Practial Maching Learning Assignment

Project Description

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. These type of devices are part of the quantified self movement – a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. One thing that people regularly do is quantify how much of a particular activity they do, but they rarely quantify how well they do it. In this project, your goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways.

Overview

This report is the final assignment of Practical Machine Learning in Coursera. In this project, the goal is to analyze data from "Weight Lifting Exercises Dataset" and predict the manner in which participants did the exercise. This prediction model will be used to predict 20 different test cases in the final testing dataset.

Data Clean

1. Preparation

Loading necessary R packages

```r
library(caret)
library(randomForest)
library(corrplot)
library(rattle)
```

```
## Warning: Failed to load RGtk2 dynamic library, attempting to install it.
```

```r
library(rpart)
```

2. Data Cleaning

- Data set is downloaded from the following URL

```r
URL_TrainingFile <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
URL_TestingFile  <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
```

- Loading data from csv file

```r
trainRaw <- read.csv("./data/pml-training.csv", na.strings=c("", "NA", "NULL"))
dim(trainRaw)
```

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

- Remove NA variables

```r
tr.notnan<- trainRaw[ , colSums(is.na(trainRaw)) == 0]
remove = c('X', 'user_name', 'raw_timestamp_part_1', 'raw_timestamp_part_2', 'cvtd_timestamp', 'new_wind
tr.core <- tr.notnan[, -which(names(tr.notnan) %in% remove)]
dim(tr.core)
```

```
## [1] 19622    53
```

- Remove variables with near zero variance variables

```r
zeroVar= nearZeroVar(tr.core[sapply(tr.core, is.numeric)], saveMetrics = TRUE)
tr.nonzero = tr.core[,zeroVar[, 'nzv']==0]
dim(tr.nonzero)
```

```
## [1] 19622    53
```

- Remove highly correlated variables (90%).

```r
corrMatrix <- cor(na.omit(tr.nonzero[sapply(tr.nonzero, is.numeric)]))
dim(corrMatrix)
```

```
## [1] 52 52
```

```r
removecor = findCorrelation(corrMatrix, cutoff = .90, verbose = TRUE)
```

```
## Compare row 10  and column  1 with corr  0.992
##   Means:  0.27 vs 0.168 so flagging column 10
## Compare row 1  and column  9 with corr  0.925
##   Means:  0.25 vs 0.164 so flagging column 1
## Compare row 9  and column  4 with corr  0.928
##   Means:  0.233 vs 0.161 so flagging column 9
## Compare row 8  and column  2 with corr  0.966
##   Means:  0.245 vs 0.157 so flagging column 8
## Compare row 19  and column  18 with corr  0.918
##   Means:  0.091 vs 0.158 so flagging column 18
## Compare row 46  and column  31 with corr  0.914
##   Means:  0.101 vs 0.161 so flagging column 31
## Compare row 46  and column  33 with corr  0.933
##   Means:  0.083 vs 0.164 so flagging column 33
## All correlations <= 0.9
```

```r
tr.decor = tr.nonzero[,-removecor]
dim(tr.decor)
```

