Practical Machine Learning - Project

Samit Patnaik

13/08/2019

Read training and testing data from the url(s) provided in the assignment. Assign the url(s) to variable to be able to read the data into csv file

```
train_url <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
test_url <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
download.file(url = train_url ,destfile = "training.csv")
download.file(url = test_url ,destfile = "testing.csv")
```

Read training and testing data, identifying “”(empty fields), “NA” and “#DIV/0!” as “NA” everywhere

```
train <- read.csv("training.csv",na.strings = c("NA","#DIV/0!",""))
test <- read.csv("testing.csv",na.strings = c("NA","#DIV/0!",""))
```

Load the librariees

```
library(caret)
```

```
## Loading required package: lattice

## Loading required package: ggplot2
```

```
library(ggplot2)
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)
```

Set aside 40 percent of the training data for cross validation purposes

```
summary(train$classe)
```

```
##    A    B    C    D    E
## 5580 3797 3422 3216 3607
```

```
inTrain <- createDataPartition(y=train$classe,p=0.6,list = FALSE)
myTrain <- train[inTrain,]
myTest <- train[-inTrain,]
```

Cleaning Train and Test data . Delete columns with all missing values

```
myTrain <- myTrain[,colSums(is.na(myTrain))==0]
myTest <- myTest[,colSums(is.na(myTest))==0]
head(colnames(myTrain), 10)
```

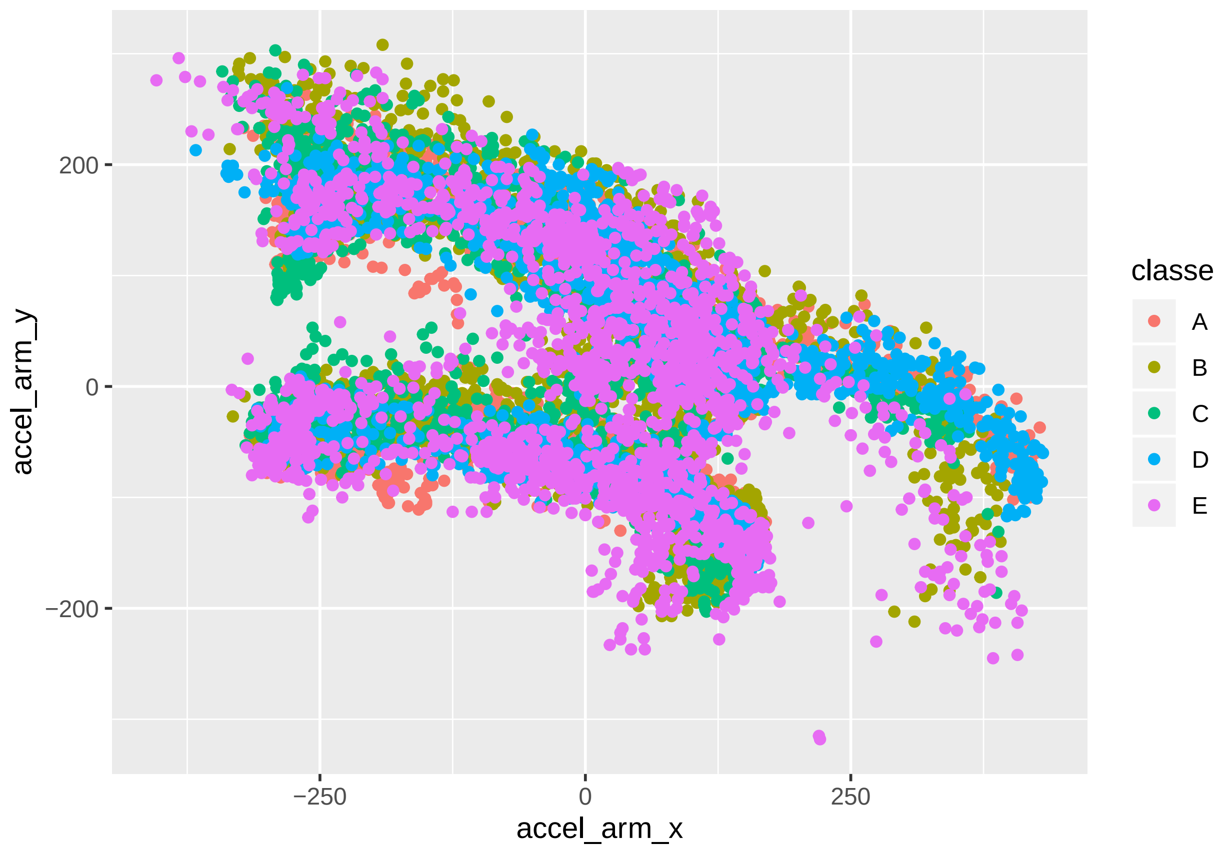
```
##  [1] "X"                    "user_name"            "raw_timestamp_part_1"
##  [4] "raw_timestamp_part_2" "cvtd_timestamp"       "new_window"
##  [7] "num_window"           "roll_belt"            "pitch_belt"
## [10] "yaw_belt"
```

