Prediction Write Up

Susana 21/02/2021

Synopsis

The goal of the project is to predict the way the subject did the exercise. This is the “classe” variable in the training set. The data was build considering the columns with no NA values and removed the first columns used for identification.

We made an exploratory analysis to look how is the distribution of classe column. We compared 3 different methods random forest (rf), Stochastic Gradient Boosting (gbm) and Linear Discriminant Analysis (lda).

We compared the accuracy to choose a model. The random forest model has the highest Accuracy.

Finally, we used prediction model with the random forest method to predict 20 different test cases.

```
library(AppliedPredictiveModeling)
```

```
## Warning: package 'AppliedPredictiveModeling' was built under R version 4.0.3
```

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.0.2
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.0.3
```

```
## Loading required package: lattice
```

```
set.seed(25)

#Get the data
training0 <- read.table("~/08_machine/pml-training.csv", header=T, sep=",", na.strings=c("NA","#
DIV/0!",""))
testing0 <- read.table("~/08_machine/pml-testing.csv", header=T, sep=",", na.strings=c("NA","#DI
V/0!",""))

#Remove rows data that has NA and remove first identity columns

training <- training0[, colSums(is.na(training0)) == 0 ]
testing0  <- testing0[, colSums(is.na(testing0)) == 0  ]
training <- training[, -c(1:7) ]

 inTrain = createDataPartition(y=training$classe, p =0.7, list=FALSE)
 training = training[ inTrain,]
 testing = training[-inTrain,]
```

Exploratory data analysis

In this section we made an exploratory data analysis first we made use the function dim to see how many rows and columns has the training and testing data, we observed the unique values for variable classe and we made a plot that shows the frequency of variable classe.

```
 dim(training)
```

```
## [1] 13737    53
```

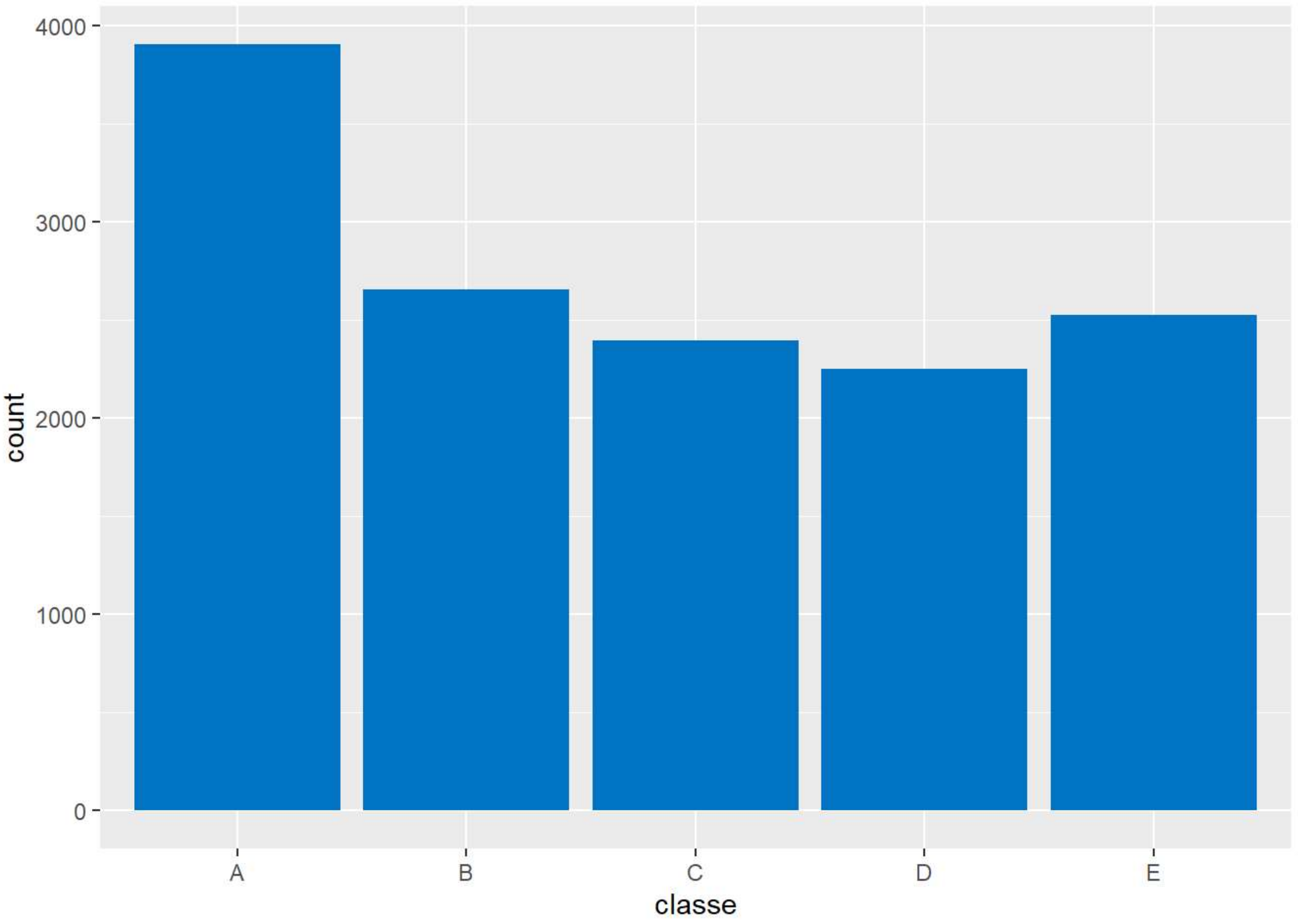
```
 dim(testing)
```

