Prediction Assignment Writeup

O.S.

16 November 2018

Executive Summary

Using devices such as Jawbone Up, Nike FuelBand, and Fitbit it is now possible to collect a large amount of data about personal activity relatively inexpensively. The purpose of this exercise is to use the data to try and predict five type of activities (the way they did the training)

Libraries

```
library(ggplot2)
library(caret)
```

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

```
library(gbm)
```

```
## Loaded gbm 2.1.4
```

```
library(parallel)
library(doParallel)
```

```
## Loading required package: foreach
```

```
## Loading required package: iterators
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

Load Data

Data observiation

```
dim(training)
```

```
## [1] 19622   160
```

The training data set contains 19622 observations and 160 variables.

```
dim(validiation)
```

```
## [1]  20 160
```

The validation set is 20 observations over 160 variables.

Data cleaning

Calculate the NA percentage.

```
NAnoTrain<-sum(is.na(training))
NAtrainPercentage<-NAnoTrain/(ncol(training)*nrow(training))
NAnoValidiation<-sum(is.na(validiation))
NAValidiationPercentage<-NAnoValidiation/(ncol(validiation)*nrow(validiation))
print(NAtrainPercentage)
```

```
## [1] 0.4100856
```

```
print(NAValidiationPercentage)
```

```
## [1] 0.625
```

It can be sen there is a large number of NA values. In the Training set 0.41 of the data is NA. in the Validation set 0.625 of the information is NA.

The approach is to check if a column contains more the 80% NA either in training or in the validation set and remove it from both sets (if true).

```
#Calculate the percentage of the NA in each column
#and remove the columns which
#have more than 80% NA - Training
TrainColNAPrec<-training %>%
   summarise_all(funs(100*mean(is.na(.))))
TrainColToRemove<-names(TrainColNAPrec[,TrainColNAPrec>99])


#Calculate the percentage of the NA in each column
#and remove the columns which
#have more than 80% NA - Validiation
ValidiationColNAPrec<-validiation %>%
   summarise_all(funs(100*mean(is.na(.))))
ValidiationToRemove<-names(ValidiationColNAPrec[,ValidiationColNAPrec>99])

#Bind the column's names (If the NA percentage is above 80% in
#one of the set remove the columns from both sets
ColToRemove<-unique(c(TrainColToRemove,ValidiationToRemove))
ColToKeep<-!(names(training) %in% ColToRemove)
Training<-training[ ,ColToKeep]
Validiation<-validiation[ ,ColToKeep]
```

Remove the names data and the timestamps related columns As they will not contribute to the prediction.

```
trainRemove <- grepl("^X|timestamp|window", names(Training))
Training <- Training[, !trainRemove]

ValidiationRemove <- grepl("^X|timestamp|window", names(Validiation))
Validiation <- Validiation[, !ValidiationRemove]
print(dim(Training))
```

```
## [1] 19622    54
```

```
print(dim(Validiation))
```

```
## [1] 20 54
```

There are 54 columns left for the Training and Validiation

Modeling

The model that will be tested are :

GBM - (Gradient Boosting Machine) (Boosting with Trees) RF - Random Forests

Split the Training set into train and test sets

```
set.seed(123444) # For reproducibile purpose
inTrain <- createDataPartition(Training$classe, p=0.70, list=F)
trainData <- Training[inTrain, ]
testData <- Training[-inTrain, ]
```

GBM Model

```
#Use parallel processing to train the model.
cluster <- makeCluster(detectCores() - 1) # convention to leave 1 core for OS
registerDoParallel(cluster)
fitControl <- trainControl(method = "cv",
                           number = 5,
                           allowParallel = TRUE)
```

Train the model

```
set.seed(13444)
gbmfit <- train(as.factor(classe)~., method="gbm",data=trainData,trControl = fitControl)
```

```
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.2329
##      2        1.4595             nan     0.1000    0.1598
##      3        1.3580             nan     0.1000    0.1191
##      4        1.2834             nan     0.1000    0.1122
##      5        1.2135             nan     0.1000    0.0841
##      6        1.1602             nan     0.1000    0.0790
##      7        1.1100             nan     0.1000    0.0668
##      8        1.0671             nan     0.1000    0.0542
##      9        1.0309             nan     0.1000    0.0658
##     10        0.9904             nan     0.1000    0.0494
##     20        0.7620             nan     0.1000    0.0249
##     40        0.5375             nan     0.1000    0.0102
##     60        0.4103             nan     0.1000    0.0065
##     80        0.3270             nan     0.1000    0.0048
##    100        0.2712             nan     0.1000    0.0056
##    120        0.2259             nan     0.1000    0.0017
##    140        0.1918             nan     0.1000    0.0016
##    150        0.1781             nan     0.1000    0.0014
```

Stop Parllel processing

```
stopCluster(cluster)
registerDoSEQ()
```

Predict using the test data in order to calculate the accuercy.

```
gbmpred <- predict(gbmfit,testData)
gbmaccuracy <- confusionMatrix(gbmpred,testData$classe)$overall['Accuracy']
```

RF Model

```
#Use parallel processing to train the model.
cluster <- makeCluster(detectCores() - 1) # convention to leave 1 core for OS
registerDoParallel(cluster)
fitControl <- trainControl(method = "cv",
                           number = 5,
                           allowParallel = TRUE)
```

Train the model

```
set.seed(13444)
RFfit <- train(as.factor(classe)~., method="rf",data=trainData,trControl = fitControl)
```

Stop Parllel processing

```
stopCluster(cluster)
registerDoSEQ()
```

Predict using the test data in order to calculate the accuercy.

```
RFpred <- predict(RFfit,testData)
RFaccuracy <- confusionMatrix(RFpred,testData$classe)$overall['Accuracy']
```

stacking (Using the two models)

```
cluster <- makeCluster(detectCores() - 1) # convention to leave 1 core for OS
registerDoParallel(cluster)
fitControl <- trainControl(method = "cv",
                           number = 5,
                           allowParallel = TRUE)
pred1 <- predict(gbmfit,testData); pred2 <- predict(RFfit,testData)
predDF <- data.frame(pred1,pred2,classe=testData$classe)
combModFit <- train(classe ~.,method="rf",data=predDF)
stopCluster(cluster)
registerDoSEQ()
combPred <- predict(combModFit,predDF)
StackAccuercy<-confusionMatrix(combPred,testData$classe)$overall['Accuracy']
```

Results

```
cat("GBM Accuercy:",gbmaccuracy)
```

```
## GBM Accuercy: 0.9617672
```

```
cat("RF Accuercy:",RFaccuracy)
```

```
## RF Accuercy: 0.9935429
```

```
cat("Stack Accuercy:",StackAccuercy)
```

```
## Stack Accuercy: 0.9935429
```