Practical Machine Learning Project - Prediction Analysis

DRZ

Introduction

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, our goal is to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. Each participant was asked to perform barbell lifts correctly and incorrectly. We will use this data to predict whether the exercise was performed correctly or incorrectly.

Intended Results

The goal of your project is to predict the manner in which they did the exercise. This is the “classe” variable in the training set. You may use any of the other variables to predict with. You should create a report describing how you built your model, how you used cross validation, what you think the expected out of sample error is, and why you made the choices you did. You will also use your prediction model to predict 20 different test cases.

1. Your submission should consist of a link to a Github repo with your R markdown and compiled HTML file describing your analysis. Please constrain the text of the writeup to < 2000 words and the number of figures to be less than 5. It will make it easier for the graders if you submit a repo with a gh-pages branch so the HTML page can be viewed online (and you always want to make it easy on graders :-).
2. You should also apply your machine learning algorithm to the 20 test cases available in the test data above. Please submit your predictions in appropriate format to the programming assignment for automated grading. See the programming assignment for additional details.

System Libraries

Install packages in RStudio to support the following system libraries:

```
library(caret) #Classification and regression training
library(corrplot) #Visualization of the correlation matrix
library(randomForest) #Random forests for classification and regression
library(rpart) #Recursive partitioning and regression trees
library(rpart.plot) #Plot rpart Models
```

Preparing the Data

Import the Datasets

```
trainingDataURL <-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testingDataURL <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
trainingDataFile <- "data/pml-training.csv"
testingDataFile  <- "data/pml-testing.csv"
if (!file.exists("data")) {
  dir.create("data")
}
if (!file.exists(trainingDataFile)) {
  download.file(trainingDataURL, destfile=trainingDataFile, method="curl")
}
if (!file.exists(testingDataFile)) {
  download.file(testingDataURL, destfile=testingDataFile, method="curl")
}
```

Read the Data

Download the data from the data source then read two csv files into data frames.

```
trainingRawData <- read.csv("data/pml-training.csv")
testingRawData <- read.csv("data/pml-testing.csv")
dim(trainingRawData) #retrieve the dimensions of trainingRawData
```

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

```
dim(testingRawData) # retrieve the dimensions of testing RawData
```

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

The training data set contains 19622 observations in 160 variables. The testing data set contains 20 observations in 160 variables. A *classe* variable in the training set is the outcome for prediction.

Clean the data

Use *complete.cases* to return a logical vector that indicates which cases are complete.

```
sum(complete.cases(trainingRawData))
```

```
## [1] 406
```

Step 1. Remove all columns that contain NA values.

```
trainingRawData <- trainingRawData[, colSums(is.na(trainingRawData)) == 0]
testingRawData <- testingRawData[, colSums(is.na(testingRawData)) == 0]
```

Step 2. Remove columns that contain irrelevant information

```
classe <- trainingRawData$classe
trainRemove <- grepl("^X|timestamp|window", names(trainingRawData))
trainingRawData <- trainingRawData[, !trainRemove]
trainCleaned <- trainingRawData[, sapply(trainingRawData, is.numeric)]
trainCleaned$classe <- classe
testRemove <- grepl("^X|timestamp|window", names(testingRawData))
testingRawData <- testingRawData[, !testRemove]
testCleaned <- testingRawData[, sapply(testingRawData, is.numeric)]
```

After cleaning the training data set it now contains 19622 observations and 53 variables. The testing data set now contains 20 observations and 53 variables. The *classe* variable remains in the cleaned training set.

