Practical Machine Learning

Mahi Aminu Aliyu

2023-09-10

Human Activity Recognition

The goal of this project is to qunatify how well people perform activity using data from accelerometers on the belt, forearm, arm and dumbbel of 6 participants.
The accelerometers masure the participants perform 5 activities (sitting-down, standing-up, standing, walking, and sitting) as A, B, C, D, and E in the classe variable.

Dataset

The dataset proposed with 5 classe (sitting-down, standing-up, standing, walking, and sitting) collected on 8 hours of activities of 4 healthy subjects. We also established a baseline performance index.

> Ugulino, W.; Cardador, D.; Vega, K.; Velloso, E.; Milidiu, R.; Fuks, H. Wearable Computing: Accelerometers' Data Classification of Body Postures and Movements. Proceedings of 21st Brazilian Symposium on Artificial Intelligence. Advances in Artificial Intelligence - SBIA 2012. In: Lecture Notes in Computer Science. , pp. 52-61. Curitiba, PR: Springer Berlin / Heidelberg, 2012. ISBN 978-3-642-34458-9. DOI: 10.1007/978-3-642-34459-6_6.

Reading dataset

```
if(!file.exists("dataset/pml-training.csv")){
download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv",
              destfile = "dataset/pml-training.csv")
}

if(!file.exists("dataset/pml-testing.csv")){
download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv",
              destfile = "dataset/pml-testing.csv")
}

pml <- read.csv("dataset/pml-training.csv")
```

Data

```
dim(pml)
```

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

The dataset has 19622 rows and 160 variables.

Removing NAs and empty characters

Here we get the percentage of variables that are Na or empty characters and discard them

First thing first we check for NAs in the data set

```
table(colSums(is.na(pml)))
```

```
##
##     0 19216
##    93    67
```

is.na gets the total NAs an non NAs for each column 93 column are non missing values will 67 columns have 19216 missing values suggest

Base on the table we can observe that columns with zero NAs are 93 and Column with 19216 NAs are 67 we can also noticed that there is uniform NAs in each column with NAs, with total rows of 19622 we got 406 NAs That's we have only 2% of non NAs in columns with NAs We will discard the tables because we wont get anything meaningful.

```
table(colSums(pml == ""))
```

```
##
##     0 19216
##    60    33
```

33 columns have 19216 for empty strings only 60 columns

there we discard all the missing columns and empty columns

```
#discard empty strings and NAs

df <-  pml[, colSums(is.na(pml)) == 0]

df <- df[,colSums(df == "") == 0]
```

```
1 - (length(names(df)) / length(names(pml)))
```

```
## [1] 0.625
```

now we can see that 62.5 of the data are either Nas of Empty characters

```
dim(df)
```

```
## [1] 19622    60
```

we are now down to 60 columns

For this prediction task variables such as X, timestamp, window and user_name are irrelevant to our aim, an we will prepare to use data that are closely related to out outcome.

```
df <- df[, -c(which(grepl("^X|timestamp|window|user_name", names(df))))]
dim(df)
```

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

```
# And lastly 53 columns
```

preProcessing

The outcome will be classe (i.e the manner of activities)

```
df$classe <- as.factor(df$classe)
df[, -53] <- lapply(df[, -53], as.numeric)
sam <- sample(nrow(pml), 500)

df <- df[sam ,]
```

Considering the enough data set we have, lets slice the training data set to subTrain and validation set

```
subTrain <- createDataPartition(y = df$classe, p = 0.60, list = F)

subTraining <- df[subTrain, ]
validation <- df[-subTrain, ]
```

#tuning parameteres

```
fitControl <- trainControl(
  method = "repeatedcv",
  number = 10,
  repeats = 2,
  classProbs = T,
  verboseIter = F
)
```