```
## [1] 4131   53
```

```
 unique(training$classe)
```

```
## [1] "A" "B" "C" "D" "E"
```

```
ggplot(training, aes(classe, na.rm=TRUE)) + geom_bar(fill = "#0073C2FF")
```



##

Random Forest Model In this step we made a random forest model. It takes several minutes to execute.

```
modFit1 <- train( classe ~ . , method="rf", data= training, metric ="Accuracy", na.rm=TRUE,
                  trControl=trainControl(method='repeatedcv',
                                         number=2,
                                         repeats=3))


modFit1
```

```
## Random Forest
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (2 fold, repeated 3 times)
## Summary of sample sizes: 6868, 6869, 6869, 6868, 6868, 6869, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9834510  0.9790605
##   27    0.9837179  0.9793995
##   52    0.9765113  0.9702813
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 27.
```

```
modFit1$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry, na.rm = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.68%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3904    1    1    0    0 0.0005120328
## B   16 2632    9    0    1 0.0097817908
## C    0   11 2376    9    0 0.0083472454
## D    0    2   29 2219    2 0.0146536412
## E    0    0    6    7 2512 0.0051485149
```

GBM Model

In this step we made a gradient boosting method for building the model. It takes several minutes to execute.

```
modFit_gbm <- train( classe ~ ., method="gbm", data= training, metric ="Accuracy", verbose=FALSE
,
                     trControl=trainControl(method='repeatedcv',
                                            number=2,
                                            repeats=3))
modFit_gbm
```

```
## Stochastic Gradient Boosting
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (2 fold, repeated 3 times)
## Summary of sample sizes: 6868, 6869, 6868, 6869, 6868, 6869, ...
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.7498969  0.6831093
##   1                  100      0.8183980  0.7701578
##   1                  150      0.8519327  0.8126143
##   2                   50      0.8519328  0.8123747
##   2                  100      0.9014098  0.8752058
##   2                  150      0.9261604  0.9065592
##   3                   50      0.8923103  0.8636515
##   3                  100      0.9363278  0.9194256
##   3                  150      0.9552304  0.9433544
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 150, interaction.depth =
##  3, shrinkage = 0.1 and n.minobsinnode = 10.
```

```
modFit_gbm$finalModel
```

```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 52 predictors of which 52 had non-zero influence.
```

Linear Discriminant Analysis

Finally, we made a model using the method Linear Discriminant Analysis.

```
modFit_lda <- train( classe ~ ., method="lda", data= training, metric ="Accuracy", verbose=FALSE
, na.rm=TRUE,
                     trControl=trainControl(method='repeatedcv', number=2, repeats=3) )
modFit_lda
```

```
## Linear Discriminant Analysis
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (2 fold, repeated 3 times)
## Summary of sample sizes: 6869, 6868, 6868, 6869, 6868, 6869, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.6976779  0.6174748
```

```
#modFit_lda$finalModel
```

Accuracy results

The Model with the highest accuracy was random forest, and the results observed with the sample error of the models, then random forest was used for prediction.

rf – Accuracy :0.9834510

Prediction

In this step we predict 20 different test cases for the variable classe.

```
predict(modFit1, testing0 )
```

```
##  [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
```