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-testing.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]
dim(finalTrain)
```

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

Near Zero Variance reduces feature from 93 to 59.

Data Partition

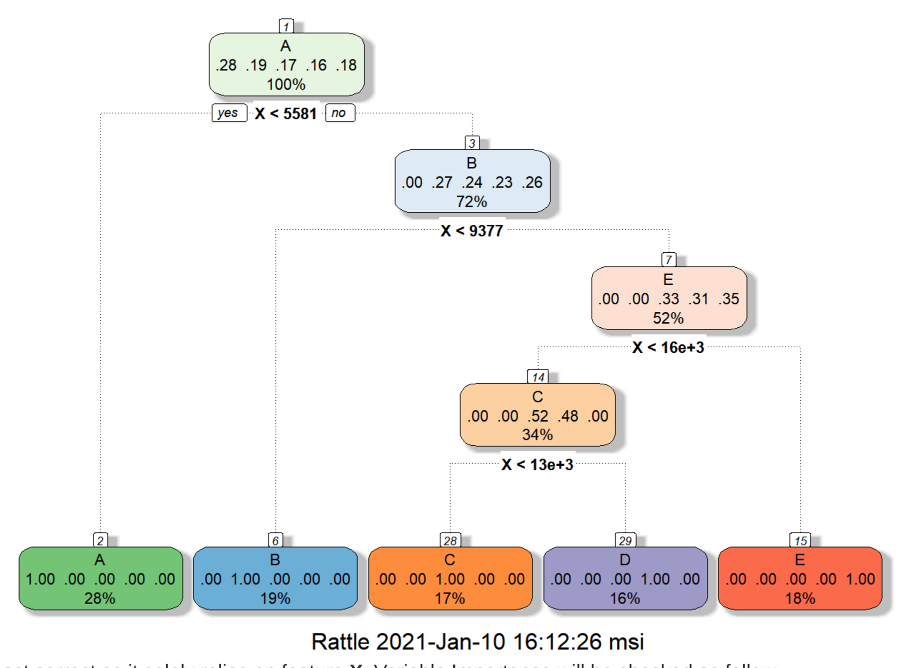
The data are now split into **train** and **test** dataset.

```
inTrain <- createDataPartition(y=finalTrain$classe, p =0.75,list=FALSE)
train <- finalTrain[inTrain,]
test <- finalTrain[-inTrain,]
set.seed(8834)
```

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')
fancyRpartPlot(decisionTree)
```



Basically, this model can predict almost correct as it solely relies on feature **X**. Variable Importance will be checked as follow.

```
decisionTree$variable.importance
```

```
##                    X       cvtd_timestamp            roll_belt
##           11636.3994            5650.4891            1283.2277
##        pitch_forearm       pitch_dumbbell raw_timestamp_part_1
##            1092.1972            1030.8593            1015.4396
##        roll_dumbbell     magnet_dumbbell_y         accel_belt_z
##             965.1384             900.1771             863.2623
##     accel_dumbbell_x        magnet_belt_y        magnet_belt_z
##             833.1536             823.4044             794.2402
##              yaw_arm          accel_arm_x           pitch_belt
##             542.5176             496.8602             384.4733
##    magnet_dumbbell_z
##             184.9365
```

So, the features are needed to be filtered and model is rebuilt.

```
train <- train[,-(1:5)]
test <- test[,-(1:5)]
decisionTree <- rpart(classe~.,data=train, method = 'class')
decisionTree$variable.importance
```

```
##            roll_belt           num_window        pitch_forearm
##           1868.27522           1202.81696            920.31111
##           pitch_belt         accel_belt_z    magnet_dumbbell_y
##            905.90386            839.25510            761.42236
##     accel_dumbbell_y total_accel_dumbbell     total_accel_belt
##            686.99714            671.39079            630.54630
##        roll_dumbbell         roll_forearm      accel_forearm_x
##            612.02537            550.09554            488.59798
##             yaw_belt         accel_belt_y    magnet_dumbbell_z
##            478.64132            418.64932            393.36468
##     accel_dumbbell_x        magnet_belt_z        magnet_belt_x
##            373.42017            368.97741            366.00917
##         accel_belt_x        magnet_belt_y    magnet_dumbbell_x
##            294.85016            262.78224            253.27746
##     accel_dumbbell_z         yaw_dumbbell              yaw_arm
##            241.70274            224.18439            223.75430
##     magnet_forearm_z     magnet_forearm_x     magnet_forearm_y
##            211.84316            207.30609            192.31783
##          yaw_forearm     gyros_dumbbell_x      accel_forearm_z
##            156.06019            106.09856             92.18627
##             roll_arm  total_accel_forearm            pitch_arm
##             84.03306             53.59033             52.89868
##          gyros_arm_x       pitch_dumbbell          gyros_arm_y
##             51.42232             45.77427             41.74882
##         gyros_belt_y     gyros_dumbbell_y         magnet_arm_x
##             38.50990             18.56340             17.56781
##         magnet_arm_z
##              5.31408
```

Now, there is no overwhelming features in model. The accuracy will be checked in model decision.

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.

