Using Machine Learning to Predict the Execution Quality of the Weight Lifting Exercises

Fang Lu

June 15, 2016

Synopsis: In this assignment, we are required to use machine learning (ML) algorithms to predict how well a weight lifting exercise is performed. The manner of lifting exercise was classified into 5 categories: A, B, C, D, E, which is the response variable (classe). The training data set contains 160 variables. The testing data set contains 20 different individuals whose movement variables (the same as the training data) were also recorded. The goal of this assignment is to accurately predict the lifting exercise manner of the 20 individuals.

To build a simple and efficient ML prediction model: 1) First, the training data set was split into two sets based on “new_window” variable. A small set with the new window equals yes was used for validation data, and a large set with the new window equals no was used as training data to build the prediction model. 2) The training data set was cleaned by removing the predictors contains “NA”, “#DIV/0!”, and blank space. Also, the firs 7 columns were removed due to these features do not contribute to the response variable. Now the training data contains 53 variables, including 52 predictors and 1 response variable. 3) Remove the highly correlated predictors to further reduce the dimenson, simplify the model and increase the model efficacy. The final training data set contains only 42 variables, including 41 predictors, and 1 response variable. 4) Pre-Process with principal component analysis on the training data 5) Build a ML prediction model using random forest algorithm. 6) Validate the prediction model with the validation data set. It turns out that the random forest model achieves 98.8% prediction accuracy. 7) Use the validated model to predict the excercise manner of 20 individuals in the testing data set. And finish the quiz portion.

Here is the R code.

First, load the data and explore the traing data features, cleaning the data set as needed.

```
require(caret)
require(kernlab)
require(randomForest)
require(ggplot2)
require(stats)
training.data<- read.csv("pml-training.csv",header = TRUE, na.strings = c("N
A", " ", "#DIV/0!"))
testing.data<- read.csv("pml-testing.csv",header = TRUE,na.strings = c("NA", "
", "#DIV/0!"))
dim(training.data)
```

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

```
dim(testing.data)
```

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

```
table(training.data$new_window)
```

```
## 
##    no   yes 
## 19216   406
```

```
table(testing.data$new_window)
```

```
## 
## no 
## 20
```

```
# Remove the varialbes with NA in the training data set
training.data<-training.data[,!apply(is.na(training.data),2,any)]
# Split the traing data into training and evaluation two parts based on the new
_window [yes/no]
evaluation.data<-training.data[training.data$new_window=="yes",]
training.data<-training.data[training.data$new_window=="no",]
# Remove the first 7 columns which are not used for model build
training.data<-training.data[,-(1:7)]
# The training data now reduced to contain 19216 obs of 53 variables (original
containes 160 variables)

# standardizing the trainging data
## preObj<-preProcess(training.data[,-53],method=c("center","scale"))
set.seed(1234)
## modelFit<-train(classe~., data=training.data, preProcess=c("center","scal
e"), method ="glm")
## table(training.data2$classe,training.data2$user_name)
```

