Peer-graded Assignment:Prediction Assignmetn Writeup

Christy_Wang

3/15/2020

Background

```
 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. T
hese type of devices are part of the quantified self movement – a group of enthusi
asts 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 pe
ople 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 dat
a from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. Th
ey were asked to perform barbell lifts correctly and incorrectly in 5 different wa
ys.
```

Goals of the Project

To predict the manner in which they did the exercise.

0.Environment preparation

```
# set working directory
setwd("~/Desktop/R learning /Coursera")

# Load the required packages
library(caret)
```

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

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

```
library(rattle)
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.3.0 Copyright (c) 2006-2018 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
library(rpart)
library(rpart.plot)
library(ggplot2)
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:rattle':
##
##     importance
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(rattle)
library(reshape2)
# import the data
training <- read.csv("pml-training.csv", na.strings = c("NA", " "))
testing <- read.csv("pml-testing.csv", na.strings = c("NA", " "))
```

The training set has 19622 observations and 160 variables, and the testing dataset has 20 observations and 160 variables.

1. Data wrangling

1.1 check missing data–delete the variables with >= 80% of the missing data

```
is.na(training)
colSums(is.na(training))
missingcol <- c()
for ( i in 1: ncol(training)) {
  pro_missing <- sum(is.na(training[i]))/nrow(training)

  if (pro_missing >= 0.8){
    missingcol <- c(missingcol, i)
  }
}
summary(missingcol)
newtraining <- training[-missingcol]
```

After removing the variables with a lot of missing data, we now have 93 variables left and remains 19622rows

1.2 Remove the variables with no variability

```
novar <- nearZeroVar(newtraining)
newtraining_1 <- newtraining[-novar]
```

59 varibles left and remains `r nrow(newtraining_1) row

1.3 Remove the identificartion variables

```
newtraining_2 <- newtraining_1[-(1:5)]
```

After the data cleaning steps, I have 54 variables and 19622 observations.

