Supervised Learning Model

Human Activity Recognition

AUTHOR
Phong Trang Tran Thanh

PUBLISHED
February 11, 2023

Overview

Data description

This project examine the Weight Lifting Exercise Data provided by Velloso et al. (2013).They used the wearable devices to measure the acceleration of 6 participants to see if the participants were doing the exercises correctly. The accelerometers on the belt, forearm, arm, and dumbbell of 6 participants provide information on the exercise movements. There are 4 classes of the labels: A, B, C, and D; only class A is the correct movement while other classes corresponds to common mistakes of the training. Detailed description of the data can be found here.

Basically the data contain a list of variables provided by accelerometers on X, Y and Z dimension while the "class" variable tells if a given observation is of correct or incorrect training. The training data for this project are available here: pml-training.csv The test data are available here: pml-testing.csv

The project goal

This project aims to use data provided by the accelerators to build a supervised learning model that predicts the outcome of correct/incorrect training movements. There are two models built to validate the prediction outcome: 1. Random Forest model 2. Support Vector machine (SVM) The below sections include:

- Model construction
- Cross validation
- Test the model with 20 testing samples
  # Data Preprocessing

```
#Load the needed library
library(ggplot2)
library(caret)
library(dplyr)
library(rpart)
```

```
library(randomForest)
library(kernlab)
library(kableExtra)
library(e1071)
library(data.table)
```

```
#Download and load the data
url1 = "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.c
url2 = 'http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv
filename1 = "pml-training.csv"
filename2 = "pml-testing.csv"
download.file(url1, filename1)
download.file(url2, filename2)
training = read.csv("pml-training.csv", na.strings = c("", "NA", "#DIV/0!"
testing = read.csv("pml-testing.csv"
                   , na.strings = c("", "NA", "#DIV/0!"))
```

```
training_data<- dim(training)
testing_data <- dim(testing)
dim_table <- rbind(training_data, testing_data)
colnames(dim_table) <- c("Rows", "Columns")
knitr::kable(dim_table)
```

Table 1: dimension of the training and testing dataset

	Rows	Columns
training_data	19622	160
testing_data	20	160

```
par(bg = "#EEECEE")
barplot(table(training$classe), col = c("#E0BBE4", "#957DAD", "#D291BC", "
```

