Prediction_pml

YS

8/19/2021

```
library("knitr")
library("ggplot2")
library("caret")
library("plotly")
library("readr")
library("corrplot")
library("rattle")
library("gridExtra")
```

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 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, the goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways.

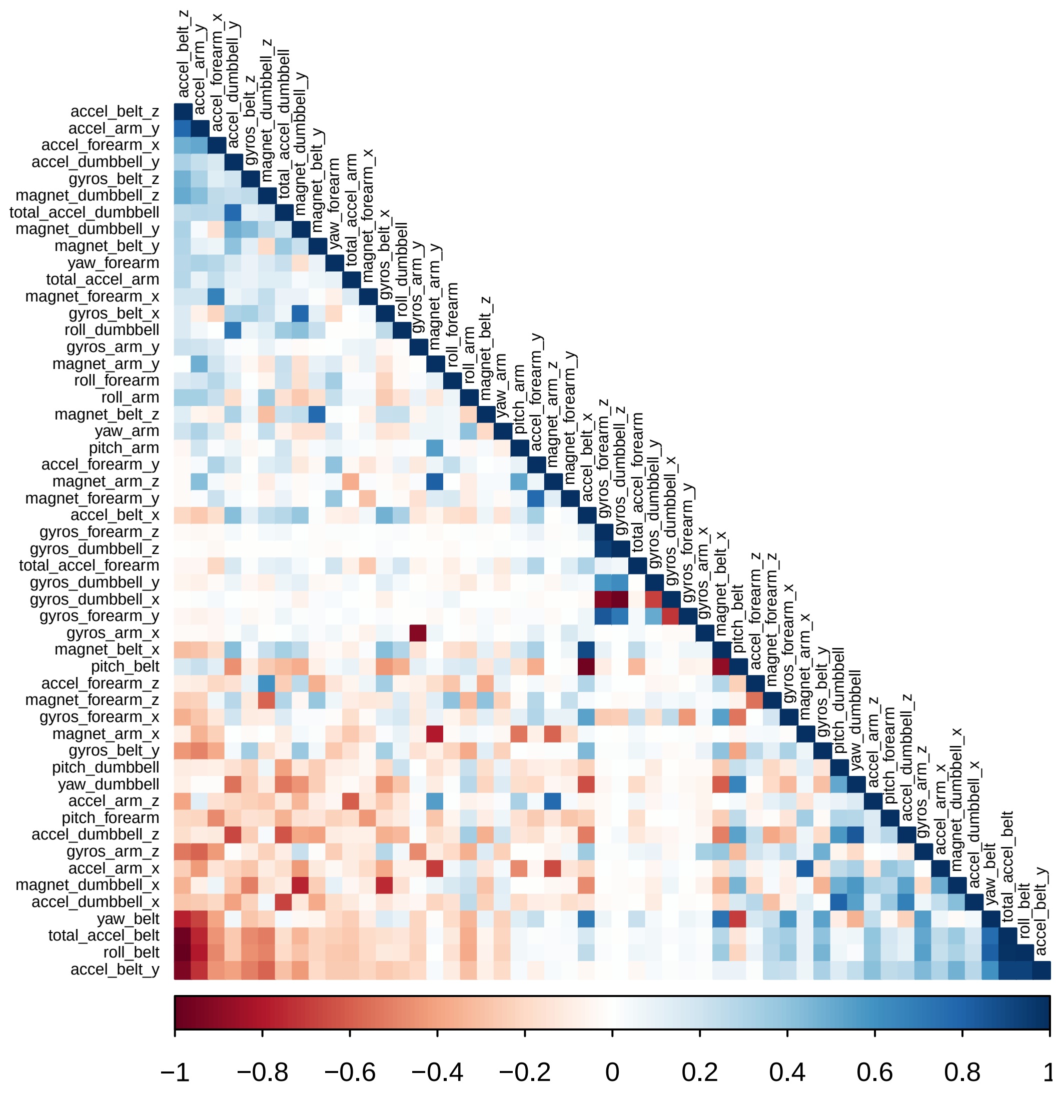
- exactly according to the specification (Class A)
- throwing the elbows to the front (Class B)
- lifting the dumbbell only halfway (Class C)
- lowering the dumbbell only halfway (Class D)
- throwing the hips to the front (Class E).

