Practical Machine Learning - Excercise Prediction

Farhan Atiq

July 30, 2017

Data Processing

First the data is downloaded via the URL given above. During the analysis it was noted that several values in the data can be considered “NA”, so these are now passed to the function reading in the data to help with clean up and speed the project up.

```
library(caret)
```

```
## 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(rpart.plot)
library(e1071)
```

Getting the data

```
trainlink <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testlink <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
training <- read.csv(url(trainlink), na.strings=c("NA","#DIV/0!",""))
test_grade<- read.csv(url(testlink), na.strings=c("NA","#DIV/0!",""))
```

Removing the first few columns as they have nothing to do with the movement we are predicting.

```
training <- training[,-c(1:7)]
test_grade <- test_grade[,-c(1:7)]
training$classe <- factor(training$classe)
```

Next,check for NA values and remove

```
sum(is.na(training))
```

```
## [1] 1925102
```

Eliminating many of the columns which are NA, leaving a little over 50 predictor values left to use during modeling.

```
notna <- sapply(training, function(x)all(!is.na(x)))
training <- training[,notna]
test_grade <- test_grade[,notna]
```

Removing additional variables based on the near zero variance function. Compare the model by creating another training set and test grading set.

```
removecol <- nearZeroVar(training, saveMetrics=TRUE)
training2 <- training[,!removecol$nzv== TRUE]

test_grade2 <- test_grade[,!removecol$nzv== TRUE]
```

We then split our training data in to seperate training and testing sets. The testing set that was downloaded above is used to grade the project, but we need a testing set to test our model on before grading.

```
inTrain = createDataPartition(y=training$classe, p=0.6, list=FALSE)
trainset = training[inTrain,]
testset= training[-inTrain,]
```

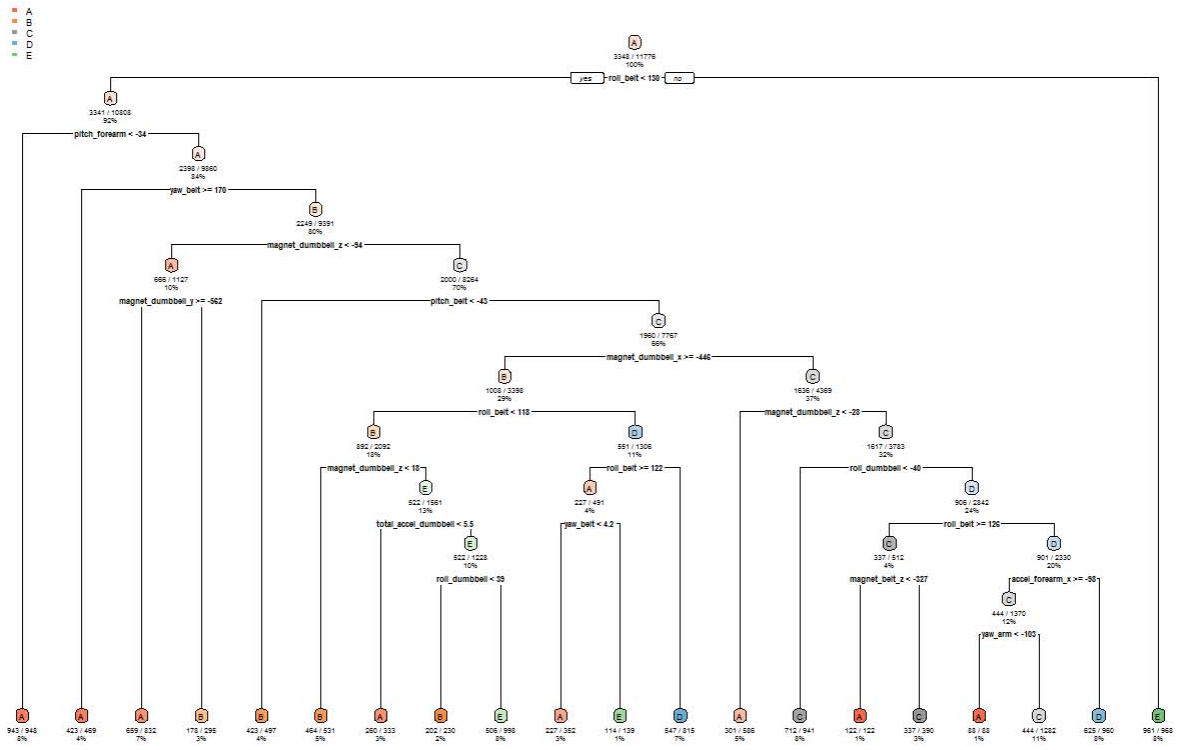
Model Building

Prediction model 1: Decision Tree

```
model1 <- rpart(classe ~ ., data=trainset, method="class")

prediction1 <- predict(model1, trainset, type = "class")

# Plot the Decision Tree
rpart.plot(model1, main="Classification Tree", extra=102, under=TRUE, faclen=0)
```



Prediction model 2: Random Forest

```
model2 <- randomForest(classe ~. , data=trainset, method="class")

# Predicting:
prediction2 <- predict(model2, trainset, type = "class")

# Test results on TestTrainingSet data set:
confusionMatrix(prediction2, trainset$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 3348    0    0    0    0
##          B    0 2279    0    0    0
##          C    0    0 2054    0    0
##          D    0    0    0 1930    0
##          E    0    0    0    0 2165
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9997, 1)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Prevalence   0.2843   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

Graded Predictions

Random Forest algorithm performed better than Decision Trees. Accuracy for Random Forest model was 0.995 (95% CI: (0.993, 0.997)) compared to Decision Tree model with 0.739 (95% CI: (0.727, 0.752)). The Random Forests model is choosen. The expected out-of-sample error is estimated at 0.005, or 0.5%.

Here is the final outcome based on the Prediction Model 2 (Random Forest) applied against the Testing dataset

```
predictfinal <- predict(model2, testset, type="class")
```

Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.