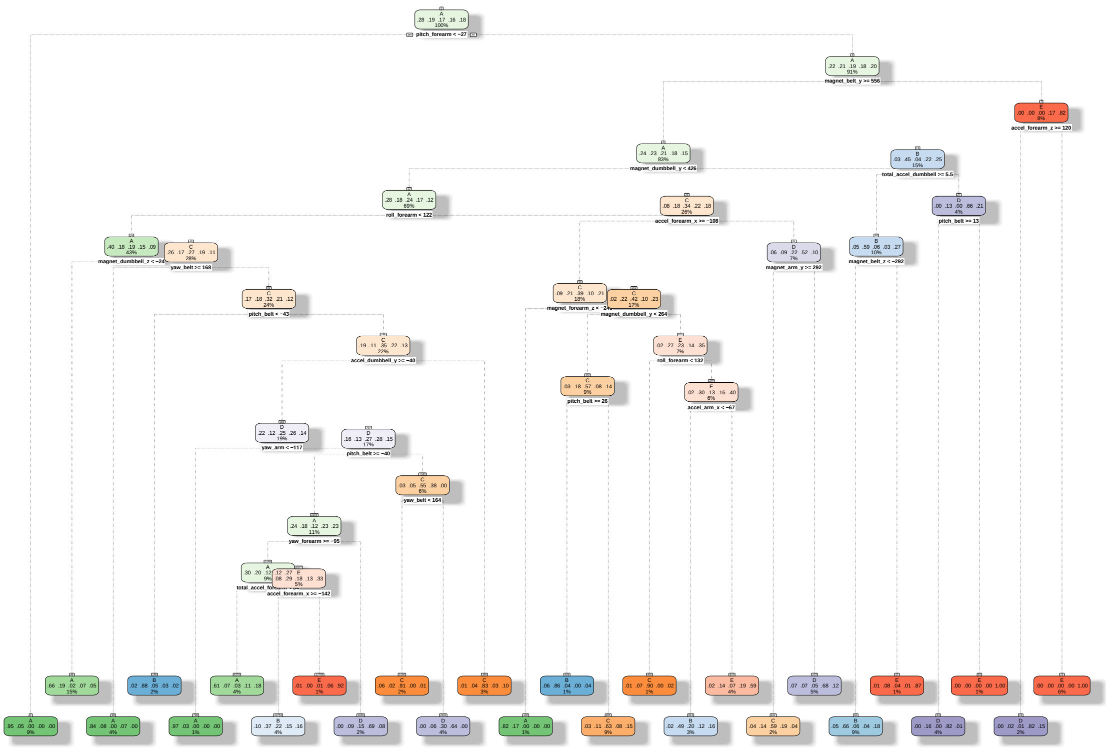
```
## [1] 19622    46
```

- Split data into training and testing for cross validation

```r
inTrain <- createDataPartition(tr.decor$classe, p=0.7, list=F)
training <- tr.decor[inTrain,]
testing <- tr.decor[-inTrain,]
```

Model 1 - Decision Tree

```r
set.seed(12345)
rf.decTree_training <- rpart(classe ~ ., data=training, method="class")
fancyRpartPlot(rf.decTree_training, sub="Descision Tree")
```



Descision Tree

Out of sampel accuracy for decision tree model

```r
predicted_desTree <- predict(rf.decTree_training, newdata=testing, type="class")
confMat_desTree <- confusionMatrix(predicted_desTree, testing$classe)
confMat_desTree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1542  259   16  102   90
##          B   84  668  140   70  169
##          C   25   86  738   68  120
##          D   16   83  110  671   62
##          E    7   43   22   53  641
```

```
##
## Overall Statistics
##
##                Accuracy : 0.7239
##                  95% CI : (0.7123, 0.7353)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6482
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9211   0.5865   0.7193   0.6961   0.5924
## Specificity            0.8891   0.9024   0.9385   0.9449   0.9740
## Pos Pred Value         0.7675   0.5906   0.7117   0.7123   0.8368
## Neg Pred Value         0.9659   0.9009   0.9406   0.9407   0.9139
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2620   0.1135   0.1254   0.1140   0.1089
## Detection Prevalence   0.3414   0.1922   0.1762   0.1601   0.1302
## Balanced Accuracy      0.9051   0.7445   0.8289   0.8205   0.7832
```

```
confMat_desTree$overall["Accuracy"]
```

```
##  Accuracy
## 0.7238743
```

Model 2 - Random Forest

```
rf.randomForest_training=randomForest(classe~.,data=training,ntree=100, importance=TRUE)
rf.randomForest_training
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = training, ntree = 100,      importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 100
## No. of variables tried at each split: 6
##
##         OOB estimate of  error rate: 0.77%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3903    2    0    0    1 0.0007680492
## B   14 2628   15    0    1 0.0112866817
## C    1   22 2366    5    2 0.0125208681
## D    0    1   30 2219    2 0.0146536412
## E    0    0    2    8 2515 0.0039603960
```

Out of sampel accuracy for random forest model

```r
predicted_rf <- predict(rf.randomForest_training, newdata=testing, type="class")
confMat_rf <- confusionMatrix(predicted_rf, testing$classe)
confMat_rf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1672   10    0    0    0
##          B    2 1126    9    0    0
##          C    0    3 1017   10    4
##          D    0    0    0  954    4
##          E    0    0    0    0 1074
##
## Overall Statistics
##
##                Accuracy : 0.9929
##                  95% CI : (0.9904, 0.9949)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.991
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9988   0.9886   0.9912   0.9896   0.9926
## Specificity            0.9976   0.9977   0.9965   0.9992   1.0000
## Pos Pred Value         0.9941   0.9903   0.9836   0.9958   1.0000
## Neg Pred Value         0.9995   0.9973   0.9981   0.9980   0.9983
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2841   0.1913   0.1728   0.1621   0.1825
## Detection Prevalence   0.2858   0.1932   0.1757   0.1628   0.1825
## Balanced Accuracy      0.9982   0.9931   0.9939   0.9944   0.9963
```

```r
confMat_rf$overall["Accuracy"]
```

```
##  Accuracy
## 0.9928632
```

Predicting Results

The accuracy of the two modeling methods shown above are:

Decision Tree : 0.7238743
Random Forest : 0.9928632

Therefore, we are using Random Forest as the final model to predict results in testing dataset:

- Loading testing dataset

```r
testingDataset <- read.csv("./data/pml-testing.csv", na.strings=c("", "NA", "NULL"))
dim(testingDataset)
```

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

- Predicting results

```r
answers <- predict(rf.randomForest_training, testingDataset)
answers
```

```
##  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
```