Course project Practical Machine learning

Bence Szikora

2025-03-05

Overview

In this project I analyzed a Weight Lifting Exercise Dataset and use some data about personal activity to predict the way how the lifting is performed. A few different prediction algorithm was tested and the best model was choosen based on accuracy and error rates.

Data

The training data for this project are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv)

The test data are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv)

The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har)

Loading and formatting data

Load the required R packages, and data files. I also set the seed and got a brief overview about the data

```
library(corrplot)
```

```
## Warning: a(z) 'corrplot' csomag az R 4.4.3 verziójával lett fordítva
```

```
## corrplot 0.95 loaded
```

```
library(caret)
```

```
## Warning: a(z) 'caret' csomag az R 4.4.3 verziójával lett fordítva
```

```
## A szükséges csomag betöltődik: ggplot2
```

```
## A szükséges csomag betöltődik: lattice
```

```
training = read.csv("training_ds_machine_learning")
testing = read.csv("testing_ds_machine_learning")
#str(training)
set.seed(417)
```

The first step was to choose the variables for the prediction. I checked for near-zero-variance (NZV) variables and I excluded them from the analysis. Next I checked for variables with NA values and I also excluded them. Finally I removed the identification variables as well.

```
#removing zero covariates

nsv=nearZeroVar(training)
training=training[,-nsv]


#removing columns with NA values
na= sapply(training,function(x) sum(is.na(x))==0)
training=training[,na==TRUE]


#removing the first 5 columns, because they are just for identification
training=training[,-(1:5)]
```

I also made some data analysis with the cleaned dataset. I checked the type of the variables and the basic information about the numeric ones. I made a correlation analysis as well to see whether this variable are highly correlated with each other or not. I found that only the 5% of the variables were highly correlated so I rejected the idea to make a PCA.

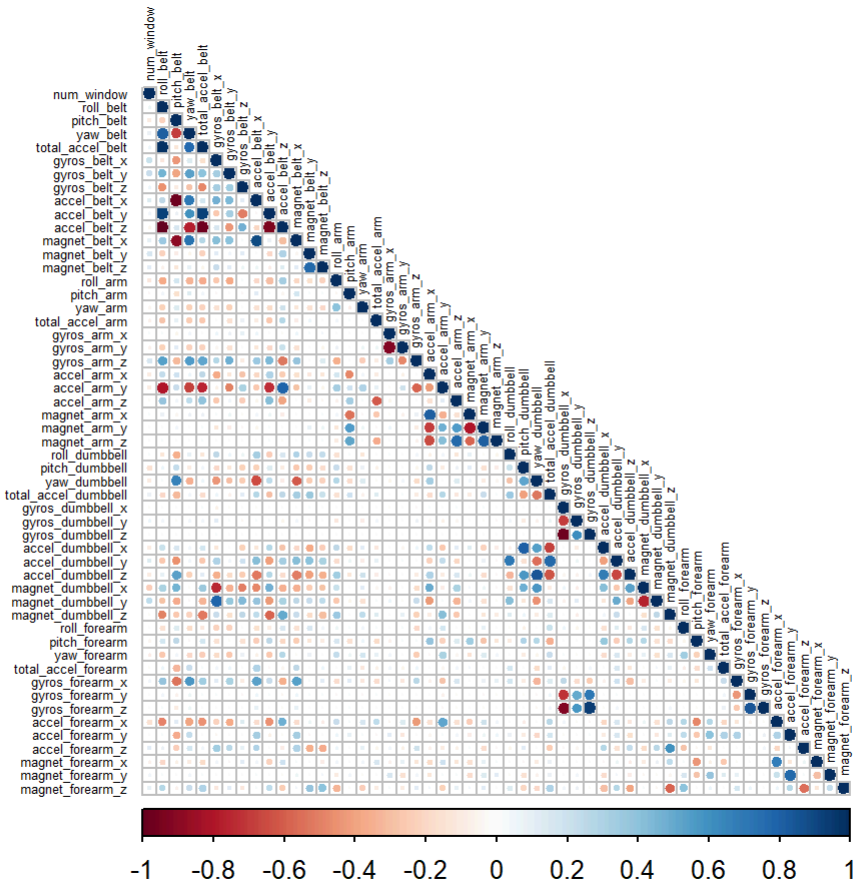
```
#Checking the class of the variables
#sapply(training, function(x) class(x))

#I checked the basic information about the dataset
#summary(training)


#Checking the correlations between the variables, few are highly (abs(cor)>0.75) correlated
(only 5%)
cor=cor(training[,-54])
sum(abs(cor)>0.75)/sum(abs(cor)<=0.75)
```

```
## [1] 0.04268745
```

```
#visualizing the correlation just because its nice
corrplot(corr = cor,type="lower", tl.col = "black",tl.cex = 0.4)
```



