Coursera Practical Machine Learning Project

Jun Zhang

March 5, 2020

```r
library(caret)
library(tidyverse)
library(glmnet)
library(rpart.plot)
```

Data Preprocessing

First, we load the data into R and check the shapes of the datasets.

```r
train <- read.csv("pml-training.csv")
test <- read.csv("pml-testing.csv")

dim(train)
```

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

```r
dim(test)
```

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

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

```r
for(i in 1:ncol(train)){
    if(colnames(train)[i] != colnames(test)[i]){
        print(colnames(train)[i])
        print(colnames(test)[i])
    }
}
```

```
## [1] "classe"
## [1] "problem_id"
```

Then, we can print out the unique variables in the two sets. We can notice that the training set has a variable name *classe*, which does not appear in the testing set. Similarly, the variable *problem_id* appears in the testing set but not the training set. **The goal of this project is to predict the *classe* in the testing set, and the following code shows that there are five classes in total.**

```r
table(train$classe)
```

```
##
##    A    B    C    D    E
## 5580 3797 3422 3216 3607
```

Before diving into the analysis, we first need to take a look at the dataset, and we will find out that there are quite a lot of missing values or blanks.

```r
number <- as.numeric()

#Change all blanks to NAs
train[train==""] <- NA

```

```r
for(i in 1:ncol(train)){
    #If there are no missings in a column
    if(is.na(as.numeric(table((is.na(train[,i])))["TRUE"]))==T){
        number[i] <- 0
    }else{
        #Record the number of missings in a column
        number[i] <- as.numeric(table((is.na(train[,i])))["TRUE"])
    }
}
number
```

```
##   [1]     0     0     0     0     0     0     0     0     0     0     0
##  [12] 19216 19216 19216 19216 19216 19216 19216 19216 19216 19216 19216
##  [23] 19216 19216 19216 19216 19216 19216 19216 19216 19216 19216 19216
##  [34] 19216 19216 19216     0     0     0     0     0     0     0     0
##  [45]     0     0     0     0     0 19216 19216 19216 19216 19216 19216
##  [56] 19216 19216 19216 19216     0     0     0     0     0     0     0
##  [67]     0     0 19216 19216 19216 19216 19216 19216 19216 19216 19216
##  [78] 19216 19216 19216 19216 19216 19216     0     0     0 19216 19216
##  [89] 19216 19216 19216 19216 19216 19216 19216 19216 19216 19216 19216
## [100] 19216 19216     0 19216 19216 19216 19216 19216 19216 19216 19216
## [111] 19216 19216     0     0     0     0     0     0     0     0     0
## [122]     0     0     0 19216 19216 19216 19216 19216 19216 19216 19216
## [133] 19216 19216 19216 19216 19216 19216 19216     0 19216 19216 19216
## [144] 19216 19216 19216 19216 19216 19216 19216     0     0     0     0
## [155]     0     0     0     0     0     0
```

The above function detects the number of missing values in each column. It's obvious to see that many columns have over 19000 missings with only 19622 total observations. Then, we can delete those variables since they provide only a little information.

```r
which(number != 0)
```

```
##  [1]  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28
## [18]  29  30  31  32  33  34  35  36  50  51  52  53  54  55  56  57  58
## [35]  59  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  87
## [52]  88  89  90  91  92  93  94  95  96  97  98  99 100 101 103 104 105
## [69] 106 107 108 109 110 111 112 125 126 127 128 129 130 131 132 133 134
## [86] 135 136 137 138 139 141 142 143 144 145 146 147 148 149 150
```

```r
colRemove <- which(number != 0)
```

```r
trainNew <- train[, -c(1, 3:5, colRemove)]
testNew <- test[, -c(1, 3:5, colRemove)]
```

Since we have removed those columns in the training set, we need to remove them in the testing set. Moreover, since column X indicates the index in the dataset, we can just remove it. Also, by roughly looking at the dataset, the columns from three to five do not provide relevant information. So, we might as well remove them for now.

Train/Test Split

Before modeling, we need to make ourselves a local validation set to test the performance of different statistical models. We can split the training set into a new training set and a new validation set with a 9/1 ratio.

```r
set.seed(123)
split <- sample(1:nrow(trainNew), .9*nrow(trainNew))
```

```
NewTrain <- trainNew[split,]
Validation <- trainNew[-split,]
```

Statistical Models

We begin with the simplest classification method: classification trees. To evaluate the accuracy of different models, we will use 10-fold cross-validation.

```
#CV with 10 folds
Control <- trainControl(method="cv", number=10)

#Classification Tree
set.seed(123)
CT.fit <- train(classe~., data=NewTrain, method="rpart", trControl=Control)

#Prediction
CT.pred <- predict(CT.fit, newdata=Validation[,-ncol(Validation)])

#Results
confusionMatrix(Validation$classe, CT.pred)$table
```

