Machine Learning Project from Coursera Specialization

Assignment

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. These type of devices are part of the quantified self movement - a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. One thing that people regularly do is quantify how much of a particular activity they do, but they rarely quantify how well they do it. In this project, your goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. More information is available from the website here: http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset).

Data Loading and Cleaning

The data was downloaded from the given URLs.

```
library(caret)
```

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

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

```
library(rattle)
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.2.0 Copyright (c) 2006-2018 Togaware Pty Ltd.
## Geben Sie 'rattle()' ein, um Ihre Daten mischen.
```

```
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
```

```
training <- read.csv(url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"))
testing <- read.csv(url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"))
```

The columns that have missing values in either the training or the testing sets are found through the following code. Only the columns that have no NAs are kept in the new dataset. Additionally all columns that are not valuable for the prediction are removed

```
NAdatatrain <- sapply(1:ncol(training), function(x) sum(is.na(training[,x])))
NAdatatest <- sapply(1:ncol(testing), function(x) sum(is.na(testing[,x])))
keeptrain <- names(training)[NAdatatrain==0]
keeptest <- names(testing)[NAdatatest==0]
keep <- names(training)[names(training) %in% keeptrain & names(training) %in% keeptest]
training2 <- training[, c(keep, "classe")]
training2 <- training2[, -c(1:7)]
testing2 <- testing[, c(keep, "problem_id")]
testing2 <- testing2[, -c(1:7)]
```

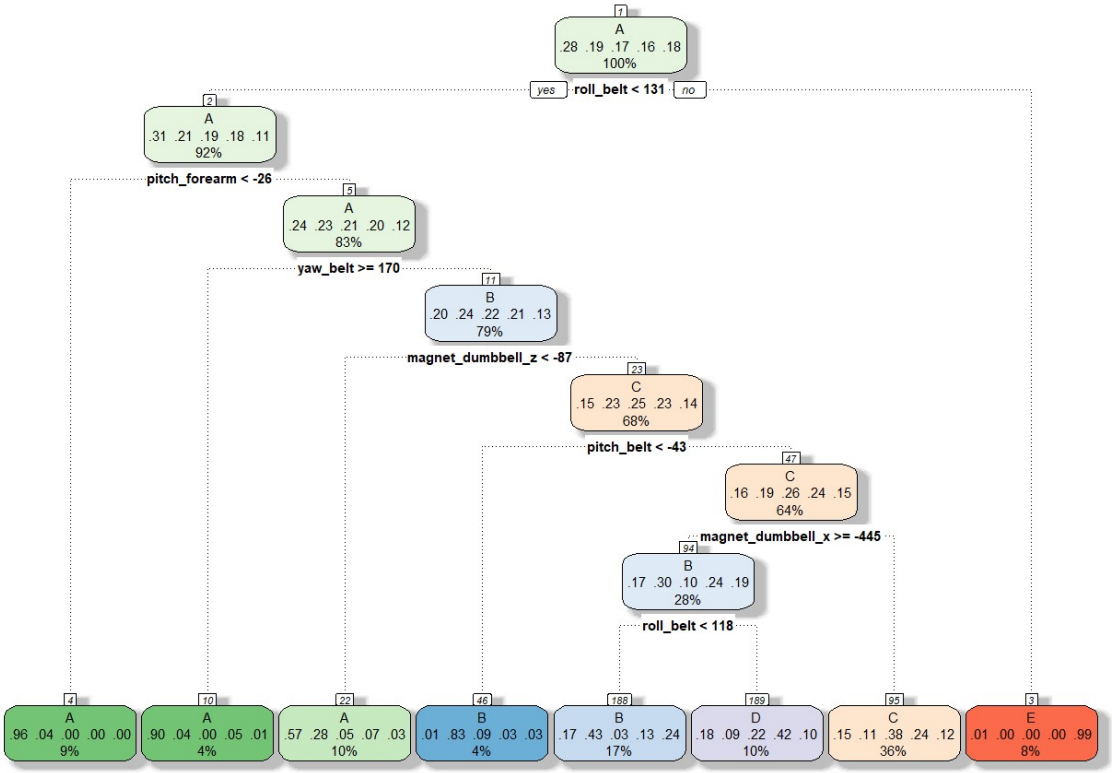
As the testing dataset has no column “classe” so it can not be used to evaluate the model. Therefore the training dataset is split into train an test dataset. The train set will be 70% of the training dataset.

```
set.seed(123)
trainindex <- createDataPartition(training2$classe, p=0.7, list=F)
train <- training2[trainindex,]
test <- training2[-trainindex,]
```

Decision Tree

As decision trees are great for classification problems and easy to interpret and understand we are starting the Modelling process with a decision tree. We use the caret package to train the model.

```
set.seed(234)
model1 <- train(classe ~ ., data=train, method="rpart")
fancyRpartPlot(model1$finalModel)
```



Rattle 2019-Jan-27 13:51:51 maier

In the following the the test dataset will be predicted with the decision tree and evaluated using the confusionMatrix-function.

```
pred1 <- predict(model1, test)
CM1 <- confusionMatrix(pred1, test$classe)
CM1
```

```
## 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
```

The accuracy of the model is 0.5522515, which is quite low.

Random Forest

Because the decision tree did not work well for the test dataset a random forest model will be trained an evaluated.

```
model2 <- randomForest(classe ~ ., data=train, method="rf")
pred2 <- predict(model2, test)
CM2 <- confusionMatrix(pred2, test$classe)
CM2
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    4    0    0    0
##          B    1 1135   12    0    0
##          C    0    0 1014   12    0
##          D    0    0    0  951    0
##          E    0    0    0    1 1082
##
## Overall Statistics
##
##                Accuracy : 0.9949
##                  95% CI : (0.9927, 0.9966)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9936
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   0.9965   0.9883   0.9865   1.0000
## Specificity            0.9991   0.9973   0.9975   1.0000   0.9998
## Pos Pred Value         0.9976   0.9887   0.9883   1.0000   0.9991
## Neg Pred Value         0.9998   0.9992   0.9975   0.9974   1.0000
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2843   0.1929   0.1723   0.1616   0.1839
## Detection Prevalence   0.2850   0.1951   0.1743   0.1616   0.1840
## Balanced Accuracy      0.9992   0.9969   0.9929   0.9933   0.9999
```

The ranodm forest model has a accuracy with 0.9949023. This is a fairly high accuracy.

Conclusion

The random forest model was used to evaluate the testing dataset and answer the questions to the quiz. As the accuracy is really good the predictions for the testing dataset were all correct.