Prediction assignment

Jan Kips

2024-10-18

Introduction

Goal of this project is to predict how well people perform a certain activity. We will do so using data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. ‘how well’ is defined as a “classe”. The goal is to build a model to predict the “classe” of 20 test cases, based on training data.

Data processing

```
library(caret)
```

```
## Loading required package: ggplot2
```

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

```
training_raw<-read.csv("pml-training.csv")
testing_raw<-read.csv("pml-testing.csv")
training_raw$classe<-as.factor(training_raw$classe)
dim(training_raw)
```

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

```
#remove parameters that are irrelevant for predicting the classe
trainRemove <- grepl("^user|^X|timestamp|window", names(training_raw))
training1<-training_raw[,!trainRemove]

#restrict to the complete cases
# training2<-training1[complete.cases(training_raw),]

# remove rows with missing values
training2<- training1[, colSums(is.na(training1)) == 0]

# Removing near-zero covariates
near_zero<-nearZeroVar(training2, saveMetrics=TRUE)
training3<-training2[,!near_zero[4]]
training_cleaned<-training3
dim(training_cleaned)
```

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

We reduced the number of variables from 160 to 53.

Cross-validation

The cleaned dataset is split in a training and a validation set. Note: In theory, one should use ca 70% of the data for training the prediction model and ca 30% for validation. In practice, working with such large training sets to calculate the prediction model really takes a LOT of computing time on my machine, so for this assessment I trained the prediction models on only 20% of the total training dataset. While this obviously will have an impact on the accuracy of the prediction model, it does not take away any of the learning opportunity while doing this assessment.

```
    #careful: the split should be set to 70/30 for training/validation, but doing so takes a LO
T of computing time to calculate the prediction models.
    #therefore, for now using only 10% for the training dataset to allow troubleshooting the sy
ntax.
inTrain <- createDataPartition(training_cleaned$classe, p=0.1, list=F)
trainData<-training_cleaned[inTrain,]
valData<-training_cleaned[-inTrain,]
```

Comparing prediction models

Random Forest

We start with random forest model:

```
    #train model
    rf_model<-train(classe~.,data=trainData,method="rf",allowParallel = TRUE)
```

Predictions of the random forest model on the validation set:

```
    #assess prediction accuracy on validation data
    rf_predict<-predict(rf_model,newdata=valData)
    rf_cm <- confusionMatrix(factor(rf_predict), factor(valData$classe))
    rf_cm
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 4935  170   12   22    8
##          B   33 3022  160   12   70
##          C   33  206 2833  126   49
##          D   17   10   67 2719   53
##          E    4    9    7   15 3066
## 
## Overall Statistics
## 
##                Accuracy : 0.9387          
##                  95% CI : (0.935, 0.9422)
##     No Information Rate : 0.2844          
##     P-Value [Acc > NIR] : < 2.2e-16       
## 
##                   Kappa : 0.9224          
## 
##  Mcnemar's Test P-Value : < 2.2e-16       
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9827   0.8844   0.9201   0.9395   0.9445
## Specificity            0.9832   0.9807   0.9716   0.9900   0.9976
## Pos Pred Value         0.9588   0.9166   0.8725   0.9487   0.9887
## Neg Pred Value         0.9930   0.9725   0.9829   0.9882   0.9876
## Prevalence             0.2844   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2795   0.1711   0.1604   0.1540   0.1736
## Detection Prevalence   0.2915   0.1867   0.1839   0.1623   0.1756
## Balanced Accuracy      0.9829   0.9325   0.9459   0.9648   0.9711
```

Gradient Boosting Model (GBM)

```
    #train model
    set.seed(12345)
    gbm_control <- trainControl(method = "repeatedcv", number = 5, repeats = 1)
    gbm_model  <- train(classe ~ ., data=trainData, method = "gbm", trControl = gbm_control, ve
rbose = FALSE)
```

Predictions of the GBM model on the validation set:

```
    #assess prediction accuracy on validation data
    gbm_predict <- predict(gbm_model, newdata=valData)
    gbm_cm <- confusionMatrix(factor(gbm_predict), factor(valData$classe))
    gbm_cm
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 4890  161    2   15   21
##          B   56 3014  181    9  111
##          C   27  217 2788  150   43
##          D   37    8   93 2670   61
##          E   12   17   15   50 3010
## 
## Overall Statistics
##                                           
##                Accuracy : 0.9272          
##                  95% CI : (0.9232, 0.931) 
##     No Information Rate : 0.2844          
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.9078          
##                                           
##  Mcnemar's Test P-Value : < 2.2e-16       
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9737   0.8821   0.9055   0.9226   0.9273
## Specificity            0.9843   0.9749   0.9700   0.9865   0.9935
## Pos Pred Value         0.9609   0.8941   0.8645   0.9306   0.9697
## Neg Pred Value         0.9895   0.9718   0.9798   0.9849   0.9838
## Prevalence             0.2844   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2769   0.1707   0.1579   0.1512   0.1705
## Detection Prevalence   0.2882   0.1909   0.1826   0.1625   0.1758
## Balanced Accuracy      0.9790   0.9285   0.9378   0.9546   0.9604
```

Selecting the best prediction model

The random forest prediction model has the highest accuracy, so the actual prediction on the test data is done using this model. The out of sample error - calculated as 1 minus the accuracy - is around 5%. That means on average 1 out of the 20 predictions will be incorrect.

```
    rf_predict_test<-predict(rf_model,testing_raw)
    as.data.frame(rf_predict_test)
```

```
##    rf_predict_test
## 1                C
## 2                A
## 3                B
## 4                A
## 5                A
## 6                E
## 7                D
## 8                D
## 9                A
## 10               A
## 11               B
## 12               C
## 13               B
## 14               A
## 15               E
## 16               E
## 17               A
## 18               B
## 19               B
## 20               B
```