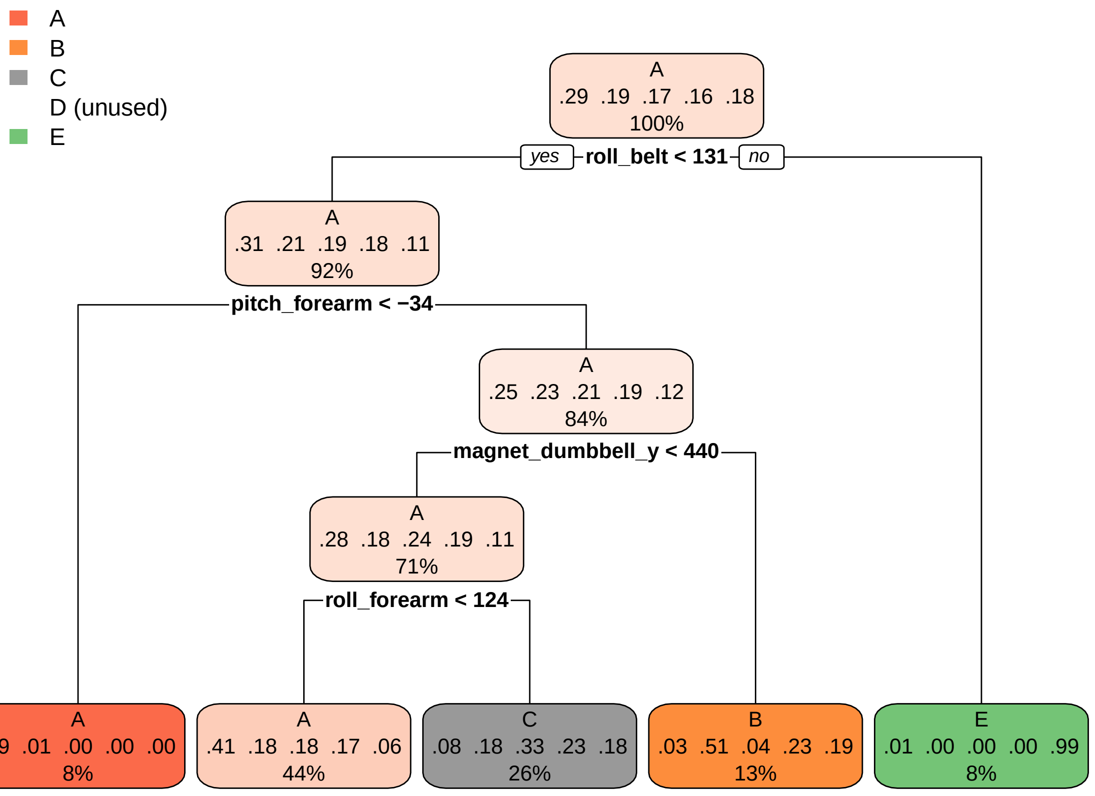
```
##           Reference
## Prediction   A   B   C   D   E
##          A 491   4  28   0   0
##          B 150 128  87   0   0
##          C 158  17 195   0   0
##          D 146  57 138   0   0
##          E  43  53 103   0 165
```

```
confusionMatrix(Validation$classe, CT.pred)$overall[1]
```

```
##  Accuracy
## 0.4987264
```