Slice the data

Split the clean training set into a training data set (70%) and a test data set (30%). The test data set will be used to perform cross validation.

```
set.seed(22519) # Use set.seed to ensure the frame is reproducible with the same data
inTrain <- createDataPartition(trainCleaned$classe, p=0.70, list=F) #partition the da
ta to 70% training and 30% test data
trainingData <- trainCleaned[inTrain, ]
testingData <- trainCleaned[-inTrain, ]
```

Data Modeling

The *train* function from caret will estimate the performance of the model in the training data set. Fitting a predictive model for activity recognition using **Random Forest** algorithm selects key variables and correlated covariates and outliers. Note the *cv* method and use of **5-fold cross validation** when running the function.

```
controlRandomForest <- trainControl(method="cv", 5)
testRandomForest <- train(classe ~ ., data=trainingData, method="rf", trControl=contr
olRandomForest, ntree=250)
testRandomForest
```

```
## Random Forest
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 10989, 10989, 10990, 10990, 10990
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD  Kappa SD
##    2    0.9906093  0.9881199  0.001419109  0.001795362
##   27    0.9908278  0.9883967  0.001418439  0.001794742
##   52    0.9852222  0.9813052  0.002942716  0.003723780
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

Use the *predict* function from caret to to generate predictions from the training data set. Then use the *confusionMatrix* function to compute summaries for classification models in the validation data set.

```
predictRf <- predict(testRandomForest, testingData)
confusionMatrix(testingData$classe, predictRf)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1672    1    0    0    1
##          B    6 1129    4    0    0
##          C    0    1 1021    4    0
##          D    0    0   13  950    1
##          E    0    0    1    7 1074
##
## Overall Statistics
##
##                Accuracy : 0.9934
##                  95% CI : (0.991, 0.9953)
##     No Information Rate : 0.2851
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9916
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9964   0.9982   0.9827   0.9886   0.9981
## Specificity            0.9995   0.9979   0.9990   0.9972   0.9983
## Pos Pred Value         0.9988   0.9912   0.9951   0.9855   0.9926
## Neg Pred Value         0.9986   0.9996   0.9963   0.9978   0.9996
## Prevalence             0.2851   0.1922   0.1766   0.1633   0.1828
## Detection Rate         0.2841   0.1918   0.1735   0.1614   0.1825
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9980   0.9981   0.9908   0.9929   0.9982
```

```
accuracy <- postResample(predictRf, testingData$classe)
accuracy
```

```
##  Accuracy     Kappa
## 0.9930331 0.9911872
```

```
oose <- 1 - as.numeric(confusionMatrix(testingData$classe, predictRf)$overall[1])
oose
```

```
## [1] 0.006966865
```

The estimated accuracy of the model indicates 99.30% and the estimated out-of-sample error indicates 0.69%.

Predict for Test Data Set

Apply the model to the testing data set downloaded from the data source. Note: remove the `problem_id` column.

```
result <- predict(testRandomForest, testCleaned[, -length(names(testCleaned))])
result
```

```
##  [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
```

Appendix - Figures

Figure 1. Correlation Matrix Visualization