Cross validation

I made a testing and a training group for cross validation with random sampling.

```
#cross validation, with random subsampling

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

Prediction models

I choosed 3 different model to predict the classe variable in the Test set.

Generalized Boosted model (gbm):

```
##Prediction models
#gbm

gbm= train(classe~.,method="gbm",data=Train)
```

```
predict_gbm=predict(gbm,Test)
conf_gbm =confusionMatrix(predict_gbm,as.factor(Test$classe))
conf_gbm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1669    9    0    0    2
##          B    5 1113    3    4    4
##          C    0   14 1019   11    1
##          D    0    3    3  949   14
##          E    0    0    1    0 1061
##
## Overall Statistics
##
##                Accuracy : 0.9874
##                  95% CI : (0.9842, 0.9901)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9841
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9970   0.9772   0.9932   0.9844   0.9806
## Specificity            0.9974   0.9966   0.9946   0.9959   0.9998
## Pos Pred Value         0.9935   0.9858   0.9751   0.9794   0.9991
## Neg Pred Value         0.9988   0.9945   0.9986   0.9969   0.9956
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2836   0.1891   0.1732   0.1613   0.1803
## Detection Prevalence   0.2855   0.1918   0.1776   0.1647   0.1805
## Balanced Accuracy      0.9972   0.9869   0.9939   0.9902   0.9902
```

Decision Tree model:

```
#tree
tree=train(classe~.,method="rpart",data=Train)
predict_tree=predict(tree,Test)
conf_tree=confusionMatrix(predict_tree,as.factor(Test$classe))
conf_tree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1456  258  155  158   25
##          B   19  384   29  164   83
##          C  178  497  842  598  301
##          D    0    0    0    0    0
##          E   21    0    0   44  673
##
## Overall Statistics
##
##                Accuracy : 0.5701
##                  95% CI : (0.5573, 0.5828)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4515
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8698  0.33714   0.8207   0.0000   0.6220
## Specificity            0.8585  0.93784   0.6761   1.0000   0.9865
## Pos Pred Value         0.7096  0.56554   0.3485      NaN   0.9119
## Neg Pred Value         0.9431  0.85498   0.9470   0.8362   0.9205
## Prevalence             0.2845  0.19354   0.1743   0.1638   0.1839
## Detection Rate         0.2474  0.06525   0.1431   0.0000   0.1144
## Detection Prevalence   0.3487  0.11538   0.4105   0.0000   0.1254
## Balanced Accuracy      0.8641  0.63749   0.7484   0.5000   0.8042
```

Random forest model:

```
#forest
forest=train(classe~.,method="rf",data=Train)
predict_forest= predict(forest,Test)
conf_forest=confusionMatrix(predict_forest,as.factor(Test$classe))
conf_forest
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    5    0    0    0
##          B    0 1132    2    0    0
##          C    0    2 1024    6    0
##          D    0    0    0  958    6
##          E    0    0    0    0 1076
##
## Overall Statistics
##
##                Accuracy : 0.9964
##                  95% CI : (0.9946, 0.9978)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9955
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9939   0.9981   0.9938   0.9945
## Specificity            0.9988   0.9996   0.9984   0.9988   1.0000
## Pos Pred Value         0.9970   0.9982   0.9922   0.9938   1.0000
## Neg Pred Value         1.0000   0.9985   0.9996   0.9988   0.9988
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1924   0.1740   0.1628   0.1828
## Detection Prevalence   0.2853   0.1927   0.1754   0.1638   0.1828
## Balanced Accuracy      0.9994   0.9967   0.9982   0.9963   0.9972
```

Results

Based on the predictive accuracy (gbm=98,74%, Decision Tree= 57,01%, random forest= 99,64%), and the error rates and kappa values, the random forest model performed the best, so I will use this one for my predictions on the quiz dataset. Nevertheless the gbm model was also very powerful.