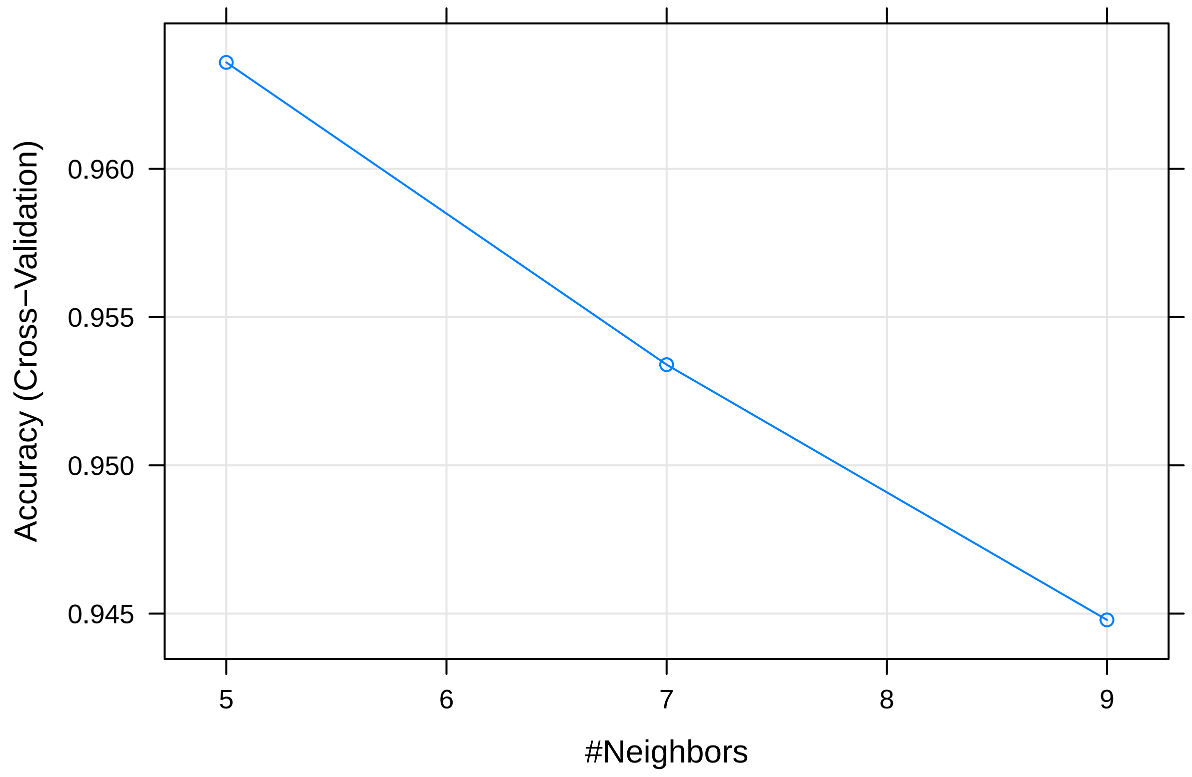
From the result above, we can see that the classification model is about 49.9% accurate, which is not a good model to consider. And below is the graph of the classification tree.

```
rpart.plot(CT.fit$finalModel)
```


A
B
C
D (unused)
E
A .29 .19 .17 .16 .18 100%
yes
roll_belt < 131
no
A .31 .21 .19 .18 .11 92%
pitch_forearm < −34
A .25 .23 .21 .19 .12 84%
magnet_dumbbell_y < 440
A .28 .18 .24 .19 .11 71%
roll_forearm < 124
A .99 .01 .00 .00 .00 8%
A .41 .18 .18 .17 .06 44%
C .08 .18 .33 .23 .18 26%
B .03 .51 .04 .23 .19 13%
E .01 .00 .00 .00 .99 8%

I also tried the method of k-nearest neighbors (KNN). The accuracy at the end is about 96.8%, which is pretty good. And from the graph below, it suggests that when k=5 (5 neighbors), we will have the highest accuracy.

```r
#KNN
set.seed(123)
KNN.fit <- train(classe~., data=NewTrain, method="knn", trControl=Control, preProcess=c("center","scale
KNN.pred <- predict(KNN.fit, newdata=Validation[,-ncol(Validation)])
plot(KNN.fit)
```



```r
confusionMatrix(Validation$classe, KNN.pred)$table
```

```
##           Reference
## Prediction   A   B   C   D   E
##          A 516   0   5   2   0
##          B   6 354   4   1   0
##          C   1  11 352   5   1
##          D   0   1  16 323   1
##          E   2   0   3   3 356
```

```r
confusionMatrix(Validation$classe, KNN.pred)$overall[1]
```

```
##  Accuracy
## 0.9684157
```

Below is the method of quadratic discriminant analysis, and that gives me an accuracy of 90.4%. The result is pretty decent but not as good as KNN.

```r
#QDA
set.seed(123)
QDA.fit <- train(classe~., data=NewTrain, method="qda", trControl=Control, verbose=FALSE)
QDA.pred <- predict(QDA.fit, newdata=Validation[,-ncol(Validation)])

confusionMatrix(Validation$classe, QDA.pred)$table
```

```
##           Reference
## Prediction   A   B   C   D   E
##          A 486  24  11   2   0
##          B   13 318  29   2   3
##          C    0  23 342   3   2
##          D    0   2  36 300   3
##          E    0   8  16  11 329
```

```
confusionMatrix(Validation$classe, QDA.pred)$overall[1]
```

```
##  Accuracy
## 0.9042282
```

I also tried a random forest; however, I wasn't able to obtain the result due to a long runtime. Since the method of KNN provides a prediction with 96.8% accuracy, I will use it as my final model.

Prediction on The Test Data

Using the KNN model we just built, the predicted classes for the test data are shown below.

```
predict(KNN.fit, newdata=test[,-ncol(test)])
```

```
##  [1] B A A A A E D B A A B C B A E E A B B B
## Levels: A B C D E
```

Reference

Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H. Qualitative Activity Recognition of Weight Lifting Exercises. Proceedings of 4th International Conference in Cooperation with SIGCHI (Augmented Human '13) . Stuttgart, Germany: ACM SIGCHI, 2013.