2. Data spliting

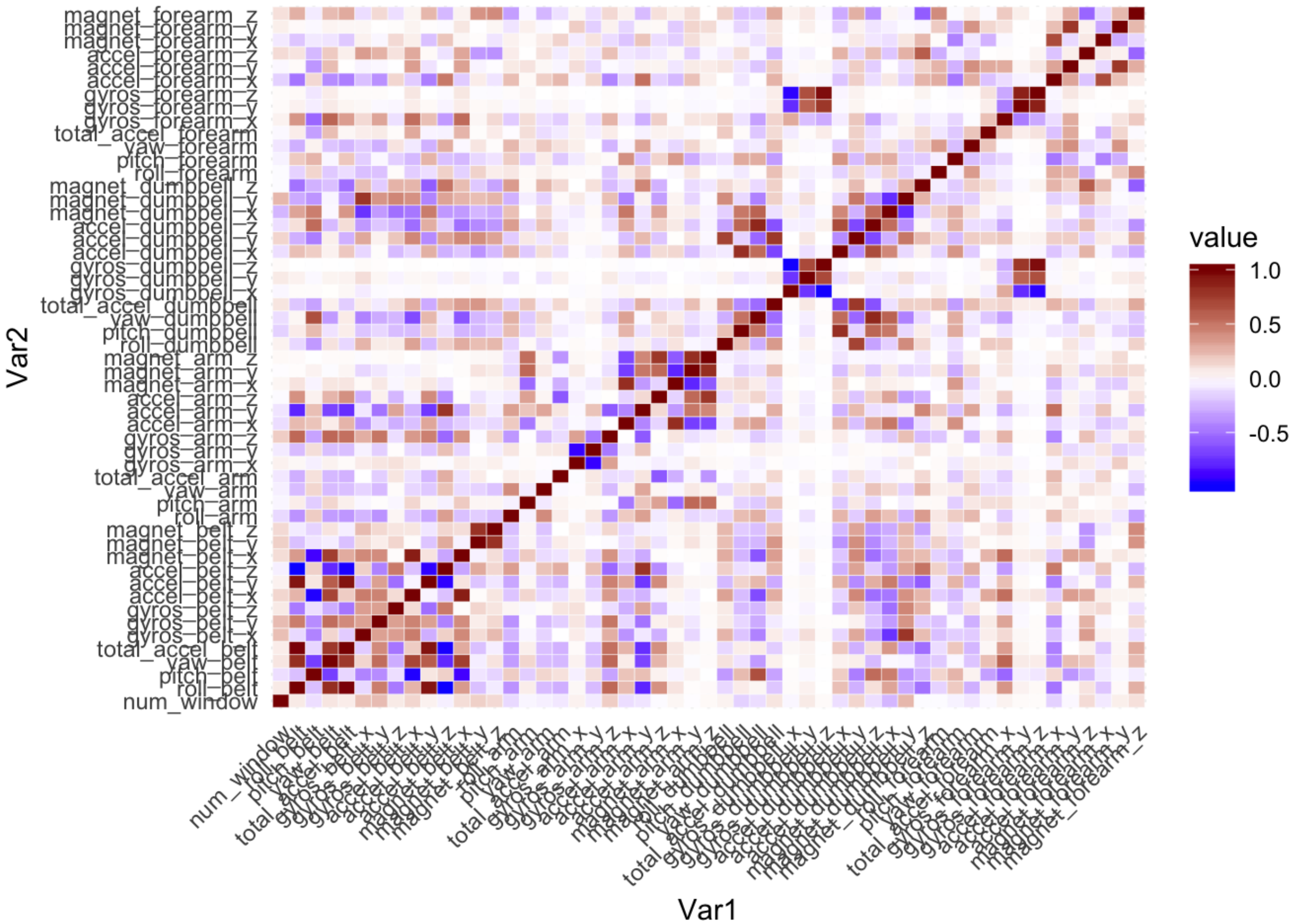
In order to get out-of-sample errors, I split the cleaned training set into a training set with 70% proportion, and the 30% for computing the out-of-sample errors

```
set.seed(123)

inTrain <- createDataPartition(y=newtraining_2$classe, p=0.7, list=FALSE)
newtraining_2_train <- newtraining_2[inTrain,]
newtraining_2_test <- newtraining_2[-inTrain,]
```

3. Exploratory Analysis

Check the correlation among variables



Accoring to the plot, there are some variables having stronger correlations than that with others. A Principle Component Analysis can be utilized to reduce the number of variables.(However, after the attempts of running with PCA, it will take too long to process data. Therefore, I turned to other methods)

4. Prediction Model Selection

4.1 random forest

```
control <- trainControl(method = "cv", number=5)
modFit.rf <- train(classe ~., data=newtraining_2_train, method="rf", trControl=con
trol)
modFit.rf
```