Data explore

read in data and clean up variable with too many empty entries and low variability.

```
# read in data
na.str = c("NA","Not Available","NOt available","","#DIV/0!","N/A")
pml_train <-read.csv("pml-training.csv", na.strings = na.str)
pml_test <- read.csv("pml-testing.csv", na.strings = na.str)

#check the precentage of NA in the training data, getting ride of the columns that are too much to impu
NA_col<-colSums(is.na(pml_train)/nrow(pml_train))
NA_rm<-names(NA_col[NA_col>0.5])

pml_train<-pml_train[,!(names(pml_train) %in% NA_rm)]
```

```r
pml_test<-pml_test[,!(names(pml_test) %in% NA_rm)]

#check the variability of the variable
library(caret)
nsv <- nearZeroVar(pml_train,saveMetrics=TRUE)
pml_train<-pml_train[,nsv$nzv==FALSE]
pml_test<-pml_test[,nsv$nzv==FALSE]

#get rid of columns that not related to movement detection
pml_train<-pml_train[,-c(1:6)]
pml_test<-pml_test[,-c(1:6)]
```



Result:There are some variables that are highly correlated, dataset may need to be pre-processed using

“PCA” method

Data training and prediction

fitting the training data with different models

```
#split the training dataset into 2 part, training and testing, save the original testing dataset for va
inTrain = createDataPartition(pml_train$classe, p = 3/4)[[1]]
training = pml_train[inTrain,]
testing = pml_train[-inTrain,]

#setup cross validation
trControl=trainControl(method="cv", 5)

#using rpart method for classification training
system.time(
fit_rpart <- train(classe ~ .,method="rpart",trControl=trControl,data = training)
)
```

```
   user  system elapsed
  6.816   0.275   7.117
```

```
#using rpart method for classification training with pca preprocessing
system.time(
fit_rpart_pca <- train(classe ~ .,method="rpart",trControl=trControl,preProcess="pca",data = training)
)
```

```
   user  system elapsed
  8.769   0.810   9.598
```

```
#using random forest method for classification training
system.time(
fit_rf <- train(classe ~ .,method="rf",trControl=trControl,data = training,ntree=250)
)
```

```
   user  system elapsed
323.066   4.660 329.039
```

```
#using random forest method for classification training with pca pre-processing
system.time(
fit_rf_pca <- train(classe ~ .,method="rf",trControl=trControl,preProcess="pca",data = training,ntree=25
)
```

```
   user  system elapsed
206.680   6.496 214.428
```

