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

After downloading the data from the data source, we can read the two csv files into two 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. The “classe” variable in the training set is the outcome for us to predict.

Clean the data

We use complete.cases to return a logical vector indicating which cases are complete, i.e., have no missing values.

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

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

First, we remove columns that contain NA missing values.

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

Next, we get rid of some columns that do not contribute much to the accelerometer measurements.

```
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)]
```

Now, the cleaned training data set contains 19622 observations and 53 variables, while the testing data set contains 20 observations and 53 variables. The “classe” variable is still in the cleaned training set.

Slice the data

Then, we can split the cleaned training set into a pure training data set (70%) and a validation data set (30%). We will use the validation data set to conduct cross validation in future steps.

```
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

We use the train function from caret to estimate model performance from a training set We fit a predictive model for activity recognition using **Random Forest** algorithm because it automatically selects important variables and is robust to correlated covariates & outliers in general. We will use **5-fold cross validation** when applying the algorithm.

```
controlRf <- trainControl(method="cv", 5)
modelRf <- train(classe ~ ., data=trainingData, method="rf", trControl=controlRf, ntr
ee=250)
modelRf
```

```
## 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, 10991, 10990, 10989
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD  Kappa SD
##    2    0.9910462  0.9886729  0.001301269  0.001647776
##   27    0.9914102  0.9891334  0.001717547  0.002174708
##   52    0.9850037  0.9810264  0.002718384  0.003439965
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

We use the predict function from caret to to generate predictions for new samples from train. The function confusionMatrix is then used to compute various summaries for classification models on the validation data set.

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

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1672    1    0    0    1
##          B    7 1128    4    0    0
##          C    0    0 1021    5    0
##          D    0    0   14  949    1
##          E    0    0    1    7 1074
##
## Overall Statistics
##
##                Accuracy : 0.993
##                  95% CI : (0.9906, 0.995)
##     No Information Rate : 0.2853
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9912
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9958   0.9991   0.9817   0.9875   0.9981
## Specificity            0.9995   0.9977   0.9990   0.9970   0.9983
## Pos Pred Value         0.9988   0.9903   0.9951   0.9844   0.9926
## Neg Pred Value         0.9983   0.9998   0.9961   0.9976   0.9996
## Prevalence             0.2853   0.1918   0.1767   0.1633   0.1828
## Detection Rate         0.2841   0.1917   0.1735   0.1613   0.1825
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9977   0.9984   0.9903   0.9922   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
```

So, the estimated accuracy of the model is 99.42% and the estimated out-of-sample error is 0.58%.

Predicting for Test Data Set

Now, we apply the model to the original testing data set downloaded from the data source. We remove the `problem_id` column first.

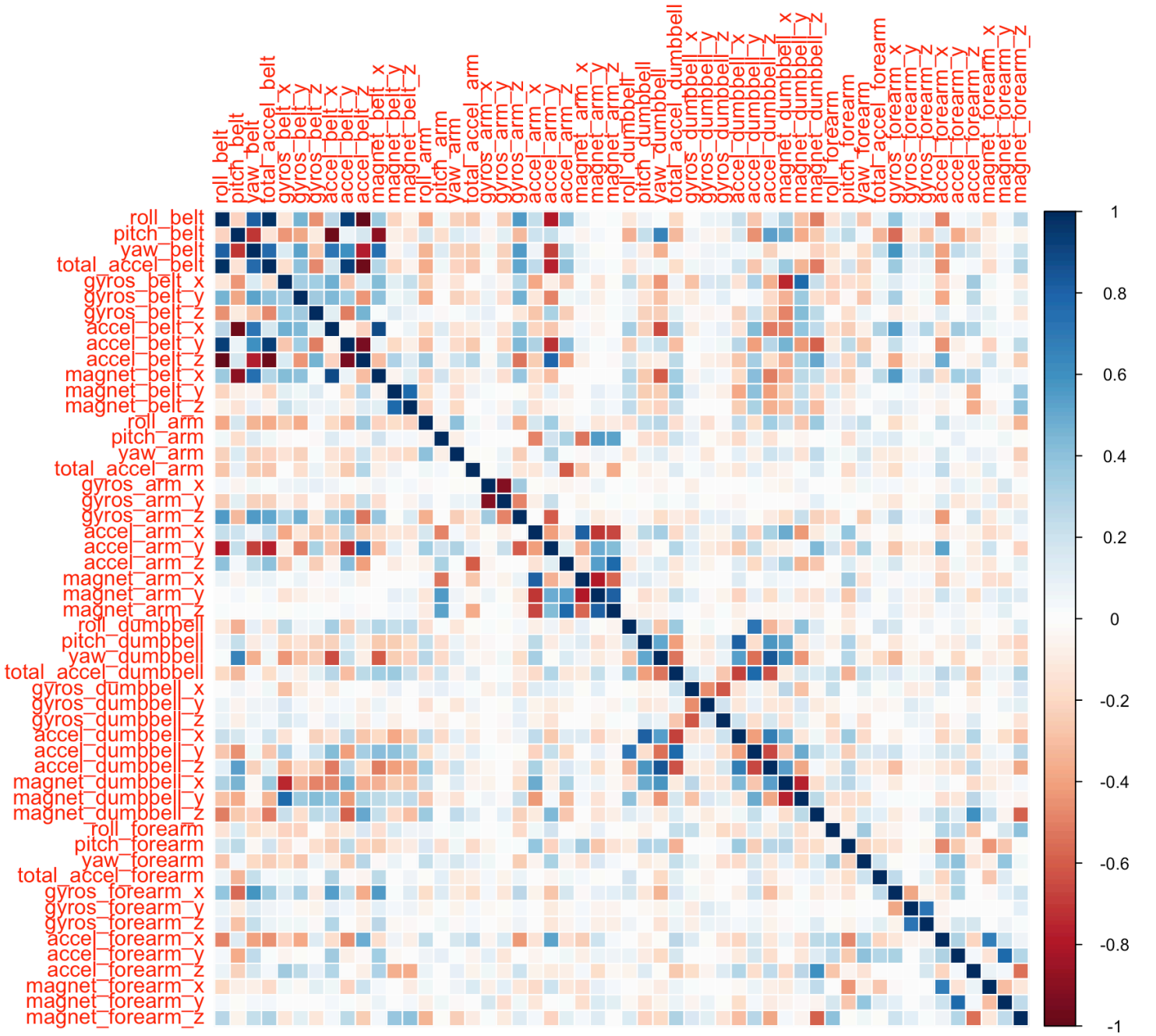
```
result <- predict(modelRf, testCleaned[, -length(names(testCleaned))])
result
```