```
corMVplot <- cor(trainingData[, -length(names(trainingData))])
corrplot(corMVplot, method="color")
```

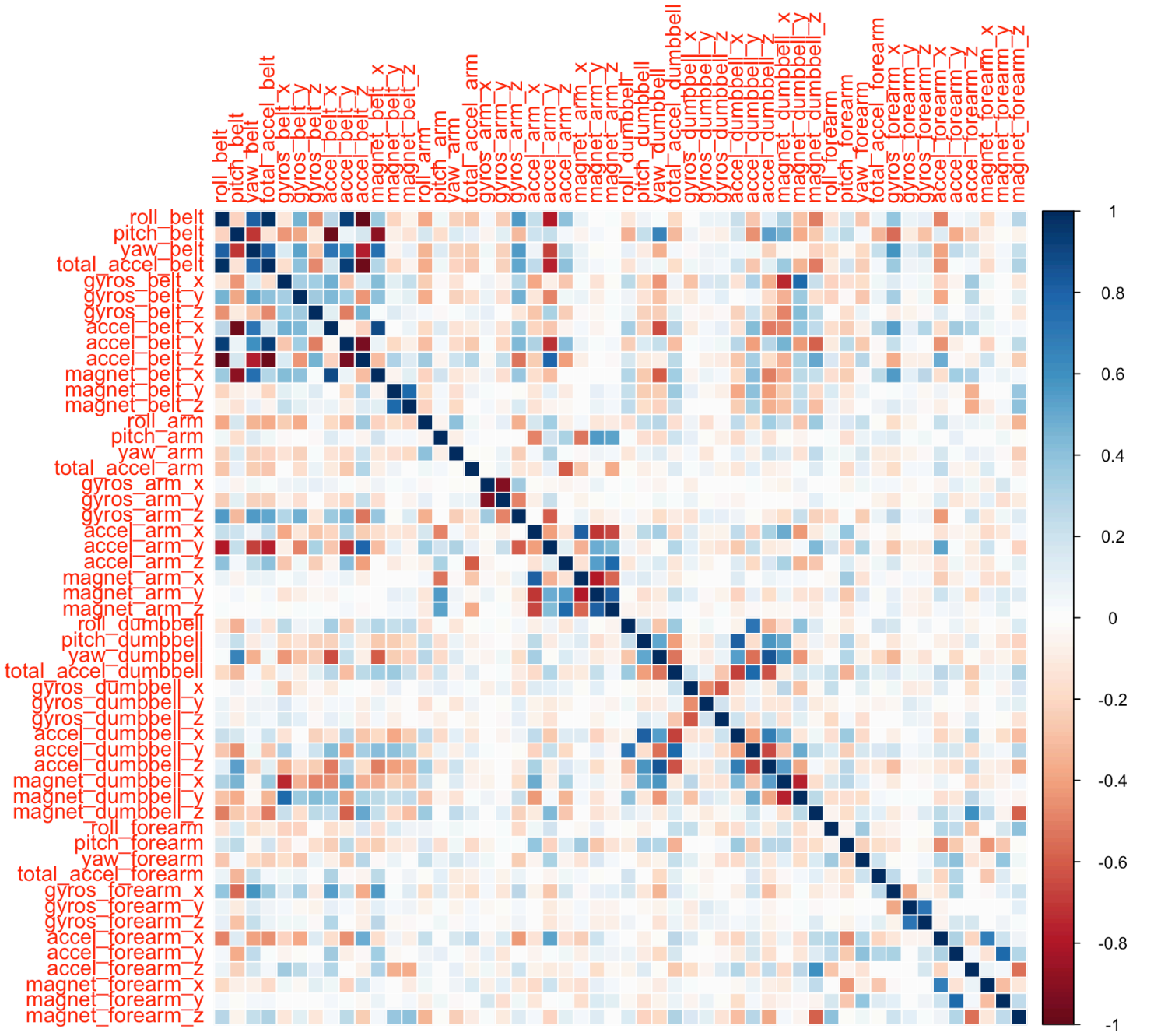
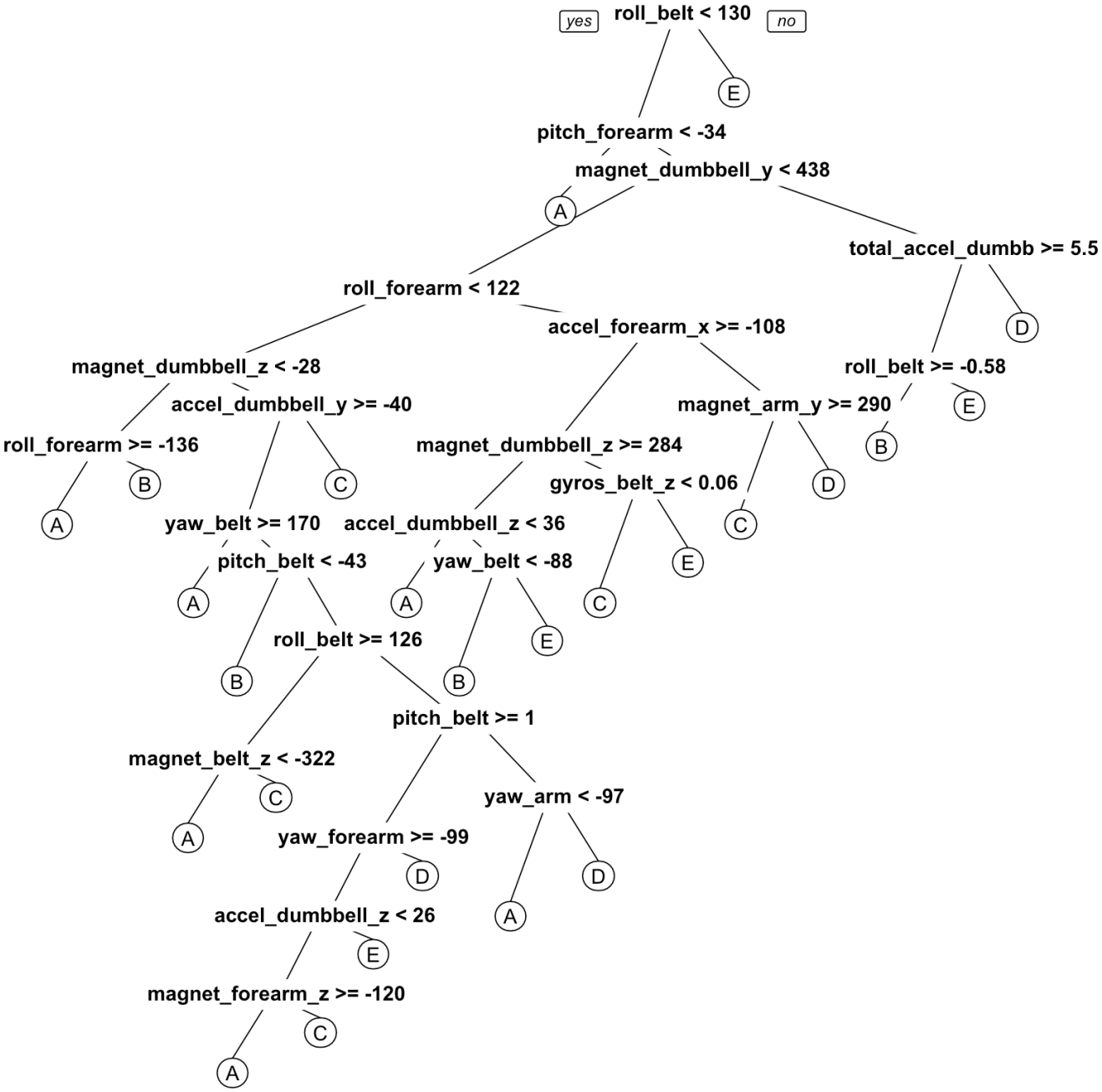


Figure 2. Decision Tree Visualization

```
DTVtreeModel <- rpart(classe ~ ., data=trainingData, method="class")
prp(DTVtreeModel) # fast plot
```


yes
roll_belt < 130
no
E
pitch_forearm < -34
magnet_dumbbell_y < 438
A
total_accel_dumbb >= 5.5
roll_forearm < 122
accel_forearm_x >= -108
D
magnet_dumbbell_z < -28
roll_belt >= -0.58
accel_dumbbell_y >= -40
magnet_arm_y >= 290
E
roll_forearm >= -136
magnet_dumbbell_z >= 284
B
B
C
gyros_belt_z < 0.06
D
A
yaw_belt >= 170
accel_dumbbell_z < 36
C
pitch_belt < -43
yaw_belt < -88
E
A
A
C
E
roll_belt >= 126
B
B
pitch_belt >= 1
magnet_belt_z < -322
yaw_arm < -97
C
A
yaw_forearm >= -99
D
D
accel_dumbbell_z < 26
A
E
magnet_forearm_z >= -120
C
A