Practical Machine Learning

Shivangi

9/22/2020

SYNOPSIS

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: http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

DATA

The training data for this project are available here:

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

The test data are available here:

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

The data for this project come from this source: http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har. If you use the document you create for this class for any purpose please cite them as they have been very generous in allowing their data to be used for this kind of assignment.

DOWNLOAD DATA AND LOAD LIBRARIES

```r
library(caret)
library(rattle)
```

Loading the test data and train data.

```r
train_data <- read.csv("train.csv")
test_data <- read.csv("test.csv")
```

DATA CLEANSING

REMOVING NA VALUES

Removing the columns i.e. predictors having near zero values.

```
nzv <- nearZeroVar(train_data)
train_data <- train_data[,-nzv]
test_data <- test_data[,-nzv]
```

Removing NA values from the datasets.

```
na_val_col <- sapply(train_data, function(x) mean(is.na(x))) > 0.95
train_data <- train_data[,na_val_col == FALSE]
test_data <- test_data[,na_val_col == FALSE]

dim(train_data)
```

```
## [1] 19622    59
```

Removing first 7 variables because they are non numeric.

```
train_data <- train_data[,-(1:7)]
test_data <- test_data[,-(1:7)]
```

PARTITIONING THE DATA

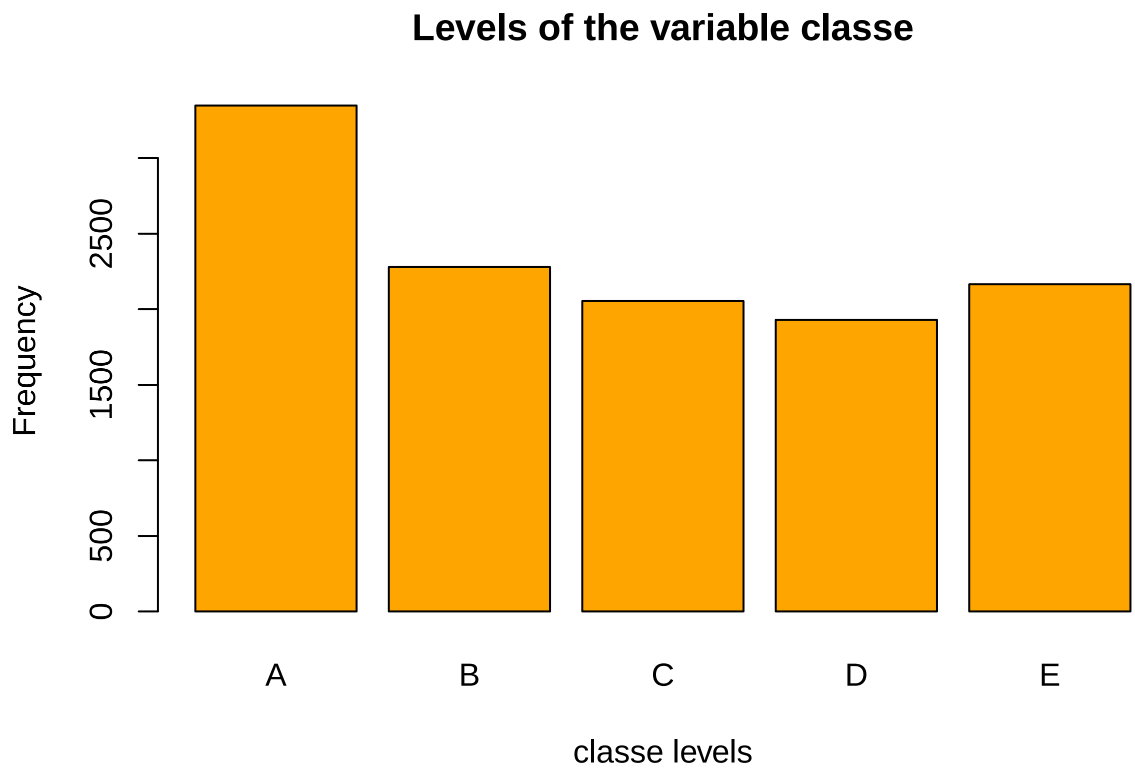
We split the train data into two dataset training set and testing set having 60% and 40% of the original window respectively.

```
inTrain<- createDataPartition(train_data$classe, p=0.6, list=FALSE)
training<- train_data[inTrain,]
testing<- train_data[-inTrain,]
```

VISUALIZING DATA

The plot is shown as follows

```
plot(as.factor(training$classe), col="orange", main="Levels of the variable classe", xlab="classe levels
     ylab="Frequency")
```