predict with different models

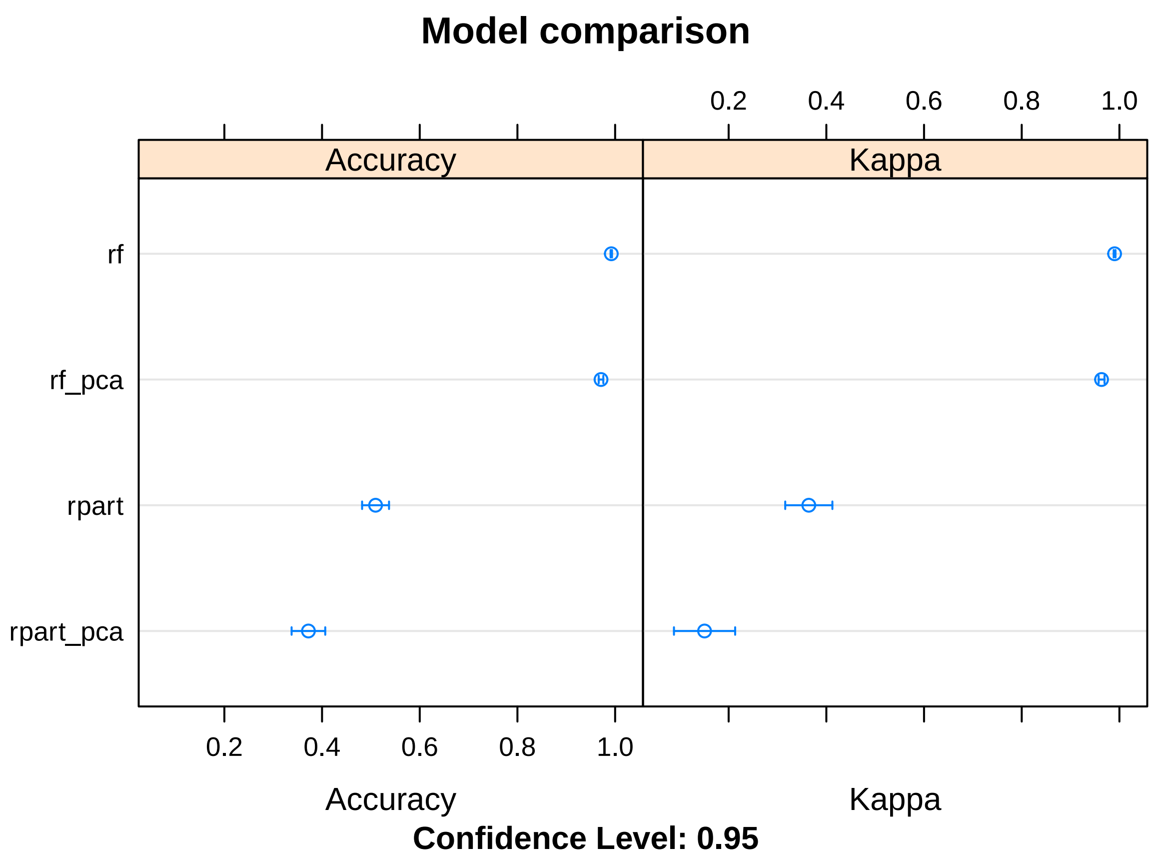
```r
predict_rpart<-predict(fit_rpart,testing)
predict_rpart_pca<-predict(fit_rpart_pca,testing)
predict_rf<-predict(fit_rf,testing)
predict_rf_pca<-predict(fit_rf_pca,testing)
# CM_rpart<-confusionMatrix(as.factor(testing$classe),predict_rpart)
# CM_rpart_pca<-confusionMatrix(as.factor(testing$classe),predict_rpart_pca)
# CM_rf<-confusionMatrix(testing$classe,predict_rf)
# CM_rf_pca<-confusionMatrix(testing$classe,predict_rf_pca)
```

Comparing Models

```r
cvValues <- resamples(list(rpart = fit_rpart, rpart_pca = fit_rpart_pca,
                           rf = fit_rf, rf_pca = fit_rf_pca))

summary(cvValues)
```

```
Call:
summary.resamples(object = cvValues)

Models: rpart, rpart_pca, rf, rf_pca
Number of resamples: 5

Accuracy
               Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
rpart     0.4930343 0.4937139 0.4994903 0.5095748 0.5159647 0.5456706    0
rpart_pca 0.3427310 0.3438668 0.3792049 0.3721288 0.3908319 0.4040095    0
rf        0.9904891 0.9915053 0.9925272 0.9921865 0.9932042 0.9932065    0
rf_pca    0.9683996 0.9684103 0.9700985 0.9711235 0.9711277 0.9775815    0

Kappa
                Min.    1st Qu.    Median      Mean   3rd Qu.      Max. NA's
rpart     0.33756690 0.33760003 0.3474158 0.3641147 0.3679581 0.4300326    0
rpart_pca 0.09680365 0.09744945 0.1684072 0.1506627 0.1867509 0.2039023    0
rf        0.98796946 0.98925307 0.9905451 0.9901151 0.9914028 0.9914050    0
rf_pca    0.96001416 0.96002015 0.9621661 0.9634633 0.9634726 0.9716433    0
```

