Prediction Assignment Writeup

ZTH

2020/9/30

Preprocess

Load packages and our datasets

```
#Load the packages we need
library(tidyverse)   #Including ggplot2 and dplyr
library(ggpubr)
library(caret)   #Machine Learning Package
library(e1071)
library(ranger)
library(ordinalForest)
library(randomForest)
library(gbm)
library(plyr)
library(MASS)

#Use parallel to boost
library(parallel)
library(doParallel)
library(foreach)
library(iterators)
```

```
#The working directory has already been set
#Load the data
train <- read.csv("pml-training.csv")
test <- read.csv("pml-testing.csv")
```

Brief overview of our dataset

```
#See the dimention of our train set
print(dim(train))
```

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

```
#See the dimention of our test set
print(dim(test))
```

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

```
#A brief overview of class(outcome)
print(unique(train$classe))
```

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

Select features we need

If we view the train set, we'll find that only samples with the column "new_window" equaling to "yes" have no missing values in all columns

```r
print(nrow(train[train$new_window == "yes",]))
```

```
## [1] 406
```

```r
print(sum(is.na(train[train$new_window == "yes",])))
```

```
## [1] 0
```

But in test set, all samples have "no" value in column "new_window",so I decide to delete those samples with the column "new_window" equaling to "yes" so as to avoid large amount of NA values.

```r
#Delete those NA columns and blank columns as well
new_train <- train %>% filter(new_window == "no")
#Delete NA columns
new_train <- new_train[,-which(apply(new_train,2,function(x) all(is.na(x))))]
#Delete blank columns
new_train <- new_train[,-which(apply(new_train,2,function(x) all(x=="")))]

#Delete the features which are of no relation with measurement,such as user_name,new_window
new_train <- new_train[,7:ncol(new_train)]
```

We want to do the same to test set, but the original train set has the label column "classe", and we see that there are 160 variables both in train set and test set, so there must be 1 column different between them.(Because test set doesn't have column "classe")

```r
#sum of the intersection of column names between original train set and test set
print(length(intersect(colnames(train),colnames(test))))   #159
```

```
## [1] 159
```

```r
#find the difference between them
print(setdiff(colnames(train),colnames(test)))  #classe in train set
```

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

```r
print(setdiff(colnames(test),colnames(train)))  #problem_id in test set
```

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

So we find it! We can just delete column problem_id in test set and do the same as what we do to the train set to get the new test set

```r
#Select intersection
new_test <- test %>% dplyr::select(intersect(colnames(train),colnames(test)))

#Delete NA columns
new_test <- new_test[,-which(apply(new_test,2,function(x) all(is.na(x))))]

#Delete the features which are of no relation with measurement,such as user_name,new_window
new_test <- new_test[,7:ncol(new_test)]
```

We can see the new dimention and if there are still some NAs in the new train set and new test set, and check the dimention and column names

```
#See the new dim of new_train
print(dim(new_train))   #54 includes label column "classe"
```

```
## [1] 19216    54
```

```
#See if there are still some NAs
print(sum(is.na(new_train)))
```

```
## [1] 0
```

```
#See the new dim fo new_test
print(dim(new_test))   #53 doesn't include "classe"
```

```
## [1] 20 53
```

```
#See if there are NAs in new_test
print(sum(is.na(new_test)))
```

```
## [1] 0
```

```
#Intersect again to check
print(length(intersect(colnames(new_train),colnames(new_test))))
```

```
## [1] 53
```

Now we have our tidy train set and test set!

Model building

In this part we'll build our machine learning model.I choose **Linear Discriminant Analysis** *for this classification problem.Here I repeat* **5-fold cross validation for 2 times** *in the new train set,and use parallel method to boost the preocess.*

```
set.seed(123456)
#Here we use parallel method to accelerate the training process
cluster <- makeCluster(detectCores()-1)  # convention to leave 1 core for OS
registerDoParallel(cluster)

#Set the train control,random search the hyperparameters
fitControl <- trainControl(method = "repeatedcv",
                           number = 5,
                           repeats = 2,
                           allowParallel = TRUE
                           )
#Fit the model
lda_fit <- train(classe~.,data = new_train,method = "lda",trControl = fitControl)

#shut down the cluster
stopCluster(cluster)
registerDoSEQ()
lda_fit
```

```
## Linear Discriminant Analysis
##
## 19216 samples
```

```
##    53 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 2 times)
## Summary of sample sizes: 15373, 15375, 15372, 15372, 15372, 15373, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.7125319  0.6362256
```

We get a just-so-so accuracy of 71.25%. But it is enough for this course project. We want to show the process of building our machine learning model rather than a high accuracy.Here is the **error report** *of our train set.*

```r
predictionsTraining <- predict(lda_fit, newdata=new_train)
confusionMatrix(predictionsTraining, new_train$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 4528  509  325  171  142
##          B  156 2435  318  127  518
##          C  360  467 2221  389  320
##          D  407  149  391 2346  330
##          E   20  158   97  114 2218
##
## Overall Statistics
##
##                Accuracy : 0.7154
##                  95% CI : (0.709, 0.7218)
##     No Information Rate : 0.2847
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6399
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8276   0.6549   0.6626   0.7455   0.6287
## Specificity            0.9166   0.9278   0.9032   0.9205   0.9752
## Pos Pred Value         0.7979   0.6851   0.5912   0.6475   0.8508
## Neg Pred Value         0.9304   0.9181   0.9268   0.9486   0.9211
## Prevalence             0.2847   0.1935   0.1744   0.1638   0.1836
## Detection Rate         0.2356   0.1267   0.1156   0.1221   0.1154
## Detection Prevalence   0.2953   0.1850   0.1955   0.1885   0.1357
## Balanced Accuracy      0.8721   0.7914   0.7829   0.8330   0.8019
```

Here we have the final prediction of our 20 examples in test set

```r
predict(lda_fit,new_test)
```

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