/ We can delete first 7 variables, because they are irrelevant to our project: "user_name", "raw_timestamp_part_1", "raw_timestamp_part_2", "cvtd_timestamp", "new_window" and "num_window" (columns 1 to 7). /

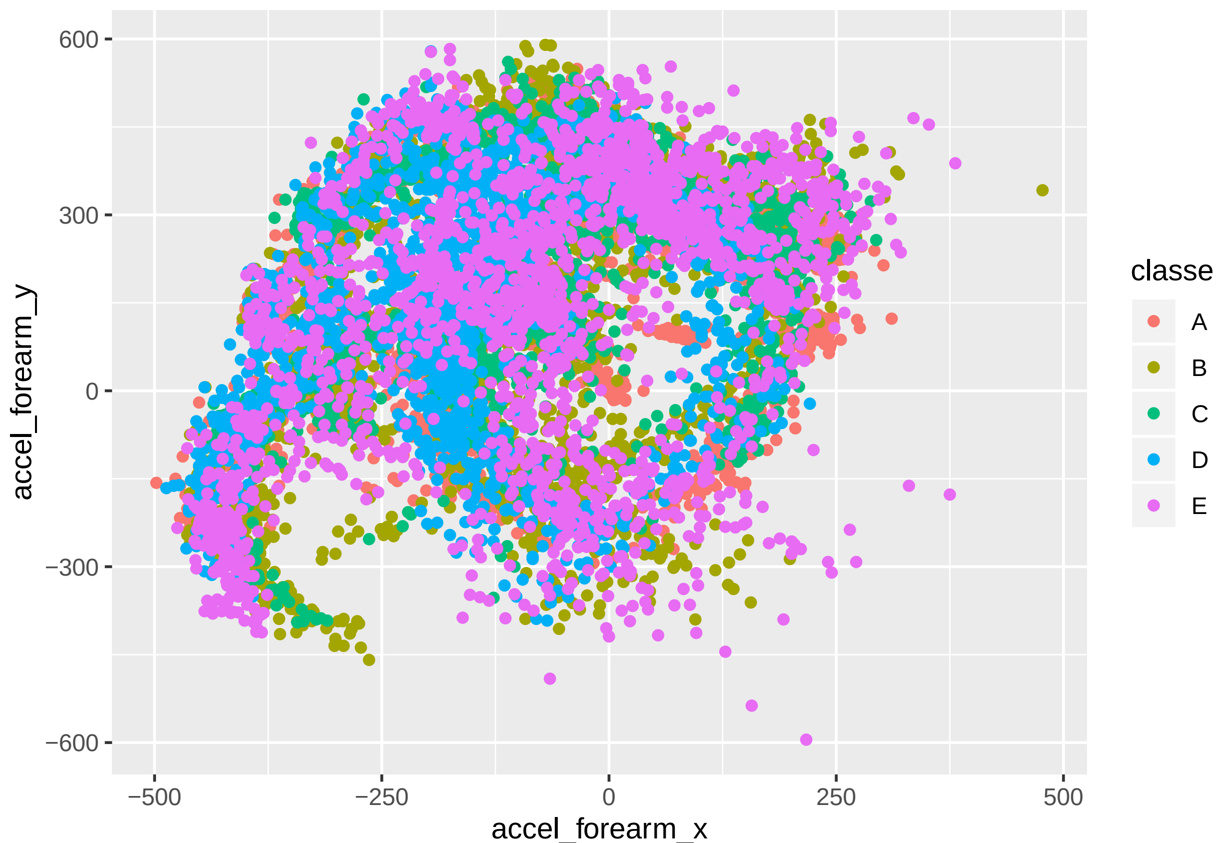
```
myTrain <- myTrain[,8:dim(myTrain)[2]]
myTest <- myTest[,8:dim(myTest)[2]]
```

Plot some data before analysis

```
qplot(accel_arm_x,accel_arm_y,col=classe,data=myTrain)
```




```r
qplot(accel_forearm_x, accel_forearm_y, col=classe, data=myTrain)
```



/Plotting some accelaration data from train data, we can see that the pattern is very similar and hard to distinguish among classes A,B,C,D,E / ## Predicting Models