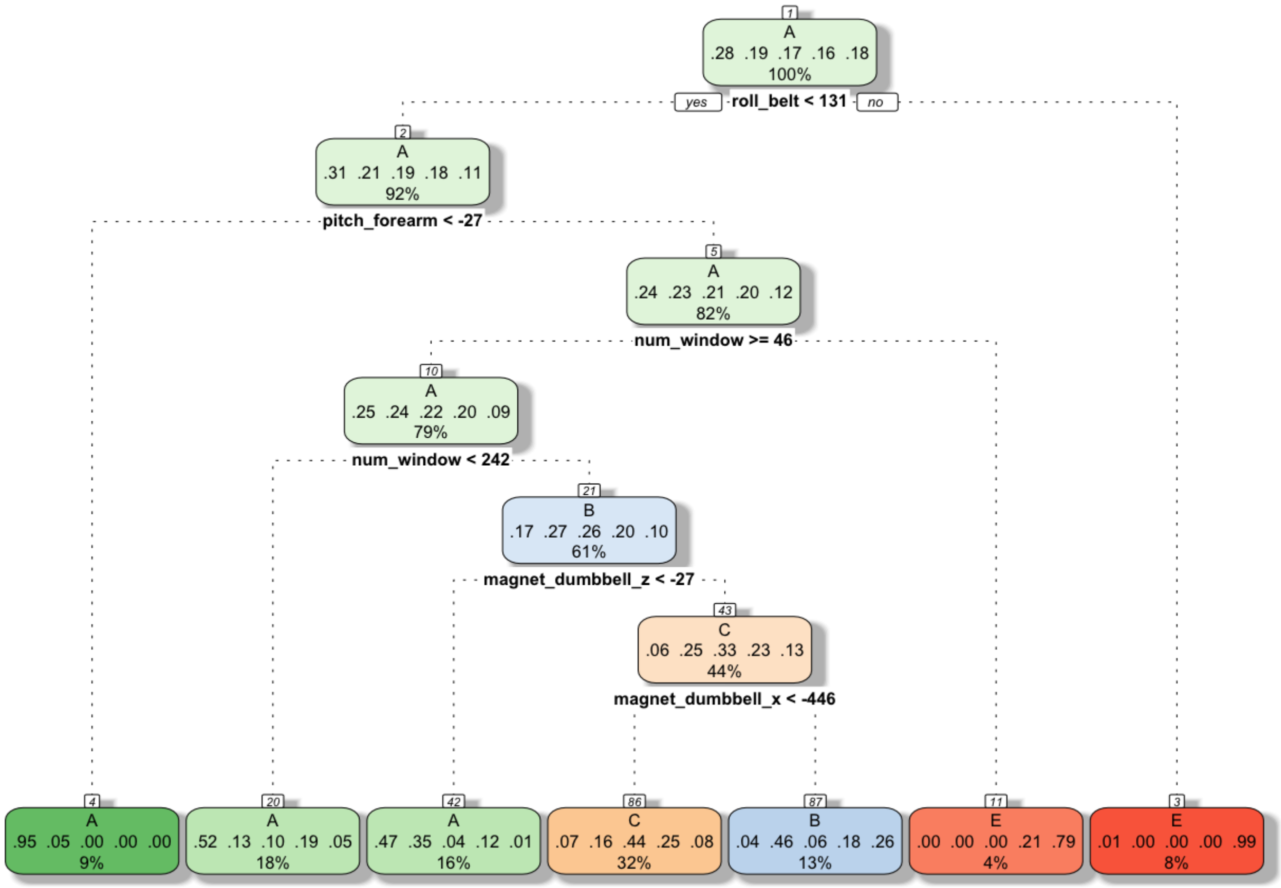
```
## Random Forest
##
## 13737 samples
##    53 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 10990, 10990, 10989, 10988, 10991
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9943221  0.9928172
##   27    0.9969430  0.9961332
##   53    0.9950500  0.9937384
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 27.
```

4.2 Decision Tree

```
set.seed(123)
control <- trainControl(method="cv", number=5)
fit_rpart <- train(classe ~ ., data=newtraining_2_train, method="rpart", trControl
= control)
print(fit_rpart)
```

```
## CART
##
## 13737 samples
##    53 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 10990, 10990, 10988, 10990, 10990
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa
##   0.04292544  0.5386998  0.40581184
##   0.04394263  0.4571563  0.27528096
##   0.11514597  0.3160009  0.04840534
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.04292544.
```

```
fancyRpartPlot(fit_rpart$finalModel)
```



Rattle 2020-Mar-16 15:05:17 ChristyWang

4.3 Support Vector Machine

```
newtraining_2_train$classe <- as.factor(newtraining_2_train$classe)
library(e1071)
svmfit <- svm(classe~., data=newtraining_2_train, kernel="radial")
```

