf_mat_lda
```

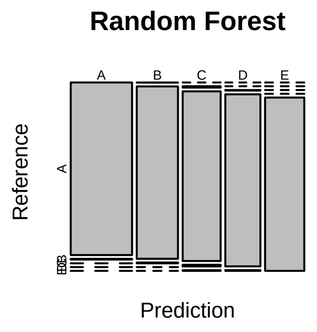
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1358  178  116   58   47
##          B   37  744  101   52  199
##          C  132  139  669  115  106
##          D  140   35  121  702  107
##          E    7   43   19   37  623
##
## Overall Statistics
##
##                Accuracy : 0.696
##                  95% CI : (0.6841, 0.7077)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6151
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8112   0.6532   0.6520   0.7282   0.5758
## Specificity            0.9052   0.9180   0.8987   0.9181   0.9779
## Pos Pred Value         0.7729   0.6567   0.5762   0.6353   0.8546
## Neg Pred Value         0.9234   0.9169   0.9244   0.9452   0.9110
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2308   0.1264   0.1137   0.1193   0.1059
## Detection Prevalence   0.2986   0.1925   0.1973   0.1878   0.1239
## Balanced Accuracy      0.8582   0.7856   0.7754   0.8232   0.7769
```

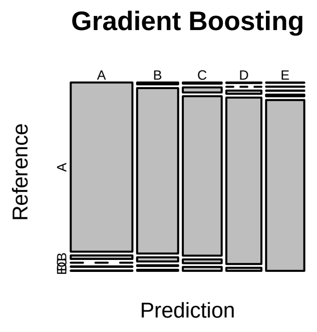
The accuracy of the LDA model is 0.6960068

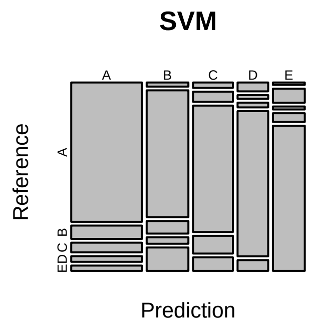
The out of sample error of the LDA model is 0.3039932

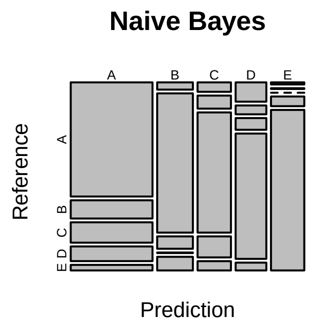
Plot the model outcomes

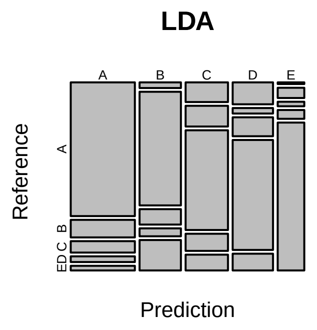
```
par(mfrow=c(2,3))
plot(conf_mat_rf$table,main="Random Forest")
plot(conf_mat_gbm$table,main="Gradient Boosting")
plot(conf_mat_svm$table,main="SVM")
plot(conf_mat_nb$table,main="Naive Bayes")
plot(conf_mat_lda$table,main="LDA")
plot(conf_mat_dtree$table,main="Decision Trees")
```

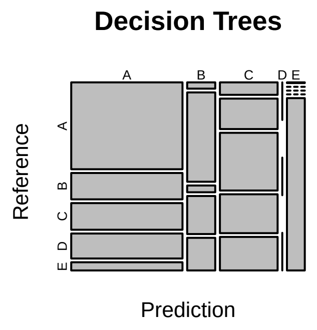












We can clearly see that the Random Forest model is the best one with an accuracy of 0.9928632

Prediction and Results

We use the Random Forest model to predict the classe for the 20 testing cases. The output is shown below:

```r
predict_test <- predict(modFit_rf,newdata = test)
predict_test
```

```
##  [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
```