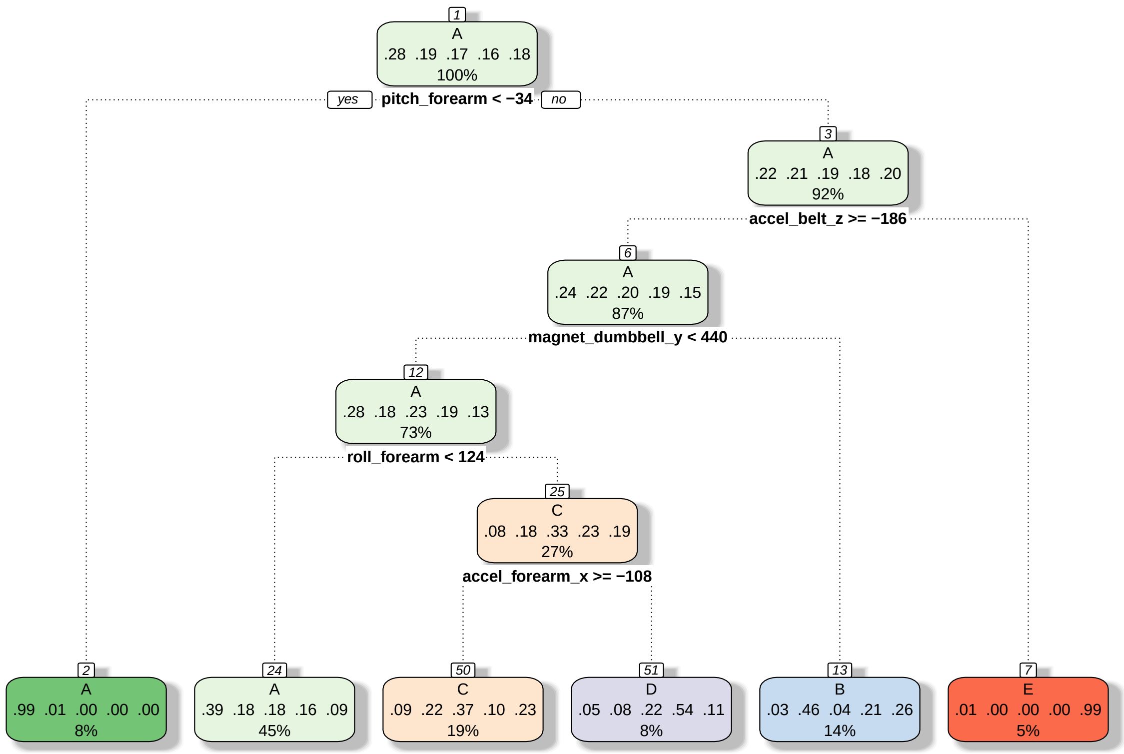


TRAINING MODELS AND PREDICTING MODELS

PREDICTING USING TRESS

Training the model

```
library(rattle)
model_t <- train(classe~., data=training, method="rpart")
fancyRpartPlot(model_t$finalModel)
```



Rattle 2020−Sep−22 23:14:13 comp

Predicting the model

```
model_p <- predict(model_t, testing)
confusionMatrix(model_p, as.factor(testing$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2021  633  647  588  341
##          B   37  524   38  219  290
##          C  125  309  542  177  335
##          D   39   52  141  302   61
##          E   10    0    0    0  415
##
## Overall Statistics
##
##                Accuracy : 0.4848
##                  95% CI : (0.4737, 0.496)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3256
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
```

```
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9055  0.34519  0.39620  0.23484  0.28779
## Specificity            0.6065  0.90771  0.85397  0.95534  0.99844
## Pos Pred Value         0.4778  0.47292  0.36425  0.50756  0.97647
## Neg Pred Value         0.9416  0.85248  0.87008  0.86429  0.86161
## Prevalence             0.2845  0.19347  0.17436  0.16391  0.18379
## Detection Rate         0.2576  0.06679  0.06908  0.03849  0.05289
## Detection Prevalence   0.5391  0.14122  0.18965  0.07583  0.05417
## Balanced Accuracy      0.7560  0.62645  0.62508  0.59509  0.64312
```

The accuracy of this method is: 0.5297

CONCLUSION

Here, we elected decision tree model, though the accuracy is not high but model works fine. ### FINAL PREDICTION

```r
Final_prediction <- predict(model_t, test_data )
Final_prediction
```

```
##  [1] D A C A A C D A A A C C C A C A E A A C
## Levels: A B C D E
```