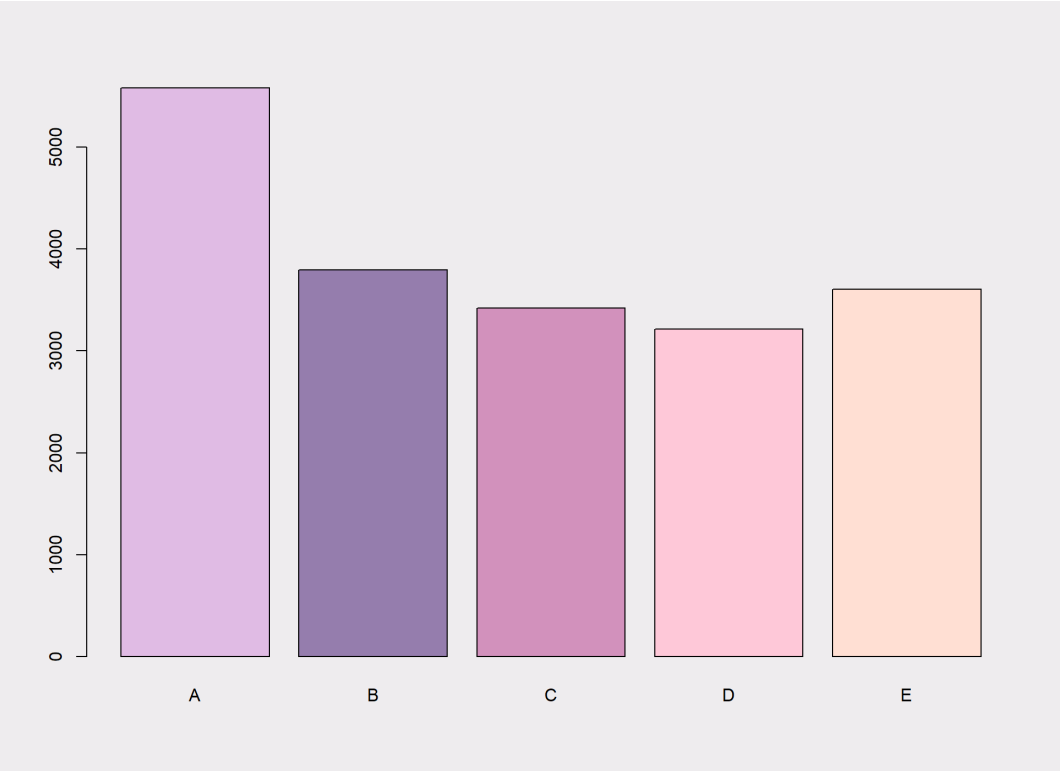


Figure 1: Number of observations in each classe

The data set is skewed toward A class and this may lead the model to be biased toward A. However, according to the data code book, this difference does not affect the data analysis due to its consistency of the rest of the labels.

```
#Split the data into train set and validation set
set.seed(123456)
inTrain = createDataPartition(training$classe, p = 0.8, list = FALSE)
Train = training[inTrain, ]
Validation = training[-inTrain, ]
dim(Train)
```

```
[1] 15699   160
```

Feature selection for modeling

This section checks if a predictor has little or no impact on the response variable: - First, Near Zero Variance technique is used to remove little-to-no impact variables. Then, descriptive columns, such as column names, ID numbers, or other meta-data, may not be directly relevant to the analysis and can be excluded as well. According to the data codebook, these descriptive variables are `X, user_name, raw_timestamp_part_1, raw_timestamp_part_2, cvtd_timestamp`.

```
#Exclude the descriptive predictors
descriptcol <- c("X", "user_name", "raw_timestamp_part_1", "raw_timestamp_
    "cvtd_timestamp", "new_window", "num_window")
Train = Train[,!names(Train) %in% descriptcol]
Validation = Validation[,!names(Validation) %in% descriptcol]
# Check for near zero variance predictors and drop them if necessary
```

```
nonzerocol = nearZeroVar(Train)
Train = Train[, -nonzerocol]
Validation=Validation[,-nonzerocol]
```

- Secondly, there are some measured statistics which are the same for all rows. These numbers need to be removed also.

```
# Grab the pattern of unneeded columns
unneededColPattern = "kurtosis_|skewness_|max_|min_|amplitude_|avg_|stddev
# Removed the columns containing the patterns
excludePattern <- function (tdata, unneededColPattern) {
  exdata <- tdata[, -grep(unneededColPattern, colnames(tdata))]
  exdata
}
Train = excludePattern(Train, unneededColPattern)
Validation = excludePattern(Validation, unneededColPattern)
```

- Finally, we make sure that there is no column that contains more than 50% percent NA over its total length

```
countlength = sapply(Train, function(x) {
    sum(!(is.na(x) | x == ""))
})
nullCol = names(countlength[countlength < 0.5 * length(Train$classe)])
Train = Train[, !names(Train) %in% nullCol]
Validation= Validation[, !names(Validation) %in% nullCol]
```

```
dim_table= data.frame(rows=dim(Train)[1],columns=dim(Train)[2])
knitr::kable(dim_table)
```

Table 2: dimension of the training data after Feature Selection

rows	columns
15699	53

After feature selection, the columns has reduced its size to 53.

#Model Training and Validation

Training the model using Random Forest

```
rfModel <- randomForest(as.factor(classe)~., data=Train)
# Summary of the model
```

```
rfModel
```

```
Call:
 randomForest(formula = as.factor(classe) ~ ., data = Train)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 7

        OOB estimate of  error rate: 0.38%
Confusion matrix:
     A    B    C    D    E  class.error
A 4460    2    0    0    2 0.0008960573
B   11 3024    3    0    0 0.0046082949
C    0   11 2724    3    0 0.0051132213
D    0    0   20 2551    2 0.0085503304
E    0    0    0    5 2881 0.0017325017
```

Checking the error rate for number of trees

```
oobData = as.data.table(plot(rfModel))
oobData[, trees := .I]
oobData2 = melt(oobData, id.vars = "trees")
setnames(oobData2, "value", "error")
ggplot(data = oobData2, aes(x = trees, y = error, color = variable)) + geo
```

