Machine_Learning_PA

Jeet Tanna

July 26, 2020

R Markdown

The goal of the assignment is to predict the manner in which they did the exercise. The goal is also to quantify how much of a particular activity they do. Thus we will first load the caret package and import the training and test data. I'll store the testing data in the valid variable and have the train data partitioned into training and testing with a 80-20 ratio.

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.5.3
```

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

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

```
## Warning: package 'ggplot2' was built under R version 3.5.3
```

```
train<-read.csv("C:/Users/JeetsPC-1/Desktop/Study Material/R DataSets/pml-training.csv")
valid<-read.csv("C:/Users/JeetsPC-1/Desktop/Study Material/R DataSets/pml-testing.csv")
training_samp <- createDataPartition(y=train$classe, p=0.7, list=FALSE)
training <- train[training_samp, ]
testing <- train[-training_samp, ]
```

Removing columns with NAs

First we'll have to identify the colums which do not have NA values and they are the ones that we will be using in our model. So we'll create a function that will do that for us.

```
all_zero_colnames <- sapply(names(valid), function(x) all(is.na(valid[,x])==TRUE))
nznames <- names(all_zero_colnames)[all_zero_colnames==FALSE]
nznames <- nznames[-(1:7)]
nznames <- nznames[1:(length(nznames)-1)]
nznames
```

```
##  [1] "roll_belt"            "pitch_belt"           "yaw_belt"            
##  [4] "total_accel_belt"     "gyros_belt_x"         "gyros_belt_y"        
##  [7] "gyros_belt_z"         "accel_belt_x"         "accel_belt_y"        
## [10] "accel_belt_z"         "magnet_belt_x"        "magnet_belt_y"       
## [13] "magnet_belt_z"        "roll_arm"             "pitch_arm"           
## [16] "yaw_arm"              "total_accel_arm"      "gyros_arm_x"         
## [19] "gyros_arm_y"          "gyros_arm_z"          "accel_arm_x"         
## [22] "accel_arm_y"          "accel_arm_z"          "magnet_arm_x"        
## [25] "magnet_arm_y"         "magnet_arm_z"         "roll_dumbbell"       
## [28] "pitch_dumbbell"       "yaw_dumbbell"         "total_accel_dumbbell"
## [31] "gyros_dumbbell_x"     "gyros_dumbbell_y"     "gyros_dumbbell_z"    
## [34] "accel_dumbbell_x"     "accel_dumbbell_y"     "accel_dumbbell_z"    
## [37] "magnet_dumbbell_x"    "magnet_dumbbell_y"    "magnet_dumbbell_z"   
## [40] "roll_forearm"         "pitch_forearm"        "yaw_forearm"         
## [43] "total_accel_forearm"  "gyros_forearm_x"      "gyros_forearm_y"     
## [46] "gyros_forearm_z"      "accel_forearm_x"      "accel_forearm_y"     
## [49] "accel_forearm_z"      "magnet_forearm_x"     "magnet_forearm_y"    
## [52] "magnet_forearm_z"
```

These are the variables that we will be using in our model. **Now we'll build the model using decision trees and random forest with the train function.**

```
model<- train(classe ~ ., data=training[, c('classe', nznames)],method='rpart')
model2<- train(classe ~ ., data=training[, c('classe', nznames)],method='rf',
  ntree=50
)
```

Cross Validation of Models

Now that we have fitted two models, we can now cross validate them to find out which is more effective.

```
acc_dt <- predict(model, newdata=testing)
cm_dt <- confusionMatrix(acc_dt, testing$classe)
acc_rf <- predict(model2, newdata=testing)
cm_rf <- confusionMatrix(acc_rf, testing$classe)
Accuracy <- data.frame(Model = c('DT','RF'),Accuracy = rbind(cm_dt$overall[1],cm_rf$overall[1]))
print(Accuracy)
```

```
##   Model  Accuracy
## 1    DT 0.4853016
## 2    RF 0.9920136
```

Since the Random Forest has proved to be the most accurate model with over 0.99 accuracy we will look at its matrix

```
cm_rf
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1673   12    0    0    0
##          B    0 1123    4    1    3
##          C    0    4 1020   13    3
##          D    0    0    2  950    4
##          E    1    0    0    0 1072
## 
## Overall Statistics
##                                           
##                Accuracy : 0.992           
##                  95% CI : (0.9894, 0.9941)
##     No Information Rate : 0.2845          
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.9899          
##                                           
##  Mcnemar's Test P-Value : NA              
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   0.9860   0.9942   0.9855   0.9908
## Specificity            0.9972   0.9983   0.9959   0.9988   0.9998
## Pos Pred Value         0.9929   0.9929   0.9808   0.9937   0.9991
## Neg Pred Value         0.9998   0.9966   0.9988   0.9972   0.9979
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2843   0.1908   0.1733   0.1614   0.1822
## Detection Prevalence   0.2863   0.1922   0.1767   0.1624   0.1823
## Balanced Accuracy      0.9983   0.9921   0.9950   0.9921   0.9953
```

Prediction

We can now predict using the Random Forest Model on our validation dataset.

```
validation <- predict(model2, newdata=valid)
validation_result <- data.frame(problem_id=valid$problem_id,predicted=validation)
print(validation_result)
```

```
##    problem_id predicted
## 1           1         B
## 2           2         A
## 3           3         B
## 4           4         A
## 5           5         A
## 6           6         E
## 7           7         D
## 8           8         B
## 9           9         A
## 10         10         A
## 11         11         B
## 12         12         C
## 13         13         B
## 14         14         A
## 15         15         E
## 16         16         E
## 17         17         A
## 18         18         B
## 19         19         B
## 20         20         B
```

Executive Summary

A reasonably accurate model was fit using random forests which had an accuracy of over 0.99. The data was hard to work with because of the NA values and identifying them was the real task. A more efficient model still can be produced using an ensemble model which would take traits of both the decision tree model and the random forest model.