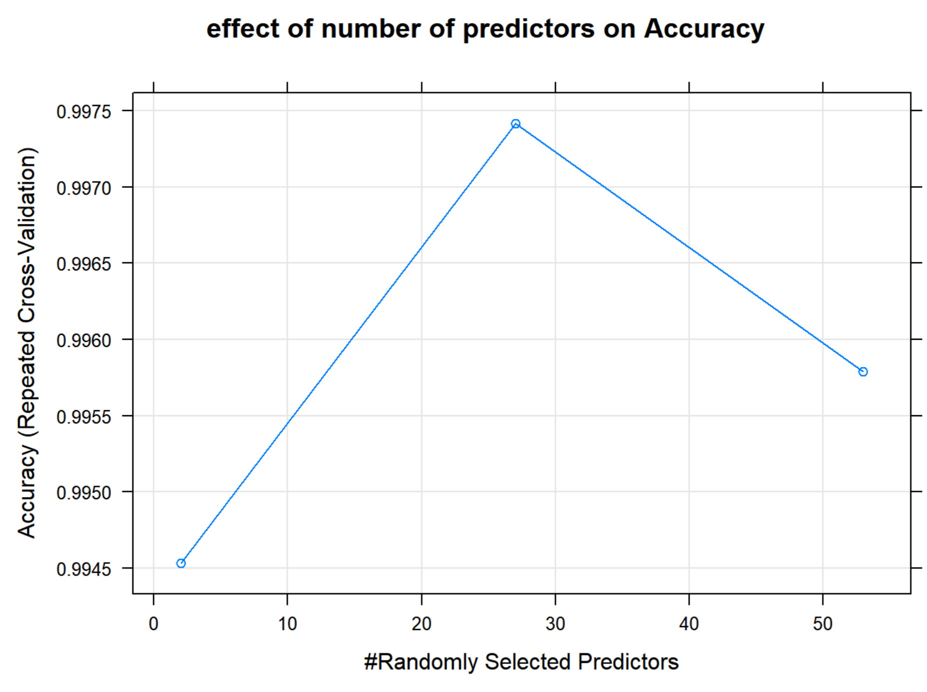
```
trControl <- trainControl(method='repeatedcv',
                          number = 5,
                          repeats = 2,
                          classProbs=TRUE)
randForest <- train(classe~.,
                    data=train,
                    method='rf',
                    trControl=trControl,
                    importance=TRUE
                    )

randForest
```

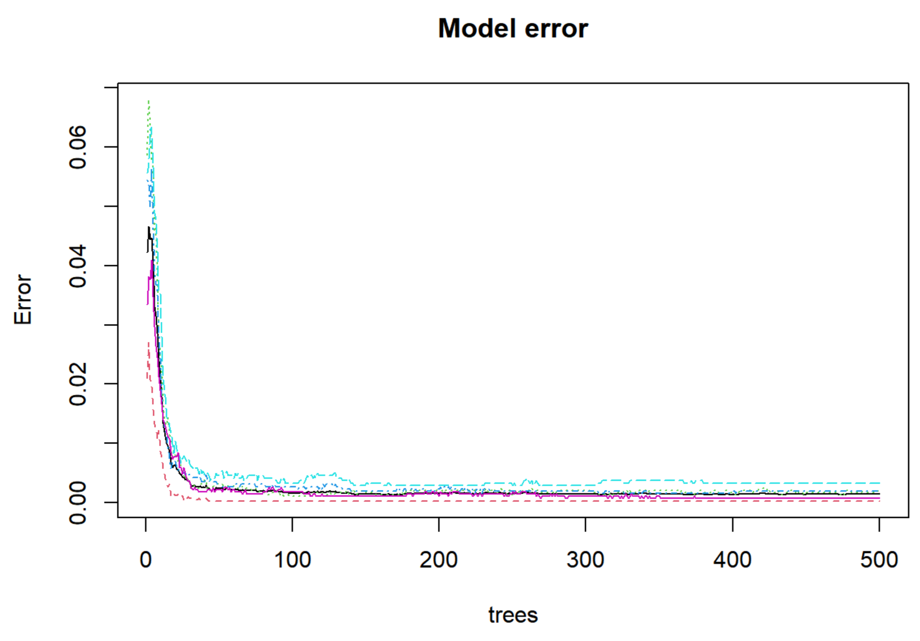
```
## Random Forest
##
## 14718 samples
##    53 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 2 times)
## Summary of sample sizes: 11774, 11774, 11774, 11775, 11775, 11774, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9945305  0.9930809
##   27    0.9974181  0.9967340
##   53    0.9957875  0.9946714
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 27.
```

Again, the accuracy is nearly perfect. There is a chance that this is due to overfitting the model. The clear result will be seen after confusion matrix is built in model decision.

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




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



Model Decision

Model will be decided based on the test dataset.

```
#Decision Tree
test$classe <- as.factor(test$classe)
decisionTreePred <- predict(decisionTree,newdata = test,type = 'class')
confusion1 <- confusionMatrix(decisionTreePred,test$classe)
confusion1$table
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1225  159   25   43   43
##          B   47  561   23   59   93
##          C   21   56  705  129   79
##          D   90  123   49  521  110
##          E   12   50   53   52  576
```

```
print(paste0('Accuracy is ',round(confusion1$overall['Accuracy'],3)))
```

```
## [1] "Accuracy is 0.732"
```

```
#Random Forest
randForestPred <- predict(randForest,newdata = test)
confusion2 <- confusionMatrix(randForestPred,test$classe)
confusion2$table
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1394    4    0    0    0
##          B    1  945    1    0    0
##          C    0    0  854    0    0
##          D    0    0    0  804    3
##          E    0    0    0    0  898
```

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

```
## [1] "Accuracy is 0.998164763458401"
```

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.

Now, using the second model to predict the test.

```
predictresult <- predict(randForest,newdata = testing)
predictresult
```

```
##  [1] 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
```