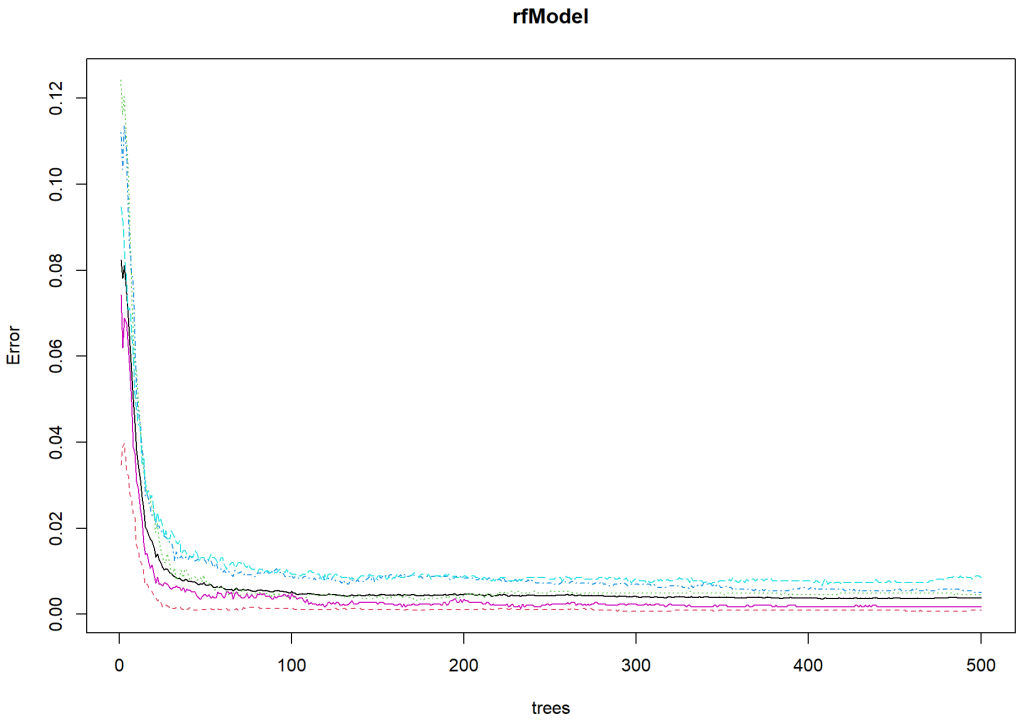


Figure 2: Number of trees vs Error plot

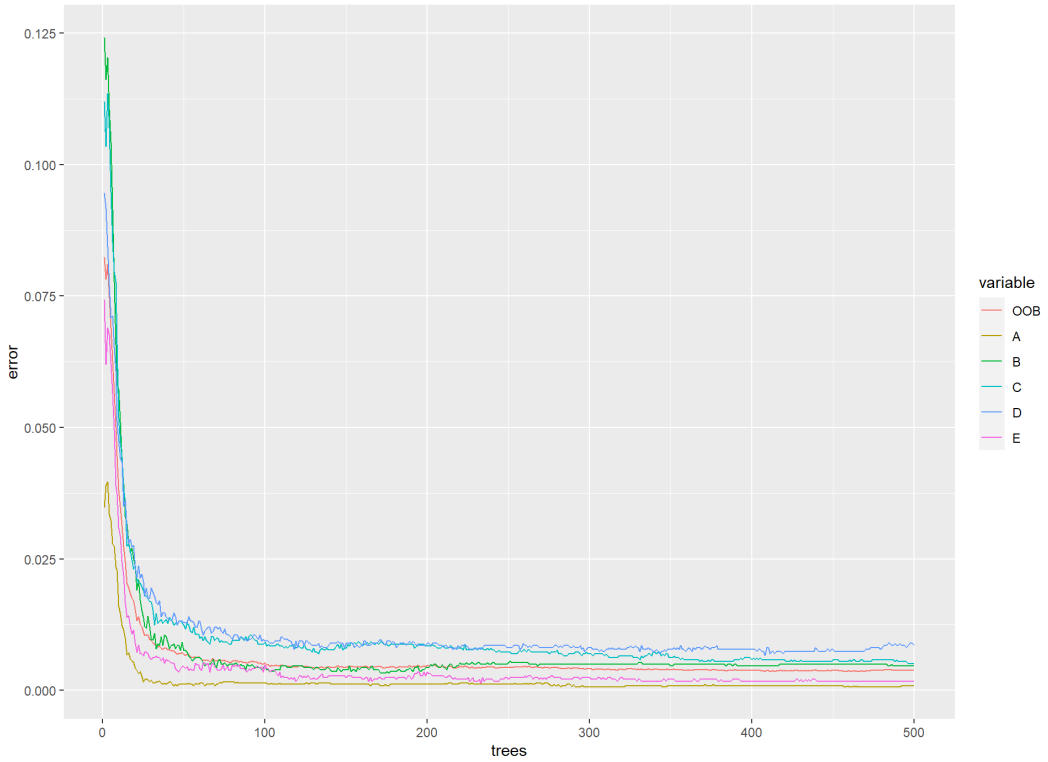


Figure 3: Number of trees vs Error plot

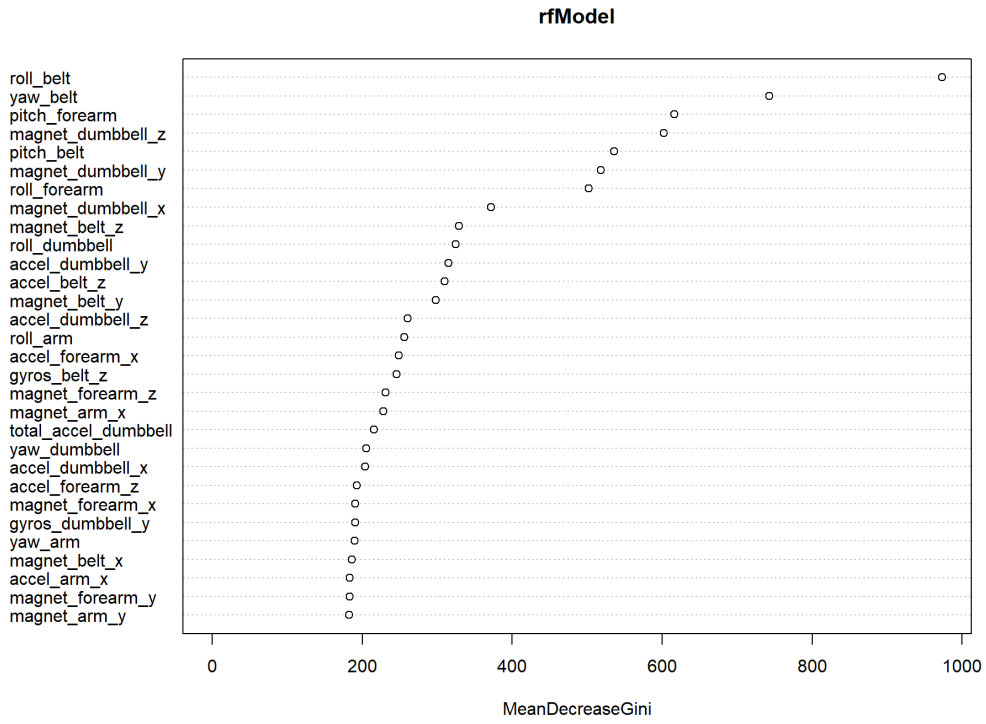
- As **?@fig-tree-err** shows that the optimal number of trees are 200, now the model is retrained with **200 trees**

```
rfModel <- randomForest(as.factor(classe)~., data=Train, ntree =200)
# Summary of the new model
rfModel
```

```
Call:
 randomForest(formula = as.factor(classe) ~ ., data = Train, ntree = 200)
               Type of random forest: classification
                     Number of trees: 200
No. of variables tried at each split: 7

        OOB estimate of  error rate: 0.41%
Confusion matrix:
     A    B    C    D    E  class.error
A 4460    2    0    1    1 0.0008960573
B   10 3024    4    0    0 0.0046082949
C    0   11 2724    3    0 0.0051132213
D    0    0   24 2547    2 0.0101049359
E    0    0    2    5 2879 0.0024255024
```

```
varImpPlot(rfModel)
```



