Manners Prediction

ThetLwinLwin

1/8/2021

Introduction

As the emerge of the hand wear bands or devices, it is now possible to collect large amount of data about personal activity. A group of enthusiasts took measurements about themselves regularly to improve their health and pattern of behavior. The main goal of this project to predict the manner in which they did the exercise using collected data from accelerometer.

Exploratory Analysis

Download and read Data

```
if(!file.exists('./data/pml-training.csv') && !file.exists('./data/pml-testing.csv')){
  download.file(url='https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv',destfile = './data/pml-training.csv')
  download.file(url='https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv',destfile = './data/pml-testing.csv')
  training = read.csv('./data/pml-training.csv')
  testing = read.csv('./data/pml-testing.csv')
}else{
  training = read.csv('./data/pml-training.csv')
  testing = read.csv('./data/pml-testing.csv')
}
```

Data Structure

The data has 160 variables and total observations of 19622. Missing values are also important in data analysis. It can be dealt with either removing or imputing the value. The detail of each variable can be read in this link.

```
require(caret)
require(rpart)
require(rattle)
Nan_value <- sapply(training, function(x) mean(is.na(x)))
#90 percent of observations is missing value.
table(Nan_value > 0.9)
```

```
##
## FALSE  TRUE
##    93    67
```

There are 67 variables with missing values. Instead of removing rows with NA values, these variables will be removed. In fact, these 67 variables has almost 97 percentage of missing values to total ones.

Models

The goal of this anaylsis is to predict the 'classe' object from trained model.

```
table(training$classe)
```

```
##
##    A    B    C    D    E
## 5580 3797 3422 3216 3607
```

Data Wrangling

There are 5 class to be predicted. This is the classification problem. First, the columns with mostly missing values are removed.

```
indToRmv <- colSums(is.na(training))
filteredtraining <- training[,indToRmv==0]
filteredtesting <- testing[,indToRmv==0]
```

Next, near zero variance will be checked and removed.

```
nzv <- nearZeroVar(filteredtraining)
finalTrain <- filteredtraining[,-nzv]
finalTest <- filteredtesting[,-nzv]
dim(finalTrain)
```

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

Near Zero Variance reduces feature from 93 to 59.

Data Partition

The data are now split into **train** and **test** dataset.

```
inTrain <- createDataPartition(y=finalTrain$classe, p =0.75,list=FALSE)
train <- finalTrain[inTrain,]
test <- finalTrain[-inTrain,]
```

Model_1

The first go-to model for classification problem is decision tree.

