Prediction Assignment Writeup

DV

22/04/2020

```r
library(tinytex)
library(ggplot2)
library(dplyr)
library(caret)
library(randomForest)

library(doParallel)
cluster <- makeCluster(3)
registerDoParallel(cluster)
```

Data importation

```r
training <- read.csv('pml-training.csv', na.strings=c("NA", "#DIV/0!"))
testing <- read.csv('pml-testing.csv', na.strings=c("NA", "#DIV/0!"))
```

Cleaning Data

```r
training_C <- select(training, -contains('timestamp'))
training_C <- select(training_C, -"X")
training_C <- select(training_C, -"user_name")
training_C <- select(training_C, -"new_window")
training_C <- training_C[,colSums(is.na(training_C)) == 0]

testing_C<- select(testing, -contains('timestamp'))
testing_C <- select(testing_C, -"X")
testing_C <- select(testing_C, -"user_name")
testing_C <- select(testing_C, -"new_window")
testing_C <- testing_C[,colSums(is.na(testing_C)) == 0]
```

Modeling

Model split

I have split the data in 70% for training.

```r
set.seed(10)
inTrain <- createDataPartition(training_C$classe, p=0.7, list=F)
trainingPart <- training_C[inTrain,]
testingPart <- training_C[-inTrain,]
```

training Model

I will compare different solution.

```r
start_time <- Sys.time()
model_gbm <- train(classe ~ ., data=trainingPart, method="gbm", verbose=T)
```

Generalized Boosted Regression

```
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.2417
##      2        1.4565             nan     0.1000    0.1625
##      3        1.3535             nan     0.1000    0.1274
##      4        1.2706             nan     0.1000    0.0919
##      5        1.2108             nan     0.1000    0.0984
##      6        1.1498             nan     0.1000    0.0813
##      7        1.0981             nan     0.1000    0.0705
##      8        1.0542             nan     0.1000    0.0633
##      9        1.0133             nan     0.1000    0.0633
##     10        0.9735             nan     0.1000    0.0478
##     20        0.7071             nan     0.1000    0.0285
##     40        0.4689             nan     0.1000    0.0145
##     60        0.3353             nan     0.1000    0.0063
##     80        0.2539             nan     0.1000    0.0052
##    100        0.1958             nan     0.1000    0.0027
##    120        0.1561             nan     0.1000    0.0023
##    140        0.1254             nan     0.1000    0.0012
##    150        0.1125             nan     0.1000    0.0026
```

```r
end_time <- Sys.time()
accuracy.gbm <- model_gbm$results$Accuracy[as.integer(row.names(model_gbm$bestTune))]

errorRate.gbm <- model_gbm$finalModel$err.rate[model_gbm$finalModel$ntree,1]
time.gmb<-end_time-start_time
```

```r
start_time <- Sys.time()
model_rf <- train(classe ~ ., data=trainingPart, method='rf', verbose=T)
end_time <- Sys.time()

accuracy.rf <- model_rf$results$Accuracy[as.integer(row.names(model_rf$bestTune))]
```

```
errorRate.rf <- model_rf$finalModel$err.rate[model_rf$finalModel$ntree,1]

time.rf<-end_time-start_time
```

Random Forest

Choose of the training model

```
a<-matrix(c(accuracy.rf,errorRate.rf,time.rf,accuracy.gbm,errorRate.gbm,time.gmb),nrow=3)
```

```
## Warning in matrix(c(accuracy.rf, errorRate.rf, time.rf, accuracy.gbm,
## errorRate.gbm, : la longueur des données [5] n'est pas un diviseur ni un
## multiple du nombre de lignes [3]
```

```
dimnames(a)=list(c("accuracy","error","exe time"),c("rf","gbm"))
a
```

```
##                   rf        gbm
## accuracy  0.995860836  0.9834894
## error     0.002475067 10.9568169
## exe time 23.828341266  0.9958608
```

The best prediction is with the random forest.

Conclusion

We will use the random forest to answer the quiz.

Prediciton for the testing data

```
quiz_answer<-predict(model_rf$finalModel, newdata=testing_C)

quiz_answer
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  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
```