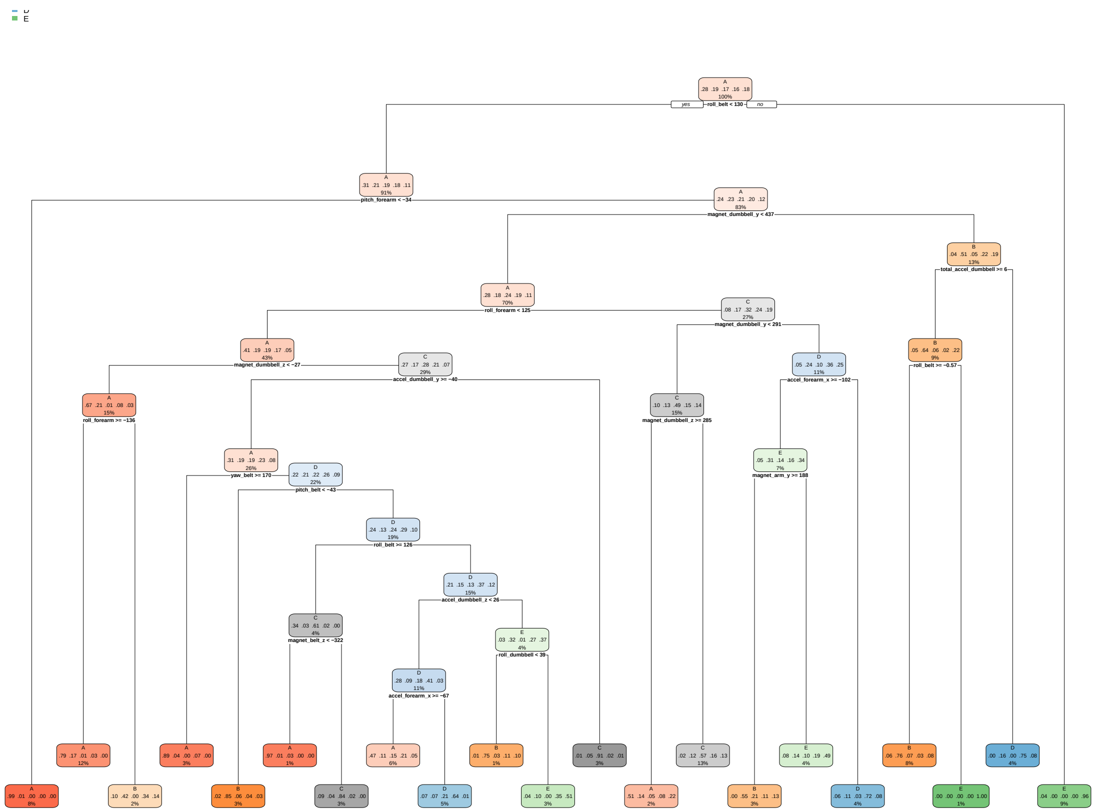
Apply Classification Tree Model

```
modelCTree <- rpart(classe ~ ., data = myTrain , method = "class")
predictionCTree <- predict(modelCTree,myTest, type= "class")
confusionMatrix(predictionCTree,myTest$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2000  269   94  163   65
##          B   75  891  114  116  120
##          C   43  168 1020  175  119
##          D   54  119  102  702   56
##          E   60   71   38  130 1082
##
## Overall Statistics
##
##                Accuracy : 0.7258
##                  95% CI : (0.7158, 0.7357)
##     No Information Rate : 0.2845
```

```
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6511
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8961   0.5870   0.7456  0.54588   0.7503
## Specificity            0.8947   0.9328   0.9220  0.94954   0.9533
## Pos Pred Value         0.7719   0.6771   0.6689  0.67957   0.7835
## Neg Pred Value         0.9559   0.9040   0.9449  0.91428   0.9443
## Prevalence             0.2845   0.1935   0.1744  0.16391   0.1838
## Detection Rate         0.2549   0.1136   0.1300  0.08947   0.1379
## Detection Prevalence   0.3302   0.1677   0.1944  0.13166   0.1760
## Balanced Accuracy      0.8954   0.7599   0.8338  0.74771   0.8518
```

```r
rpart.plot(modelCTree)
```



Apply Random Forest Models

```r
modelRF <- randomForest(classe ~ ., data=myTrain ,method="class")
predictionRF <- predict(modelRF , myTest , type = "class")
confusionMatrix(predictionRF,myTest$classe)
```