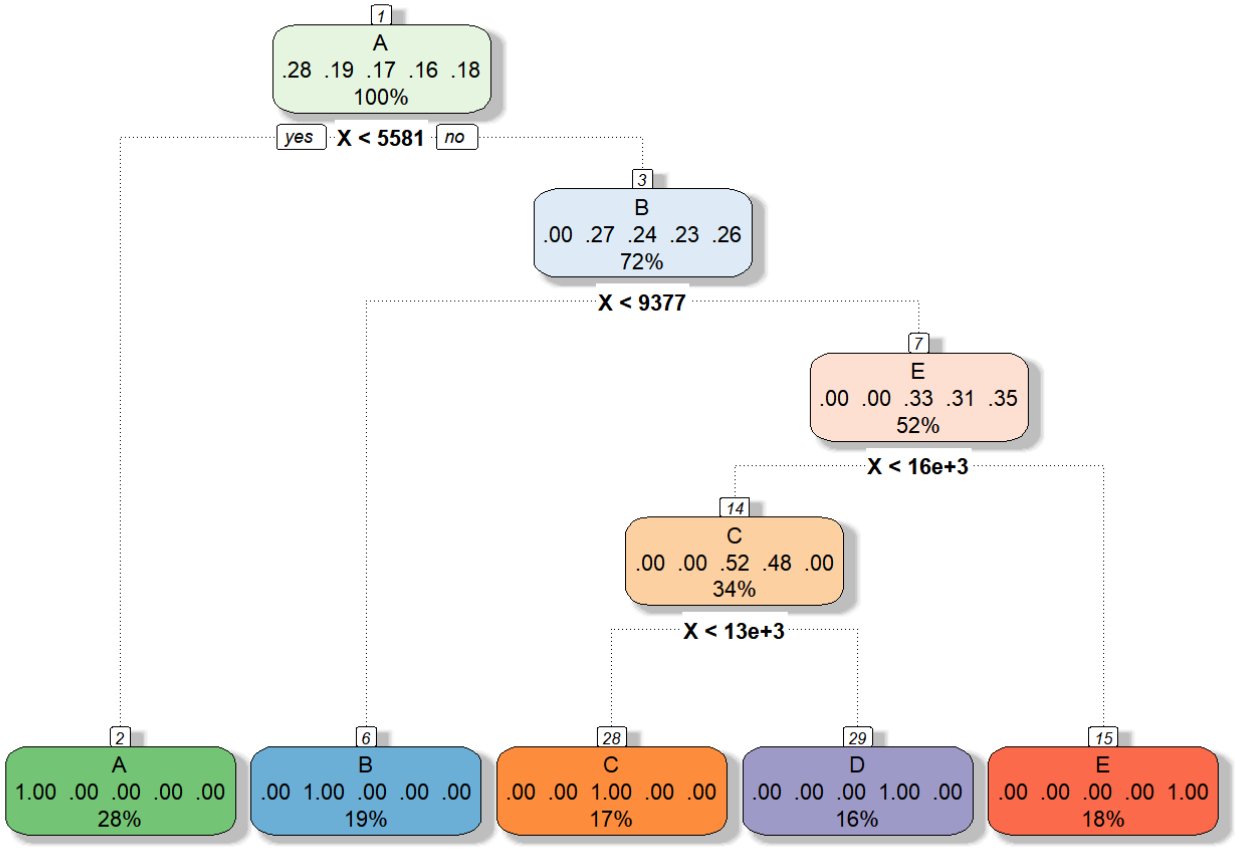
```
decisionTree <- rpart(classe~.,data=train, method = 'class')
decisionTreePred <- predict(decisionTree,newdata = test,type = 'class')

#convert outcome to be factor variable.
test$classe = as.factor(test$classe)
confusion1<- confusionMatrix(decisionTreePred,test$classe)
print(confusion1$table)
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    0    0    0    0
##          B    0  948    0    0    0
##          C    0    1  855    0    0
```

```
##          D    0    0    0  804    0
##          E    0    0    0    0  901
```

Basically, this model can predict almost correct. It may be due to several reasons. The first reason is overfitting. The second one is that the outcome is somehow included in features. To make it clear, tree was plotted.

```
fancyRpartPlot(decisionTree)
```



The tree graph indicates that prediction is governed by X. So, the features are needed to be filtered and model is rebuilt.

```
train <- train[,-(1:5)]
test <- test[,-(1:5)]
testing <- testing[,-(1:5)]
decisionTree <- rpart(classe~.,data=train, method = 'class')
decisionTreePred <- predict(decisionTree,newdata = test,type = 'class')

confusion1<- confusionMatrix(decisionTreePred,test$classe)
print(paste0('Accuracy is ',round(confusion1$overall['Accuracy'],3)))
```

```
## [1] "Accuracy is 0.737"
```

The Accuracy is far less than the previous decision tree model.

Model_2

Random Forest is a flexible, easy to use machine learning algorithm that produces even without hyper-parameter tuning, a great result most of the time.

```
trControl <- trainControl(method='repeatedcv',
                          number = 5,
                          repeats = 2,
                          classProbs=TRUE)
randForest <- train(classe~.,
                    data=train,
                    method='rf',
                    trControl=trControl,
                    importance=TRUE
                    )
randForestPred <- predict(randForest,newdata = test)
confusion2 <- confusionMatrix(randForestPred,test$classe)
confusion2$table
```

```
##          Reference
## Prediction    A    B    C    D    E
```

