Applied Machine Learning Final Project

Synopsis

One thing people regularly quantify is how much they do, weather that be walking, running or any weighted exercise at the gym. Very rarely do people get the chance or are interested in tracking how well they do something. The goal of this project is to change that, using data collected from accelerometers on the belt, forearm, arm, and dumbbell of 6 participants we will try to build a predictive model that tells how how well a certain task was performed.

```r
library(caret)
```

```
## Loading required package: lattice

## Loading required package: ggplot2
```

```r
library(rpart)
library(rpart.plot)
library(RColorBrewer)
#library(RGtk2)
library(rattle)
```

```
## Loading required package: tibble

## Loading required package: bitops

## Rattle: A free graphical interface for data science with R.
## Version 5.4.0 Copyright (c) 2006-2020 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```r
library(randomForest)
```

```
## randomForest 4.6-14

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

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:rattle':
##
##     importance

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

```r
library(gbm)
```

```
## Loaded gbm 2.1.8
```

```r
## load in the training and test sets
trainingdata = read.csv("pml-training.csv")
testdata= read.csv("pml-testing.csv")






trainingdata$classe= as.factor(trainingdata$classe)
```

Cleaning The Data

```r
#remove variable with little to no variance
non_zero_var <- nearZeroVar(trainingdata)

trainingdata <- trainingdata[, -non_zero_var]
testdata = testdata[, -non_zero_var]

#remove columns with too many missing values (>95%)
na_val_col <- sapply(trainingdata, function(x) mean(is.na(x))) > .95

trainingdata <- trainingdata[, -na_val_col]
testdata = testdata[, -na_val_col]

#remove all the non-numeric columns as they wont be needed in our model
trainingdata   <-trainingdata[,-c(1:7)]
testdata <-testdata[,-c(1:7)]

dim(trainingdata)
```

```
## [1] 19622    92
```

Data Partitioning

As per the courser recommendation, we will split the training data further to get out a training set and a test set to allow for the model to be tested befor exposing it to the final testing dataset.

```r
set.seed(10) #set seed for reproducibility

inTrain<- caret::createDataPartition(trainingdata$classe, p=0.6, list=FALSE)
model_training= trainingdata[inTrain, ]
model_testing= trainingdata[-inTrain, ]

dim(model_training)
```

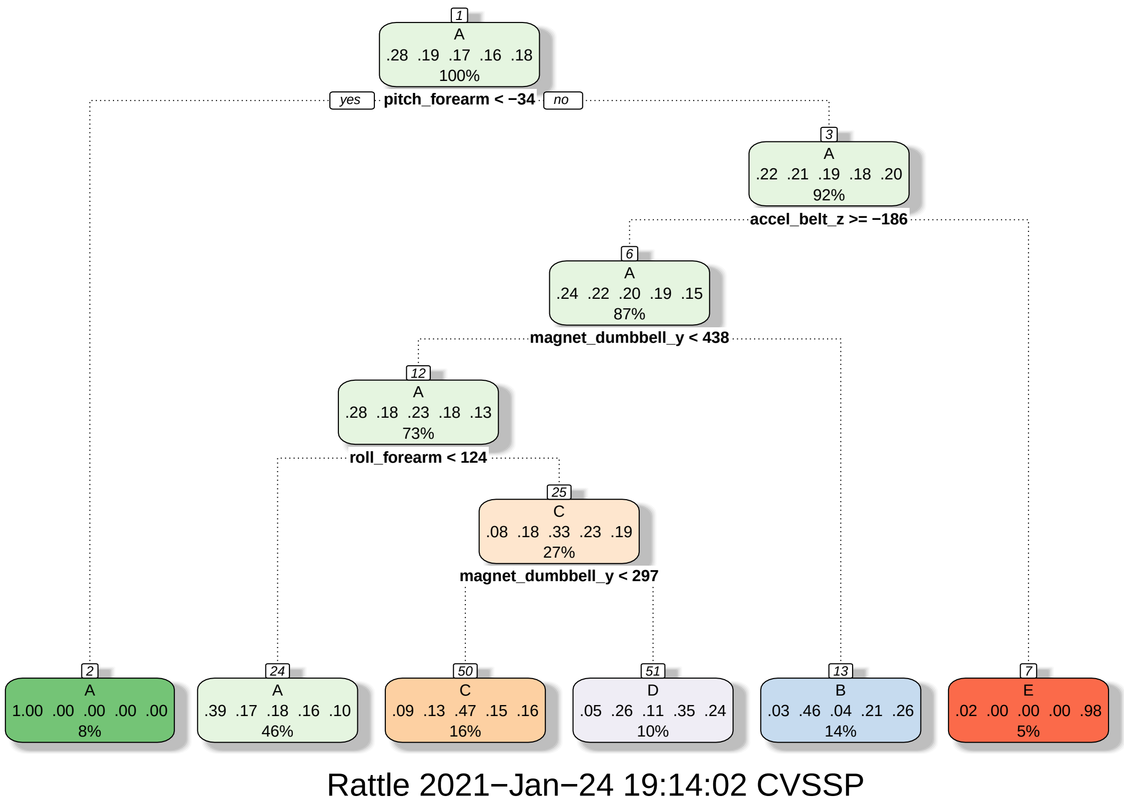
```
## [1] 11776    92
```