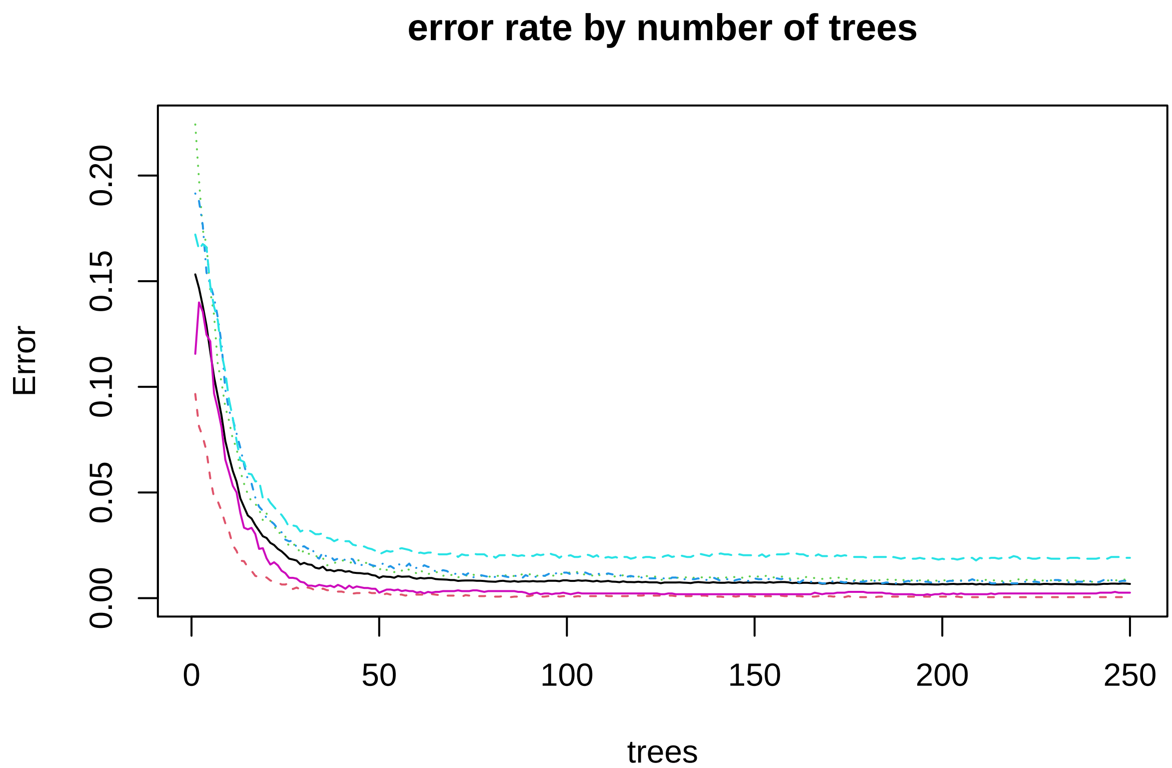
```r
dotplot(cvValues,main = "Model comparison")
```