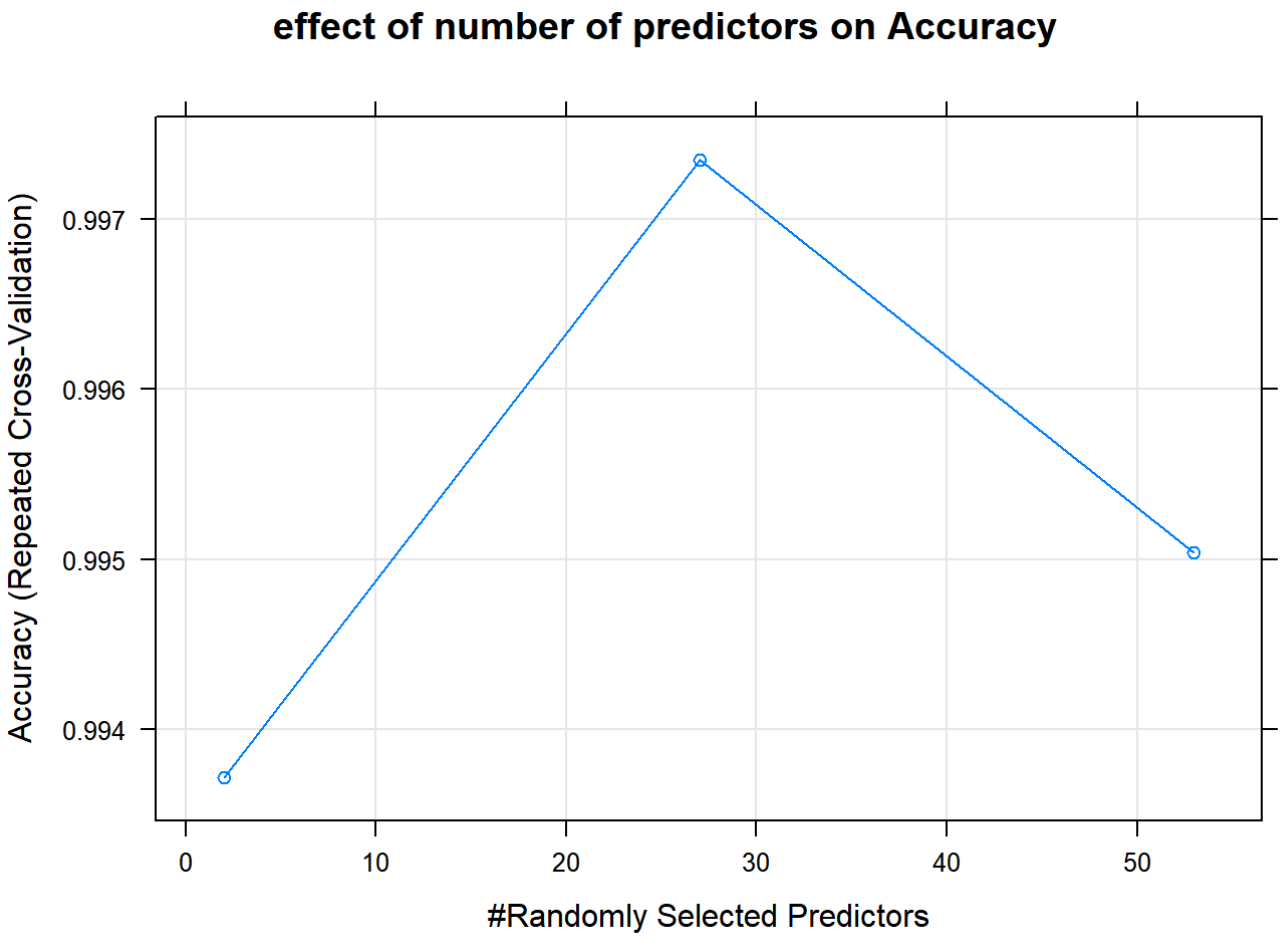
```
##          A 1395    2    0    0    0
##          B    0  947    2    0    0
##          C    0    0  853    4    0
##          D    0    0    0  800    0
##          E    0    0    0    0  901
```

```
print(paste0('Accuracy is ',round(confusion2$overall['Accuracy'],3)))
```

