Machine Learning Prediction

Ammar Akram

10/25/2020

Loading the required libraries

```
library(lattice)
library(ggplot2)
library(caret)
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:ggplot2':
##
##     margin
```

```
library(rpart)
library(rpart.plot)
```

Data Load and Clean up

```
set.seed(123)
trainingset <- read.csv("pml-training.csv", na.strings=c("NA","#DIV/0!", ""))
testingset <- read.csv("pml-testing.csv", na.strings=c("NA","#DIV/0!", ""))
```

Perform Explorartory Data Analysis

```
#dim(trainingset)
#dim(testingset)
#summary(trainingset)
#summary(testingset)
#str(trainingset)
#str(testingset)
#head(trainingset)
#head(testingset)
```

Delete Columns with all missing vaues

```
trainingset<-trainingset[,colSums(is.na(trainingset)) == 0]
testingset <-testingset[,colSums(is.na(testingset)) == 0]
```

Delete the Variable irrelevent to our current Project

```
trainingset   <-trainingset[,-c(1:7)]
testingset <-testingset[,-c(1:7)]
```

Partition the data so that 75% of the training dataset into training and reamaining 25% to testing

```
traintrainset <- createDataPartition(y=trainingset$classe, p=0.75, list=FALSE)
TrainTrainingSet <- trainingset[traintrainset, ]
TestTrainingSet <- trainingset[-traintrainset, ]
```

The variable "classe" contains 5 levels: A,B,C,D and E. A plot of the outcome variable will allow us to see the frequency of each levels in the TrainTrainingSet data set and compare one another

```
qplot(TrainTrainingSet$classe, main="Plot of levels of variable classe within the TrainTrainingSet data
     xlab="classe", ylab="Frequency")
```

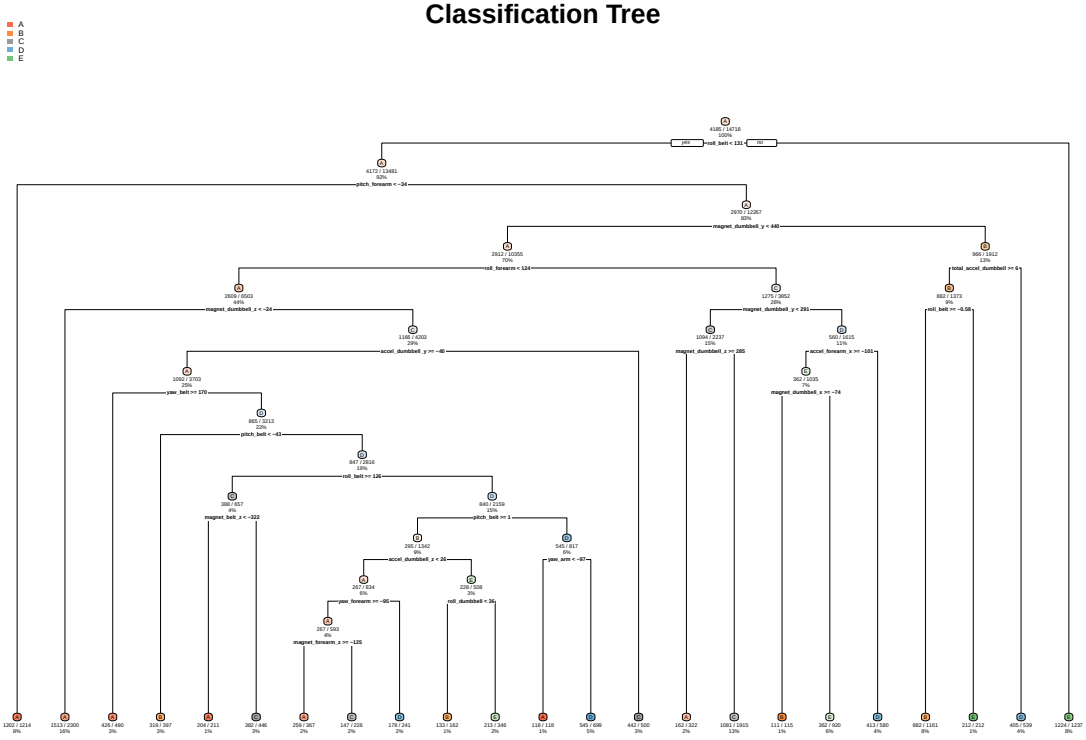
Based on this graph, we can see that each level frequency is within the same order of magnitude of each other.

Level A is most frequent and level D is least frequent

Prediction Model 1: Decision Tree

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

# Predicting
prediction1<- predict(model1, TestTrainingSet, type="class")
# Plot the decision tree
rpart.plot(model1, main="Classification Tree", extra=102, under=TRUE, faclen=0)
```



Test the result on TestTrainingSet data set:

```
confusionMatrix(prediction1, as.factor(TestTrainingSet$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1304  185   31  102   45
##          B   28  479   34   16   29
##          C   25  125  689  130  109
##          D   18   69   37  477   50
##          E   20   91   64   79  668
##
## Overall Statistics
##
##                Accuracy : 0.7376
##                  95% CI : (0.725, 0.7498)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6659
##
##  Mcnemar's Test P-Value : < 2.2e-16
```

```
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9348  0.50474   0.8058  0.59328   0.7414
## Specificity            0.8966  0.97295   0.9039  0.95756   0.9365
## Pos Pred Value         0.7822  0.81741   0.6391  0.73272   0.7245
## Neg Pred Value         0.9719  0.89115   0.9566  0.92311   0.9415
## Prevalence             0.2845  0.19352   0.1743  0.16395   0.1837
## Detection Rate         0.2659  0.09768   0.1405  0.09727   0.1362
## Detection Prevalence   0.3399  0.11949   0.2198  0.13275   0.1880
## Balanced Accuracy      0.9157  0.73884   0.8549  0.77542   0.8390
```

Prediction Model 2: Random Forest

```
model2 <- randomForest(as.factor(classe) ~. , data=TrainTrainingSet, method="class")
# Predicting
prediction2<- predict(model2, TestTrainingSet)
```

Test the result on TesttrainingSet data set:

```
confusionMatrix(as.factor(prediction2),as.factor( TestTrainingSet$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    0    0    0    0
##          B    0  948    9    0    0
##          C    0    1  846    6    0
##          D    0    0    0  798    2
##          E    0    0    0    0  899
##
## Overall Statistics
##
##                Accuracy : 0.9963
##                  95% CI : (0.9942, 0.9978)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9954
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9989   0.9895   0.9925   0.9978
## Specificity            1.0000   0.9977   0.9983   0.9995   1.0000
## Pos Pred Value         1.0000   0.9906   0.9918   0.9975   1.0000
```

```
## Neg Pred Value         1.0000   0.9997   0.9978   0.9985   0.9995
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1933   0.1725   0.1627   0.1833
## Detection Prevalence   0.2845   0.1951   0.1739   0.1631   0.1833
## Balanced Accuracy      1.0000   0.9983   0.9939   0.9960   0.9989
```

Decision on which Prediction Model to Use:

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%.

Submission

predict outcome levels on the original Testing data set using Random Forest algorithm

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

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  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
```