Practicle Machine Learning Project - Prediction

A*** F***

14/07/2020

In this assignment, we will be predicting the manner of exercise using data from accelerometers on the belt, forearm, and dumbell of 6 participants. The main variable we are concerned with is the "classe" variable and we will use other variables to predict, build our model and use the final prediction model to predict different test cases.

##Load Libaries##

```
## Loading required package: lattice

## Loading required package: ggplot2
```

##Load Dataset##

Next we will load the training data and test data.

```
training_data <-
read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-
training.csv",header=TRUE, na.strings=c("NA","#DIV/0!",""))

test_data <-
read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-
testing.csv", header=TRUE,na.strings=c("NA","#DIV/0!",""))

# dimension of train and test data
dim(training_data)

## [1] 19622   160

dim(test_data)

## [1]  20 160
```

As we can see there are 19622 observations for 160 variables in training data and 20 observations for 160 variables in test data. We will likely have to reduce the variables in training data to build the model more efficiently.

#Clean data#

```
#delete columns with missing values
column_index <- colSums(is.na(training_data))/nrow(training_data)<0.95
training_data <- training_data[,column_index]

test_index <- colSums(is.na(test_data))/nrow(test_data)<0.95
test_data <- test_data[,test_index]
```

```
# delete irrelevant variables
training_data <- training_data[,-c(1:7)]
test_data <- test_data[,-c(1:7)]
```

Build Model

After loading and checking the basic characteristics of the data, we will build a model by first getting a training and test set from within the train data.

For reproducibility reasons, we will set a seed as well.

```
# set seed for reproducibility
set.seed(1234)

# partitioning for cross-validation
inTrain <- createDataPartition(y=training_data$classe, p=0.7, list=FALSE)

training <- training_data[inTrain,]
testing <- training_data[-inTrain,]

dim(training)

## [1] 13737    53

dim(testing)

## [1] 5885   53
```

To perform cross validation, we partitioned the train data into 2 subsets: 70% training and 30% testing.

Classe variable has 5 levels with Level A having the most frequencies and Level D with the least frequencies.

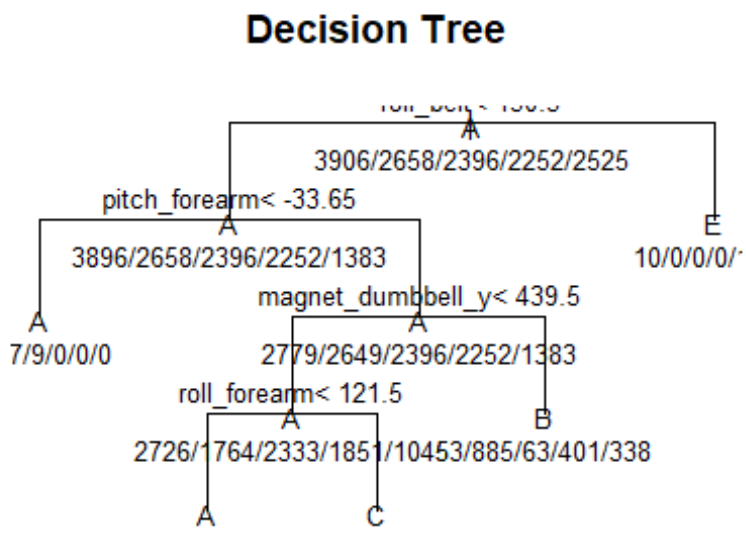
The levels of classe represent: 1. Exactly according to the specification (Class A) 2. Throwing the Elbows to the front (Class B) 3. Lifting the dumbbell only half way (Class C) 4. Lowering the dumbbell only halfway (Class D) 5. Throwing the hips to the front (Class E)

We will build 2 models: Decision Tree and Random Forest Algorithm. Both these algorithms are known for their ability to detect features important for classification.

Decision Tree

```
# create model for decision tree
model_tree <- train(classe~., method="rpart",data=training)

# plot decision tree
plot(model_tree$finalModel, uniform=TRUE, main = "Decision Tree")
text(model_tree$finalModel, use.n = TRUE, all=TRUE, cex=.8)
```



We will then predict using the model built and display test result.

```
# predict using prediction model
prediction_tree <- predict(model_tree, newdata=testing)

confusionMatrix(prediction_tree, as.factor(testing$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1519  473  484  451  156
##          B   28  401   45  167  148
##          C  123  265  497  346  289
##          D    0    0    0    0    0
##          E    4    0    0    0  489
##
## Overall Statistics
##
##                Accuracy : 0.4938
##                  95% CI : (0.4809, 0.5067)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.338
##
##  Mcnemar's Test P-Value : NA
```

```
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9074  0.35206  0.48441   0.0000  0.45194
## Specificity            0.6286  0.91825  0.78946   1.0000  0.99917
## Pos Pred Value         0.4927  0.50824  0.32697      NaN  0.99189
## Neg Pred Value         0.9447  0.85518  0.87881   0.8362  0.89002
## Prevalence             0.2845  0.19354  0.17434   0.1638  0.18386
## Detection Rate         0.2581  0.06814  0.08445   0.0000  0.08309
## Detection Prevalence   0.5239  0.13407  0.25828   0.0000  0.08377
## Balanced Accuracy      0.7680  0.63516  0.63693   0.5000  0.72555
```

Random Forest

```
# create model for random forest
model_forest <- train(classe~., method="rf",data=training,ntree=128)
```

Once we created the prediction model with random forest algorithm, we will use it on test data and display the results.

```
# predict using prediction model
prediction_forest <- predict(model_forest, newdata=testing)

confusionMatrix(prediction_forest, as.factor(testing$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    4    0    0    0
##          B    0 1130   12    0    0
##          C    1    5 1012    8    1
##          D    0    0    2  955    0
##          E    0    0    0    1 1081
##
## Overall Statistics
##
##                Accuracy : 0.9942
##                  95% CI : (0.9919, 0.996)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9927
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   0.9921   0.9864   0.9907   0.9991
```

```
## Specificity            0.9991   0.9975   0.9969   0.9996   0.9998
## Pos Pred Value         0.9976   0.9895   0.9854   0.9979   0.9991
## Neg Pred Value         0.9998   0.9981   0.9971   0.9982   0.9998
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2843   0.1920   0.1720   0.1623   0.1837
## Detection Prevalence   0.2850   0.1941   0.1745   0.1626   0.1839
## Balanced Accuracy      0.9992   0.9948   0.9916   0.9951   0.9994
```

Conclusion

From the output of the confusionMatrix command, we can see that Random Forest Algorithm had a higher performance rate than Decision Tree Algorithm.

Hence, we will choose random forest algorithm as the accuracy is 0.995 with the expected out-of-sample error estimated to be 0.005.

Using Model on Test Data

```
predict(model_forest, test_data)
```

```
##  [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
```