Model Comparison

4.1 Random Forest

```
testrf <- predict(modFit.rf, newtraining_2_test[,-54])
confu.rf <- confusionMatrix(newtraining_2_test$classe,testrf)
confu.rf
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    0    0    0    0
##          B    1 1136    2    0    0
##          C    0    1 1025    0    0
##          D    0    0    1  963    0
##          E    0    0    0    2 1080
## 
## Overall Statistics
## 
##                Accuracy : 0.9988
##                  95% CI : (0.9976, 0.9995)
##     No Information Rate : 0.2846
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.9985
## 
##  Mcnemar's Test P-Value : NA
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   0.9991   0.9971   0.9979   1.0000
## Specificity            1.0000   0.9994   0.9998   0.9998   0.9996
## Pos Pred Value         1.0000   0.9974   0.9990   0.9990   0.9982
## Neg Pred Value         0.9998   0.9998   0.9994   0.9996   1.0000
## Prevalence             0.2846   0.1932   0.1747   0.1640   0.1835
## Detection Rate         0.2845   0.1930   0.1742   0.1636   0.1835
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9997   0.9992   0.9984   0.9989   0.9998
```

4.2 Decision Tree

```
testdt <- predict(fit_rpart, newtraining_2_test[,-54])
confu.dt <- confusionMatrix(newtraining_2_test$classe, testdt)
confu.dt
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1496   38  138    0    2
##          B  474  390  275    0    0
##          C  166   50  810    0    0
##          D  317  137  471    0   39
##          E   59  190  178    0  655
## 
## Overall Statistics
## 
##                Accuracy : 0.5694          
##                  95% CI : (0.5566, 0.5821)
##     No Information Rate : 0.4268          
##     P-Value [Acc > NIR] : < 2.2e-16       
## 
##                   Kappa : 0.4443          
## 
##  Mcnemar's Test P-Value : < 2.2e-16       
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.5955  0.48447   0.4327       NA   0.9411
## Specificity            0.9472  0.85256   0.9462   0.8362   0.9177
## Pos Pred Value         0.8937  0.34241   0.7895       NA   0.6054
## Neg Pred Value         0.7587  0.91256   0.7814       NA   0.9915
## Prevalence             0.4268  0.13679   0.3181   0.0000   0.1183
## Detection Rate         0.2542  0.06627   0.1376   0.0000   0.1113
## Detection Prevalence   0.2845  0.19354   0.1743   0.1638   0.1839
## Balanced Accuracy      0.7714  0.66852   0.6894       NA   0.9294
```

4.3 Support Vector Machine

```
svmPred <- predict(svmfit, newtraining_2_test[,-54])
confu.svm <- confusionMatrix(newtraining_2_test$classe,svmPred)
confu.svm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1654   11    9    0    0
##          B   74 1031   30    2    2
##          C    2   34  985    5    0
##          D    4    0   92  868    0
##          E    1    3   32   25 1021
##
## Overall Statistics
##
##                Accuracy : 0.9446
##                  95% CI : (0.9385, 0.9503)
##     No Information Rate : 0.2948
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9298
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9533   0.9555   0.8580   0.9644   0.9980
## Specificity            0.9952   0.9775   0.9913   0.9807   0.9875
## Pos Pred Value         0.9881   0.9052   0.9600   0.9004   0.9436
## Neg Pred Value         0.9808   0.9899   0.9665   0.9935   0.9996
## Prevalence             0.2948   0.1833   0.1951   0.1529   0.1738
## Detection Rate         0.2811   0.1752   0.1674   0.1475   0.1735
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9742   0.9665   0.9247   0.9726   0.9927
```

5. Prediction on Testing set *Based on the results from the above models, Random Forest has the best performance with accuracy rate 99.9% on tested-training set (Decision tree: 56.9%, Support Vectir Machine: 94.5%). So I will use Random Forest on the test set and estimated that I will get 0.1% of error rate.*

```
predictest<- predict(modFit.rf, testing)
predictest
```

```
##  [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
```

The End