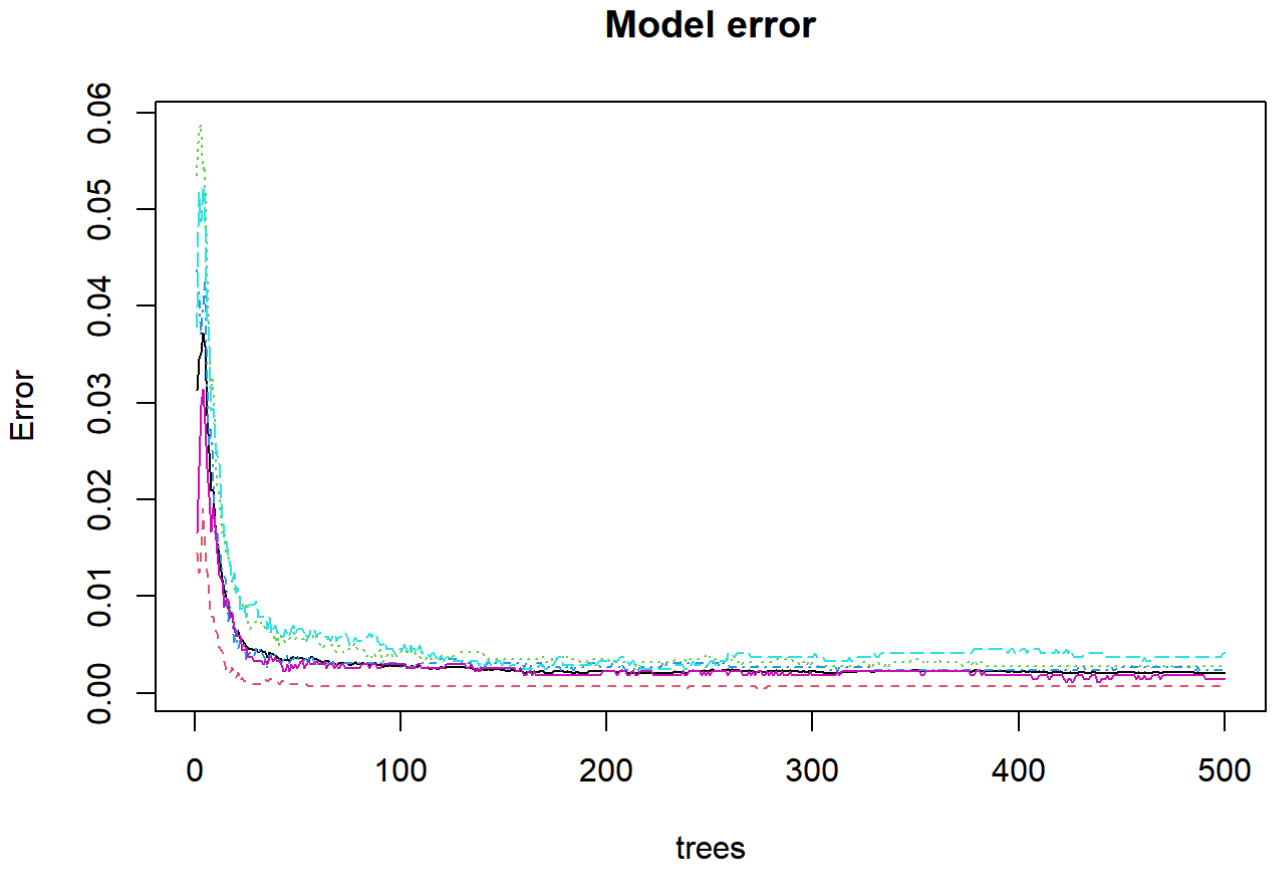
```
## [1] "Accuracy is 0.998"
```

Again, the accuracy is nearly perfect. There is a chance that this is due to overfitting the model. When validating with testing data set, the accuracy will be less than this above result.

```
plot(randForest, main='effect of number of predictors on Accuracy')
```




```
plot(randForest$finalModel, main = 'Model error')
```



This model is pretty reasonable as far as we concerned. Model accuracy is excellent and use less features. In real life, training other models is suggested.

Model Decision

Model will be decided based on the testing data set as we loaded at the beginning.

```
#convert testing data set outcome.
testing$classe <- as.factor(testing$classe)
```

```
model1Pred <- predict(decisionTree,newdata = testing,type='class')
model1ConfuMat <- confusionMatrix(model1Pred,testing$classe)

model2Pred <- predict(randForest,newdata = testing)
model2ConfuMat <- confusionMatrix(model2Pred,testing$classe)

model2ConfuMat$table
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 5580    2    0    0    0
##          B    0 3795    2    0    0
##          C    0    0 3420    3    0
##          D    0    0    0 3213    1
##          E    0    0    0    0 3606
```

```
print(paste0('The accuracy of Decision Tree Model is ',
             model1ConfuMat$overall['Accuracy']))
```

```
## [1] "The accuracy of Decision Tree Model is 0.740903067984915"
```

```
print(paste0('The accuracy of Random Forest Model is ',
             model2ConfuMat$overall['Accuracy']))
```

```
## [1] "The accuracy of Random Forest Model is 0.99959229436347"
```

The aim of this project is to build a accurate prediction model on common incorrect gestures during barbell lifts based on several variables collected by accelerometers. For Random Forest model, the accuracy increases in validation dataset. This model perform best for this classification. Other classification models with boosting or bagging will likely be able to achieve high results.