Training

we are going to use three classification algorithms for namely: 1. Stochastic Gradient Boosting Machine 2. Random Forest, and 3. Support Vector Machine

```
modFitGBM <- train(classe ~ ., method = "gbm", data = subTraining,
                   trControl = fitControl,
                   verbose = F
)
modFitFor <- train(classe ~., method = "rf", data = subTraining,
                   trContol = fitControl
)

modFitSVM <- train(classe ~., method = "svmLinear2", data = subTraining,
                  trControl = fitControl
)
```

Prediction

Using the validation dataset will predict and evaluate our algorithm

```
preLogit <- predict(modFitGBM, newdata = validation)
preFor <- predict(modFitFor, newdata = validation)
preSVM <- predict(modFitSVM,newdata = validation)
```

Confusion Matrix

```
confusionMatrix(preLogit, validation$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  A  B  C  D  E
##          A 49 12  1  1  1
##          B  2 20  1  4  6
##          C  1 11 26  2  2
##          D  1  3  5 19  1
##          E  1  0  1  3 24
##
## Overall Statistics
##
##                Accuracy : 0.7005
##                  95% CI : (0.6313, 0.7635)
##     No Information Rate : 0.2741
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6199
##
```

```
##  Mcnemar's Test P-Value : 0.006993
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9074   0.4348   0.7647  0.65517   0.7059
## Specificity            0.8951   0.9139   0.9018  0.94048   0.9693
## Pos Pred Value         0.7656   0.6061   0.6190  0.65517   0.8276
## Neg Pred Value         0.9624   0.8415   0.9484  0.94048   0.9405
## Prevalence             0.2741   0.2335   0.1726  0.14721   0.1726
## Detection Rate         0.2487   0.1015   0.1320  0.09645   0.1218
## Detection Prevalence   0.3249   0.1675   0.2132  0.14721   0.1472
## Balanced Accuracy      0.9013   0.6743   0.8333  0.79782   0.8376
```

```
confusionMatrix(preFor, validation$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  A  B  C  D  E
##          A 49 13  2  1  0
##          B  4 24  3  3 11
##          C  1  6 27  3  3
##          D  0  3  2 20  1
##          E  0  0  0  2 19
##
## Overall Statistics
##
##                Accuracy : 0.7056
##                  95% CI : (0.6366, 0.7682)
##     No Information Rate : 0.2741
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6242
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9074   0.5217   0.7941   0.6897  0.55882
## Specificity            0.8881   0.8609   0.9202   0.9643  0.98773
## Pos Pred Value         0.7538   0.5333   0.6750   0.7692  0.90476
## Neg Pred Value         0.9621   0.8553   0.9554   0.9474  0.91477
## Prevalence             0.2741   0.2335   0.1726   0.1472  0.17259
## Detection Rate         0.2487   0.1218   0.1371   0.1015  0.09645
## Detection Prevalence   0.3299   0.2284   0.2030   0.1320  0.10660
## Balanced Accuracy      0.8978   0.6913   0.8572   0.8270  0.77328
```

```
confusionMatrix(preSVM, validation$classe)
```

```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction  A  B  C  D  E
##          A 46 16  9  3  2
##          B  5 13  0  2  5
##          C  1  6 21  2  7
##          D  1  7  4 20  3
##          E  1  4  0  2 17
##
## Overall Statistics
##
##                Accuracy : 0.5939
##                  95% CI : (0.5218, 0.6631)
##     No Information Rate : 0.2741
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4827
##
##  Mcnemar's Test P-Value : 0.0007793
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8519  0.28261   0.6176   0.6897  0.50000
## Specificity            0.7902  0.92053   0.9018   0.9107  0.95706
## Pos Pred Value         0.6053  0.52000   0.5676   0.5714  0.70833
## Neg Pred Value         0.9339  0.80814   0.9188   0.9444  0.90173
## Prevalence             0.2741  0.23350   0.1726   0.1472  0.17259
## Detection Rate         0.2335  0.06599   0.1066   0.1015  0.08629
## Detection Prevalence   0.3858  0.12690   0.1878   0.1777  0.12183
## Balanced Accuracy      0.8210  0.60157   0.7597   0.8002  0.72853
```