Validate the model

```
predictions = predict(rfModel, newdata = Validation)
#Ensure the same level
Validation$classe = factor(Validation$classe, levels = levels(predictions)
confusion_matrix = confusionMatrix(predictions,Validation$classe)
model_table <- data.frame(
  Model = "Random Forest",
  `Number of Trees` = rfModel$ntree,
  `Out-of-Bag Error` = rfModel$err.rate[rfModel$ntree],
  `Accuracy` = confusion_matrix$overall[1],
  `Kappa` = confusion_matrix$overall[2]
)
kable(model_table, caption = "Random Forest Model Summary", row.names = FA
```

Table 3: Summary of the model validation

Model	Number.of.Trees	Out.of.Bag.Error	Accuracy	Kappa
Random Forest	200	0.0041404	0.9974509	0.9967756

```
#Plot the confusion matrix
confusion_df = as.data.frame(confusion_matrix$table)
ggplot(confusion_df, aes(x = Reference, y = Prediction, fill = Freq)) +
  geom_tile(color = "#EEECEE") +
  scale_fill_gradient(low = "white", high = "#D291BC") +
  geom_text(aes(label = Freq), color = "black", size = 6) +
  labs(title = paste("Confusion Matrix Random Forest: Accuracy =", round(c
  xlab("Reference") +
```

```
  ylab("Prediction") +
  theme_minimal()
```