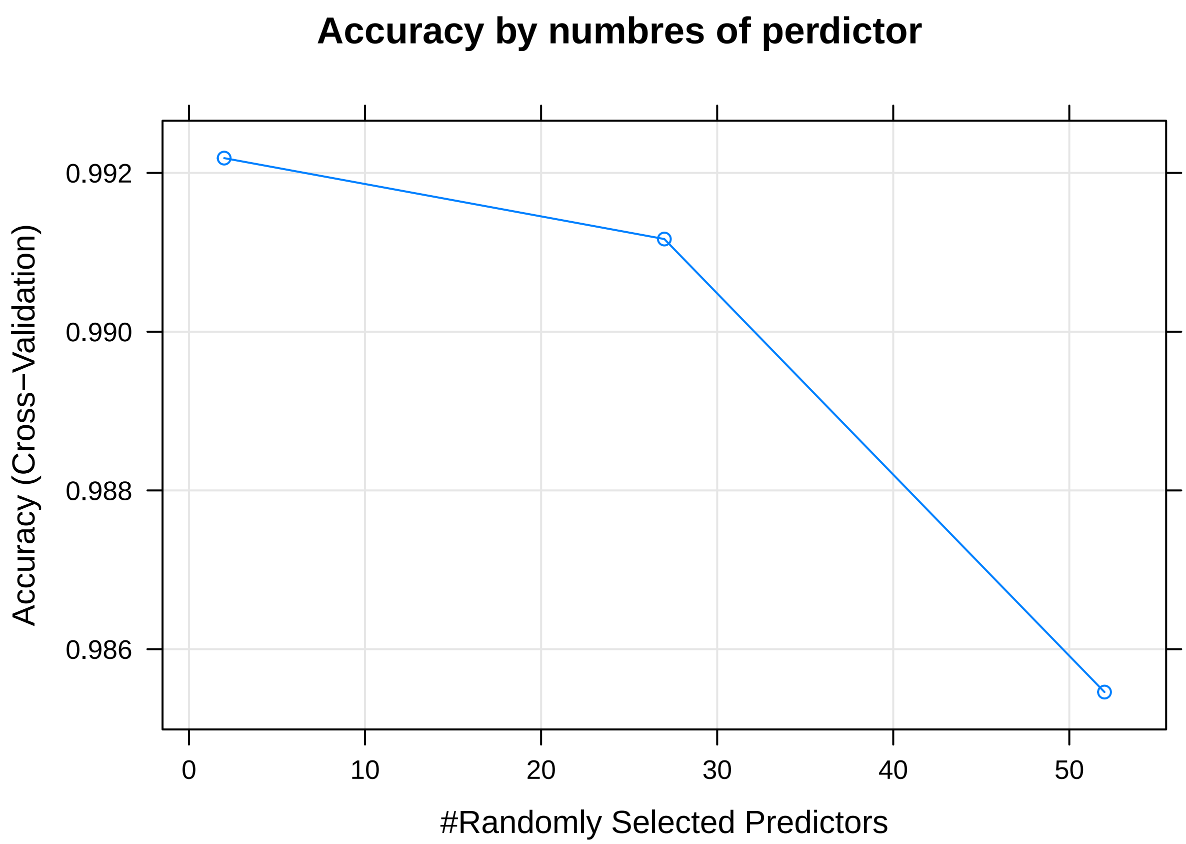
result:

1. Pre-processing with PCA decrease the computing time but not improving the accuracy
2. Random forest with cross validation yield the best result from the selected testing models

```
plot(fit_rf$finalModel,main="error rate by number of trees")
```




```
plot(fit_rf,main="Accuracy by numbres of perdictor")
```




```r
varImp(fit_rf, scale = T)
```

```
rf variable importance

  only 20 most important variables shown (out of 52)

                  Overall
roll_belt          100.00
yaw_belt            83.26
magnet_dumbbell_z   72.10
pitch_belt          61.83
magnet_dumbbell_y   59.78
pitch_forearm       59.73
magnet_dumbbell_x   57.26
roll_forearm        50.28
accel_belt_z        48.06
accel_dumbbell_y    47.35
magnet_belt_z       44.90
magnet_belt_y       41.93
roll_dumbbell       41.84
roll_arm            37.68
accel_dumbbell_z    37.43
accel_forearm_x     34.82
yaw_dumbbell        33.47
gyros_belt_z        33.36
```

```
accel_dumbbell_x    31.43
gyros_dumbbell_y    31.39
```

Final prediction

```
predict_final<-predict(fit_rf,pml_test)
predict_final
```

```
 [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
```