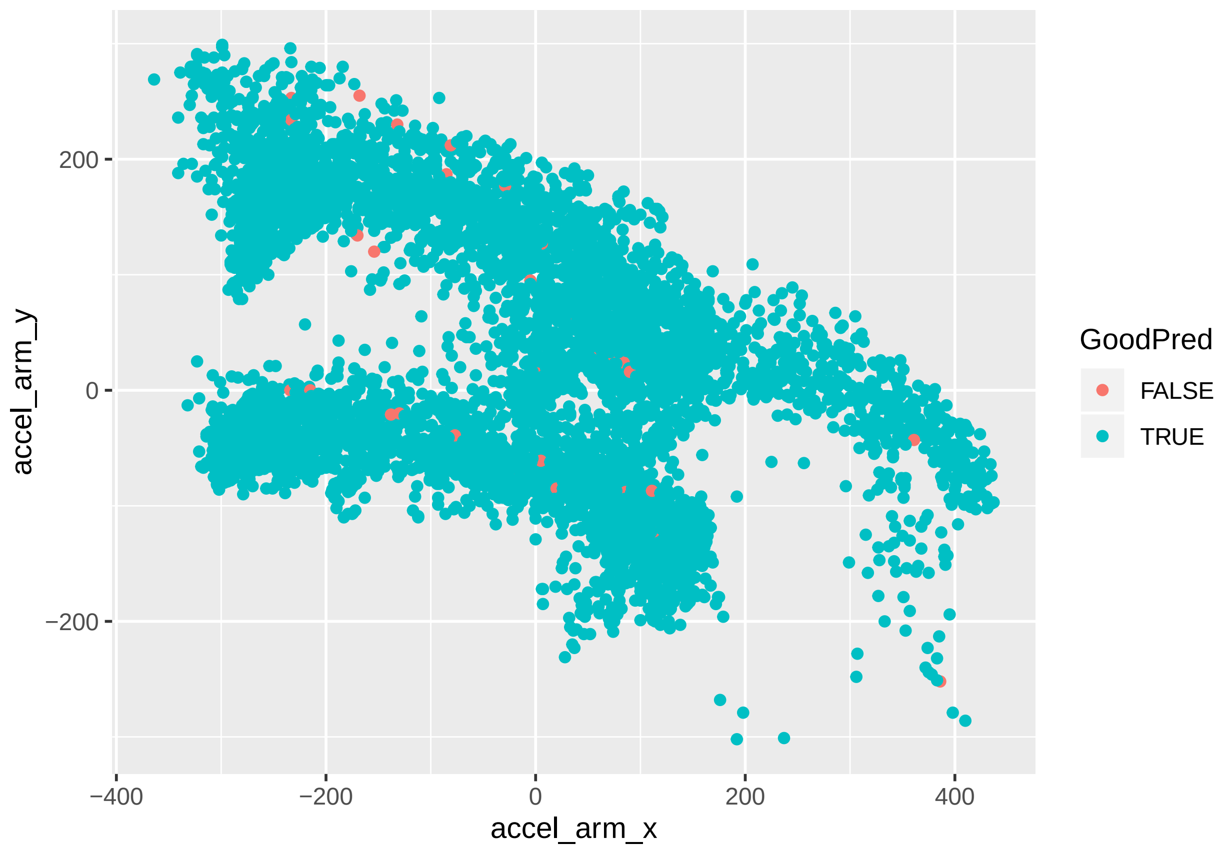
```
## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2227   15    0    0    0
##          B    5 1498   14    0    0
##          C    0    5 1354   28    0
##          D    0    0    0 1258    6
##          E    0    0    0    0 1436
##
## Overall Statistics
##
##                Accuracy : 0.9907
##                  95% CI : (0.9883, 0.9927)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9882
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9978   0.9868   0.9898   0.9782   0.9958
## Specificity            0.9973   0.9970   0.9949   0.9991   1.0000
## Pos Pred Value         0.9933   0.9875   0.9762   0.9953   1.0000
## Neg Pred Value         0.9991   0.9968   0.9978   0.9957   0.9991
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2838   0.1909   0.1726   0.1603   0.1830
## Detection Prevalence   0.2858   0.1933   0.1768   0.1611   0.1830
## Balanced Accuracy      0.9975   0.9919   0.9923   0.9887   0.9979
```

Adding a new column GoodPred to myTest_predindicator to determine predicted value for the test data vs true value in the test data

```r
myTest_predindicator <- myTest
myTest_predindicator$GoodPred <- myTest$classe == predictionRF

qplot(accel_arm_x,accel_arm_y,col=GoodPred,data=myTest_predindicator)
```



Final Prediction

```
FinaPred <- predict(modelRF,test)
table(t(data.frame(FinaPred)))
```

```
##
## A B C D E
## 7 8 1 1 3
```

```
FinaPred
```

```
##  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
```

#Conclusion : Prediction evaluations were based on maximizing the accuracy and minimizing the out-of-sample error. All other available variables after cleaning were used for prediction. Two models were tested using decision tree and random forest algorithms. The model with the highest accuracy were chosen as final model.

Random Forest Prediction is much more accurate then classification Tree model