Check correlated predictors

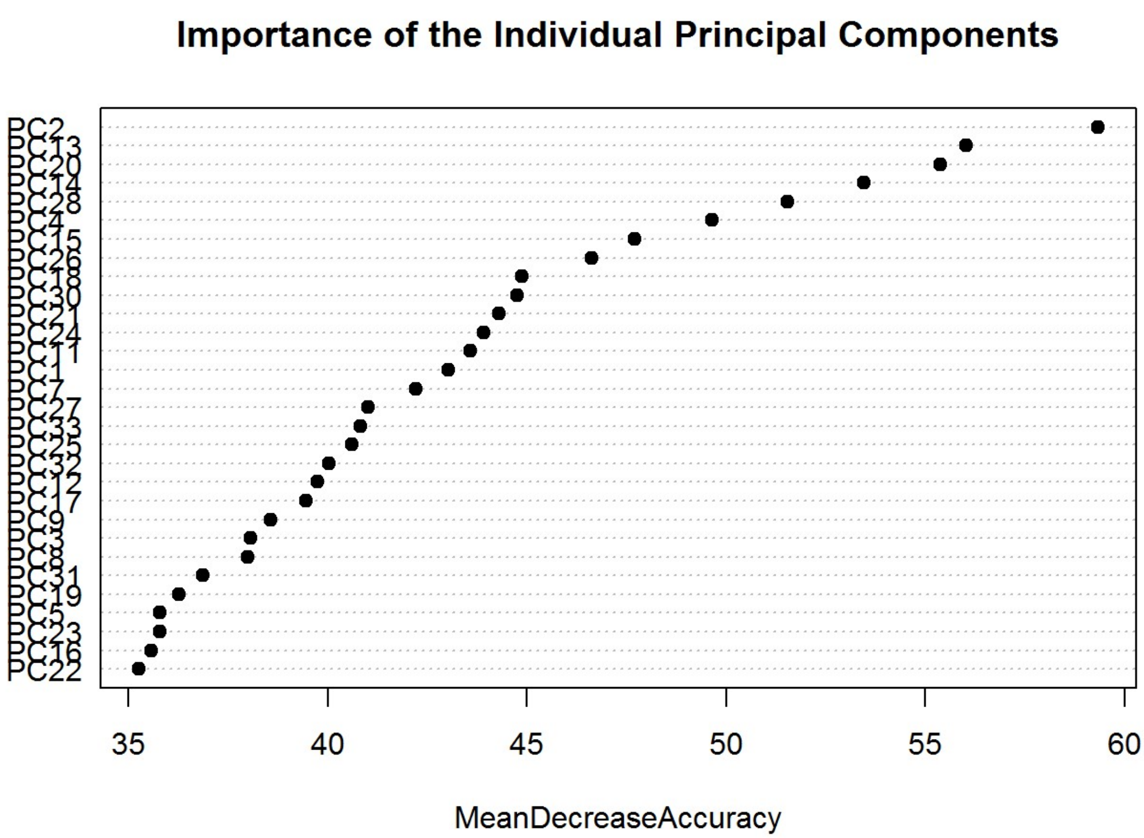
```
M<-abs(cor(training.data[,-53]))
diag(M)<-0
which(M>0.8, arr.ind=T)
```

```
##                  row col
## yaw_belt           3   1
## total_accel_belt   4   1
## accel_belt_y       9   1
## accel_belt_z      10   1
## accel_belt_x       8   2
## magnet_belt_x     11   2
## roll_belt          1   3
## roll_belt          1   4
## accel_belt_y       9   4
## accel_belt_z      10   4
## pitch_belt         2   8
## magnet_belt_x     11   8
## roll_belt          1   9
## total_accel_belt   4   9
## accel_belt_z      10   9
## roll_belt          1  10
## total_accel_belt   4  10
## accel_belt_y       9  10
## pitch_belt         2  11
## accel_belt_x       8  11
## gyros_arm_y       19  18
## gyros_arm_x       18  19
## magnet_arm_x      24  21
## accel_arm_x       21  24
## magnet_arm_z      26  25
## magnet_arm_y      25  26
## accel_dumbbell_x  34  28
## accel_dumbbell_z  36  29
## gyros_dumbbell_z  33  31
## gyros_forearm_z   46  31
## gyros_dumbbell_x  31  33
## gyros_forearm_z   46  33
## pitch_dumbbell    28  34
## yaw_dumbbell      29  36
## gyros_forearm_z   46  45
## gyros_dumbbell_x  31  46
## gyros_dumbbell_z  33  46
## gyros_forearm_y   45  46
```

```
# Remove the highly correlated predictors (r>0.8), resulting in 41 predictors p
lus classe
cleanTrainData<-training.data[,-c(4,9,10,8,19,33,46,24,26,34,36)]

# preprocessing with principal component analysis
## prComp<-prcomp(cleanTrainData[,-42])
preProc<-preProcess(cleanTrainData[,-42], method="pca",thresh = 0.99)
trainPC<-predict(preProc,cleanTrainData[,-42])
## pca_modelFit<-train(cleanTrainData$classe~.,method="glm",data=cleanTrainDat
a)

modelFit <- train(cleanTrainData$classe ~ ., method = "rf", data = trainPC, trC
ontrol = trainControl(method = "cv", number = 4), importance = TRUE)
varImpPlot(modelFit$finalModel, sort = TRUE, type = 1, pch = 19, col = 1, cex
= 1, main = "Importance of the Individual Principal Components")
```



cross-evaluation

```
evaluationPC <- predict(preProc, evaluation.data[, -60])
evaluation.predict <- predict(modelFit, evaluationPC)
evalutation.check <- confusionMatrix(evaluation.data$classe, evaluation.predic
t)
evalutation.check$table
```

```
##           Reference
## Prediction   A   B   C   D   E
##          A 107   1   1   0   0
##          B   0  78   1   0   0
##          C   1   0  69   0   0
##          D   0   0   0  69   0
##          E   0   0   1   0  78
```

```
postResample(evaluation.predict, evaluation.data$classe)
```

```
##  Accuracy     Kappa
## 0.9876847 0.9844872
```

Prediction on the testing data

```
testPC <- predict(preProc, testing.data[, -160])
prediction.testdata<-predict(modelFit, testPC)
prediction.testdata
```

```
##  [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
```