```
##  [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

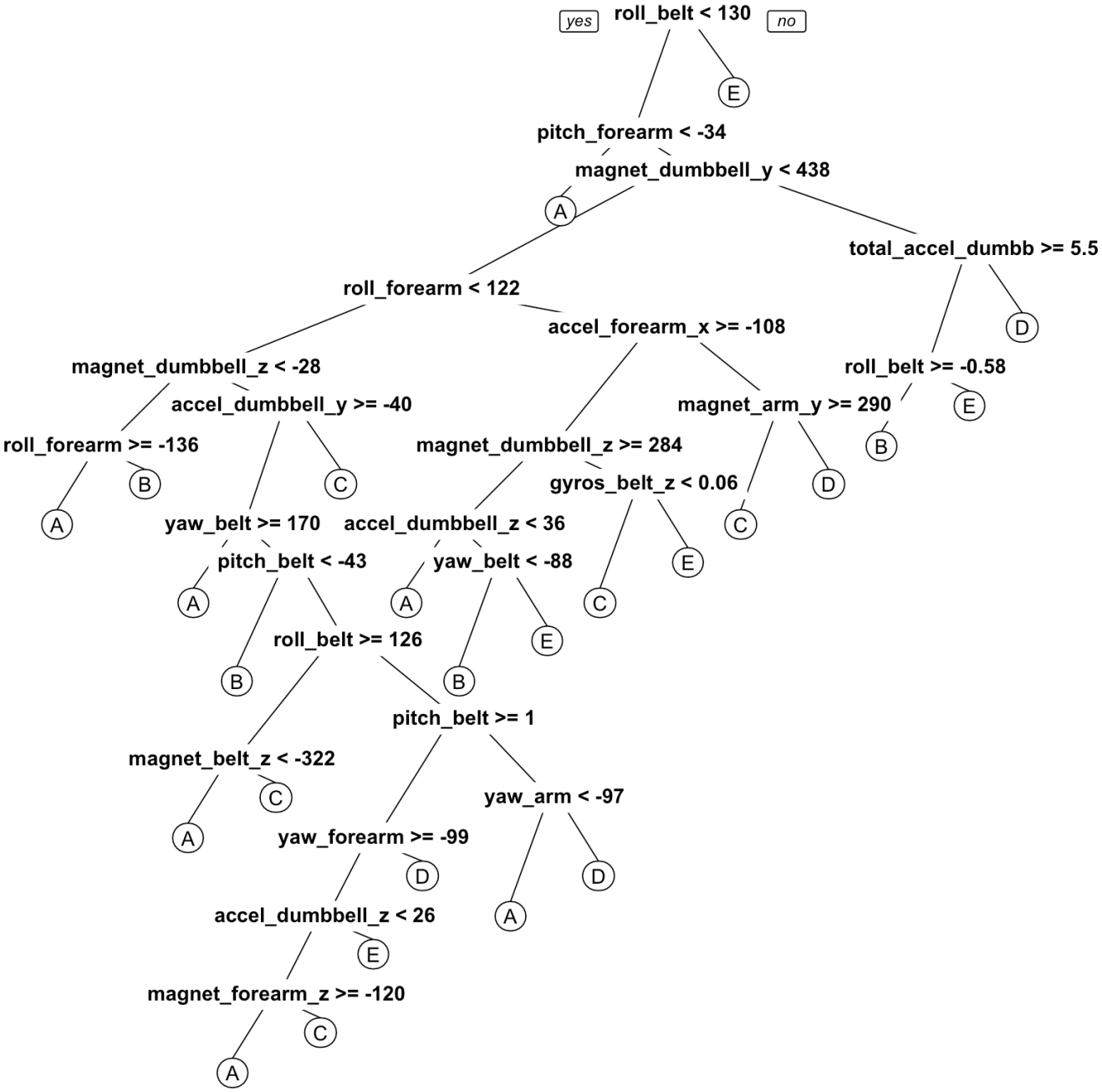
1. Correlation Matrix Visualization

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



2. Decision Tree Visualization

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