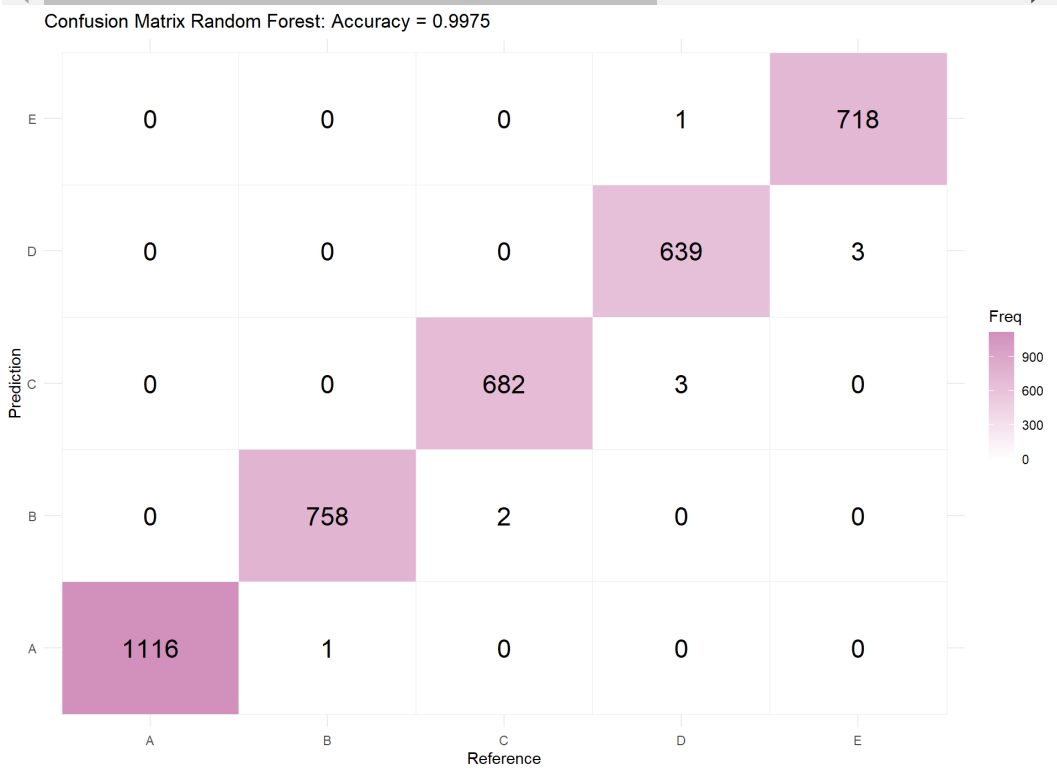


Figure 4: **?(caption)**

Training the model using SVM

```
svmModel = svm(as.factor(classe) ~. , data=Train)
#prediction
svmPredictions <- predict(svmModel, newdata= Validation)
# Confusion matrix
cmSVM <- confusionMatrix(svmPredictions, Validation$classe)
```

```
#Plot the confusion matrix
confusion_df2 = as.data.frame(cmSVM$table)
ggplot(confusion_df2, aes(x = Reference, y = Prediction, fill = Freq)) +
  geom_tile(color = "#EEECEE") +
  scale_fill_gradient(low = "white", high = "#957DAD") +
  geom_text(aes(label = Freq), color = "black", size = 6) +
  labs(title = paste("Confusion Matrix: Accuracy =", round(confusion_matri
  xlab("Reference") +
  ylab("Prediction") +
  theme_minimal()
```

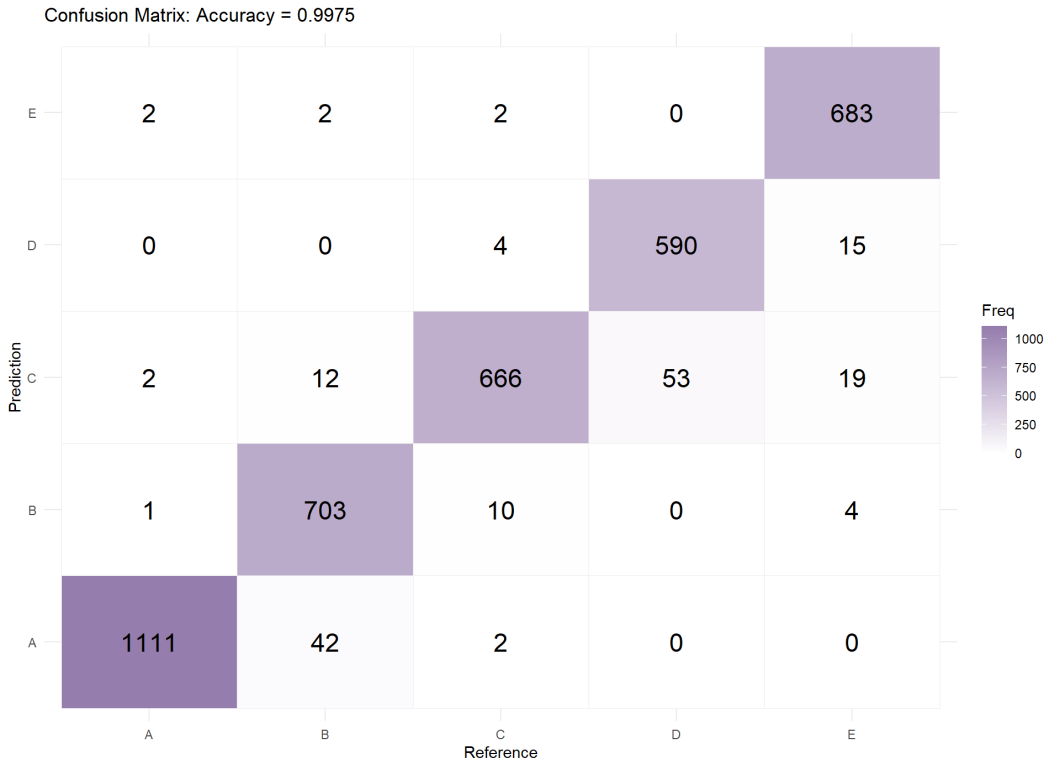


Figure 5: **?(caption)**

Predicting the result on the Test data

Prediction made by the Random Forest Model

```
#Select the columns that was used to train the model, except classe be  us
namecol=colnames(Train)[!colnames(Train) %in% c("classe")]
testing2=testing[,namecol]
rfPredictions <- predict(rfModel, newdata = testing2)
rfPredictions
```

```
 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
 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
```

Prediction made by the SVM Model

```
svmPrediction <- predict(svmModel, newdata = testing2)
svmPrediction
```

```
 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
 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
```