Practical Machine Learning Project

2024-07-22

Background

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 – a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. 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, your 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. More information is available from the website here (see the section on the Weight Lifting Exercise Dataset). The goal of this project is to predict the manner in which they did the exercise.

Data

The training data for this project are available here

The test data are available here

The data for this project come from this source: http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har .

Load data and libraries

```
# Read files
training <- read.csv("pml-training.csv")
testing <- read.csv("pml-testing.csv")

# Load libraries
library(caret)
library(nnet)
library(e1071)
library(xgboost)
library(kernlab)
library(corrplot)
library(randomForest)
library(ggplot2)
```

Data preprocessing

```
# Remove unnecessary variables and NA variables.
training <- training[,-c(1:7)]
training <- training[,colMeans(is.na(training)) < 0.9]
```

```r
# Remove near zero variance variables.
nzv <- nearZeroVar(training)
training <- training[,-nzv]
dim(training)
```

```
## [1] 19622    53
```

```r
# Subset sub-training and validation set
set.seed(42)
inTrain <- createDataPartition(training$classe, p=0.7, list=F)
subTrain <- training[inTrain,]
validation <- training[-inTrain,]
```

Model fitting and Cross validation

I will compare 4 models such as Multinomial Linear Model, Random Forest, Gradient Boosted Trees, Support Vector Machine.

To evaluate the generalization performance of our model, I used 5-fold cross validation.

```r
# Set up 5-fold cross validation
set.seed(42)
train_control <- trainControl(method="cv", number=5, verboseIter=FALSE)
```

Multinomial Linear Model

```r
# Model fitting
mnom_model <- train(classe ~., data=subTrain, method="multinom", trControl=train_control, trace=FALSE)

# Prediction
mnom_pred <- predict(mnom_model, validation)
mnom_CM <- confusionMatrix(mnom_pred, factor(validation$classe))
mnom_CM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1447  171  161  132   74
##          B   50  559   96   27  159
##          C   53  156  612   97   72
##          D  102  103   95  659  142
##          E   22  150   62   49  635
##
## Overall Statistics
##
##                Accuracy : 0.6647
##                  95% CI : (0.6525, 0.6768)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5735
##
##  Mcnemar's Test P-Value : < 2.2e-16
```

```
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8644  0.49078   0.5965   0.6836   0.5869
## Specificity            0.8722  0.93005   0.9222   0.9102   0.9411
## Pos Pred Value         0.7290  0.62738   0.6182   0.5985   0.6917
## Neg Pred Value         0.9418  0.88386   0.9154   0.9362   0.9100
## Prevalence             0.2845  0.19354   0.1743   0.1638   0.1839
## Detection Rate         0.2459  0.09499   0.1040   0.1120   0.1079
## Detection Prevalence   0.3373  0.15140   0.1682   0.1871   0.1560
## Balanced Accuracy      0.8683  0.71041   0.7593   0.7969   0.7640
```

Random Forest

```
# Model fitting
rf_model <- train(classe ~., data=subTrain, method="rf", trControl=train_control)

# Prediction
rf_pred <- predict(rf_model, validation)
rf_CM <- confusionMatrix(rf_pred, factor(validation$classe))
rf_CM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1672    1    0    0    0
##          B    2 1136    6    0    0
##          C    0    2 1020   14    0
##          D    0    0    0  949    5
##          E    0    0    0    1 1077
##
## Overall Statistics
##
##                Accuracy : 0.9947
##                  95% CI : (0.9925, 0.9964)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9933
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9988   0.9974   0.9942   0.9844   0.9954
## Specificity            0.9998   0.9983   0.9967   0.9990   0.9998
## Pos Pred Value         0.9994   0.9930   0.9846   0.9948   0.9991
## Neg Pred Value         0.9995   0.9994   0.9988   0.9970   0.9990
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2841   0.1930   0.1733   0.1613   0.1830
## Detection Prevalence   0.2843   0.1944   0.1760   0.1621   0.1832
## Balanced Accuracy      0.9993   0.9978   0.9954   0.9917   0.9976
```

Gradient Boosted Trees

```
# Model fitting
gbm_model <- train(classe ~., data=subTrain, method="gbm", trControl=train_control, verbose=FALSE)

# Prediction
gbm_pred <- predict(gbm_model, validation)
gbm_CM <- confusionMatrix(gbm_pred, factor(validation$classe))
gbm_CM
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1647   47    0    1    1
##          B   13 1060   34    4   11
##          C   10   29  978   23    6
##          D    1    1   11  932   20
##          E    3    2    3    4 1044
## 
## Overall Statistics
## 
##                Accuracy : 0.9619
##                  95% CI : (0.9567, 0.9667)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.9518
## 
##  Mcnemar's Test P-Value : 3.757e-08
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9839   0.9306   0.9532   0.9668   0.9649
## Specificity            0.9884   0.9869   0.9860   0.9933   0.9975
## Pos Pred Value         0.9711   0.9447   0.9350   0.9658   0.9886
## Neg Pred Value         0.9936   0.9834   0.9901   0.9935   0.9921
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2799   0.1801   0.1662   0.1584   0.1774
## Detection Prevalence   0.2882   0.1907   0.1777   0.1640   0.1794
## Balanced Accuracy      0.9861   0.9588   0.9696   0.9800   0.9812
```

Support Vector Machine

```
# Model fitting
svm_model <- train(classe ~., data=subTrain, method="svmRadial", trControl=train_control, verbose=FALSE

# Prediction
svm_pred <- predict(svm_model, validation)
svm_CM <- confusionMatrix(svm_pred, factor(validation$classe))
svm_CM
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1665  121    6    4    1
##          B    4  973   51    3   11
##          C    5   41  960   83   39
##          D    0    0    8  873   28
##          E    0    4    1    1 1003
## 
## Overall Statistics
## 
##                Accuracy : 0.9302
##                  95% CI : (0.9233, 0.9365)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.9114
## 
##  Mcnemar's Test P-Value : < 2.2e-16
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9946   0.8543   0.9357   0.9056   0.9270
## Specificity            0.9687   0.9855   0.9654   0.9927   0.9988
## Pos Pred Value         0.9265   0.9338   0.8511   0.9604   0.9941
## Neg Pred Value         0.9978   0.9657   0.9861   0.9817   0.9838
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2829   0.1653   0.1631   0.1483   0.1704
## Detection Prevalence   0.3054   0.1771   0.1917   0.1545   0.1715
## Balanced Accuracy      0.9816   0.9199   0.9505   0.9491   0.9629
```

Model selection

```r
models <- c("Multinom", "RF", "GBM", "SVM")
accuracy <- c(mnom_CM$overall[1], rf_CM$overall[1], gbm_CM$overall[1], svm_CM$overall[1])
oos_error <- 1-accuracy

data.frame(accuracy = accuracy, oos_error = oos_error, row.names = models)
```

```
##           accuracy  oos_error
## Multinom 0.6647409 0.33525913
## RF       0.9947324 0.00526763
## GBM      0.9619371 0.03806287
## SVM      0.9301614 0.06983857
```

The best model is Random Forest model, which has the highest accuracy (0.9947324) and the lowest expected out of sample error (0.0052676). I used this model to apply test set.

Prediction on Test set

```
pred_test <- predict(rf_model, testing)
pred_test
```

```
##  [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
```