```r
dim(model_testing)
```

```
## [1] 7846   92
```

Decision Tree Model

```r
DT_modfit <- train(classe ~ ., data = model_training, method="rpart", na.action = na.rpart )
fancyRpartPlot(DT_modfit$finalModel)
```




```r
DT_prediction <- predict(DT_modfit, model_testing, na.action = na.pass)
confusionMatrix(DT_prediction, model_testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2028  625  633  576  327
##          B   38  529   42  225  291
##          C  110  163  607  194  213
##          D   50  201   86  291  191
```

```
##          E    6    0    0    0  420
##
## Overall Statistics
##
##                Accuracy : 0.4939
##                  95% CI : (0.4828, 0.505)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3381
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9086  0.34848  0.44371  0.22628  0.29126
## Specificity            0.6151  0.90582  0.89503  0.91951  0.99906
## Pos Pred Value         0.4841  0.47022  0.47164  0.35531  0.98592
## Neg Pred Value         0.9442  0.85285  0.88398  0.85840  0.86226
## Prevalence             0.2845  0.19347  0.17436  0.16391  0.18379
## Detection Rate         0.2585  0.06742  0.07736  0.03709  0.05353
## Detection Prevalence   0.5339  0.14339  0.16403  0.10438  0.05430
## Balanced Accuracy      0.7618  0.62715  0.66937  0.57290  0.64516
```

From the confusion matrix above we see that the accuracy is 49% which is not a high enough percentage for this decision tree model to be considered successful.

Gradient Boosting Model

```r
r_forest=model_training[ , colSums(is.na(model_training)) == 0]
set.seed(25621)
gbm_model<- train(classe~., data=r_forest, method="gbm", verbose= FALSE)
gbm_model$finalmodel
```

```
## NULL
```

```r
gbm_prediction<- predict(gbm_model, model_testing)
gbm_cm<-confusionMatrix(gbm_prediction, model_testing$classe)
gbm_cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2182   52    0    1    2
##          B   28 1410   62   16   10
##          C   18   45 1278   54   19
##          D    3    6   25 1203   22
##          E    1    5    3   12 1389
##
```

```
## Overall Statistics
##
##                Accuracy : 0.9511
##                  95% CI : (0.9461, 0.9557)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9381
##
##  Mcnemar's Test P-Value : 2.707e-09
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9776   0.9289   0.9342   0.9355   0.9632
## Specificity            0.9902   0.9817   0.9790   0.9915   0.9967
## Pos Pred Value         0.9754   0.9240   0.9038   0.9555   0.9851
## Neg Pred Value         0.9911   0.9829   0.9860   0.9874   0.9918
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2781   0.1797   0.1629   0.1533   0.1770
## Detection Prevalence   0.2851   0.1945   0.1802   0.1605   0.1797
## Balanced Accuracy      0.9839   0.9553   0.9566   0.9635   0.9800
```

As seen from the model above we are able to predict the classes with an accuracy of 95% percent which is satisfactory for an in sample testing rate.

Random Forest Model

```r
RF_modfit <- train(classe ~ ., data = r_forest, method = "rf", ntree = 100)
RF_prediction <- predict(RF_modfit, model_testing)
RF_pred_conf <- confusionMatrix(RF_prediction, model_testing$classe)
RF_pred_conf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2227   15    0    0    0
##          B    3 1496   12    0    0
##          C    1    6 1350   22    2
##          D    0    0    5 1263    1
##          E    1    1    1    1 1439
##
## Overall Statistics
##
##                Accuracy : 0.991
##                  95% CI : (0.9886, 0.9929)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9886
```

```
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9978   0.9855   0.9868   0.9821   0.9979
## Specificity            0.9973   0.9976   0.9952   0.9991   0.9994
## Pos Pred Value         0.9933   0.9901   0.9776   0.9953   0.9972
## Neg Pred Value         0.9991   0.9965   0.9972   0.9965   0.9995
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2838   0.1907   0.1721   0.1610   0.1834
## Detection Prevalence   0.2858   0.1926   0.1760   0.1617   0.1839
## Balanced Accuracy      0.9975   0.9916   0.9910   0.9906   0.9986
```

As we can see from above the random forest model has an excellent accuracy of 99.1% and so this is the model of choice to use on the unseen data and to try and predict which classe each set of information belongs to.

```
Final_prediction = predict(RF_modfit, testdata)
Final_prediction
```

```
##  [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
```

Conclusion

We were able to use accelerometer data to build different models that can predict the quality of movement and how well an exercise is being performed. It was found that the random forest model performed the best out of all the machine learning models and we applied this model to the unseen data to produce some predictions.