Practical Machine Learning Project

Karen Jiang

3/3/2020

Executive Summary

We will use the data from http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har), The purpose of this project is to build a classification model to predict the human activity in which they did the exercise, based on the exercise data supplied. The human activity or the target variable “classe” has five categories, A, B, C, D and E, thus the desirable model would be a multi-class classificatioin model. We will explore using decision tree and generalized linear model to fit the model and perform the prediction.

Exploratory Analysis and Data Cleaning

The data were pre-partitioned into training and test sets. The training data has 160 variables. The target variable “classe” is distributed as follows:

```
#cat("Summary Statistics on Target Variable: \n")
table(training$classe)
```

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

We noticed that the first few variables, “X”, “user_name” and “timestamps” do not represent the exercise activity metrics, and they are not deemed useful for this model. Thus we will exclude them from the model. Additionally, there are many NAs which may cause the model fitting to fail.

The target variable `classe` does not have any NAs. However, there are 67 predictors that mostly consists NAs. We will exclude these variable from the predictor set.

```
sum(is.na(training$classe))
```

```
## [1] 0
```

```
na_count <-sapply(training, function(y) sum(is.na(y)))
cat("Variables with more than 19000 NAs:", sum(na_count>19000), '\n')
```

```
## Variables with more than 19000 NAs: 67
```

```
keep <- names(na_count[na_count==0])
training <- training[, keep]
training <- training[, -c(1:5)]
```

Additionally, we used `nearZeroVar` function in the caret package to identify and remove the variables with nearly zero variance.

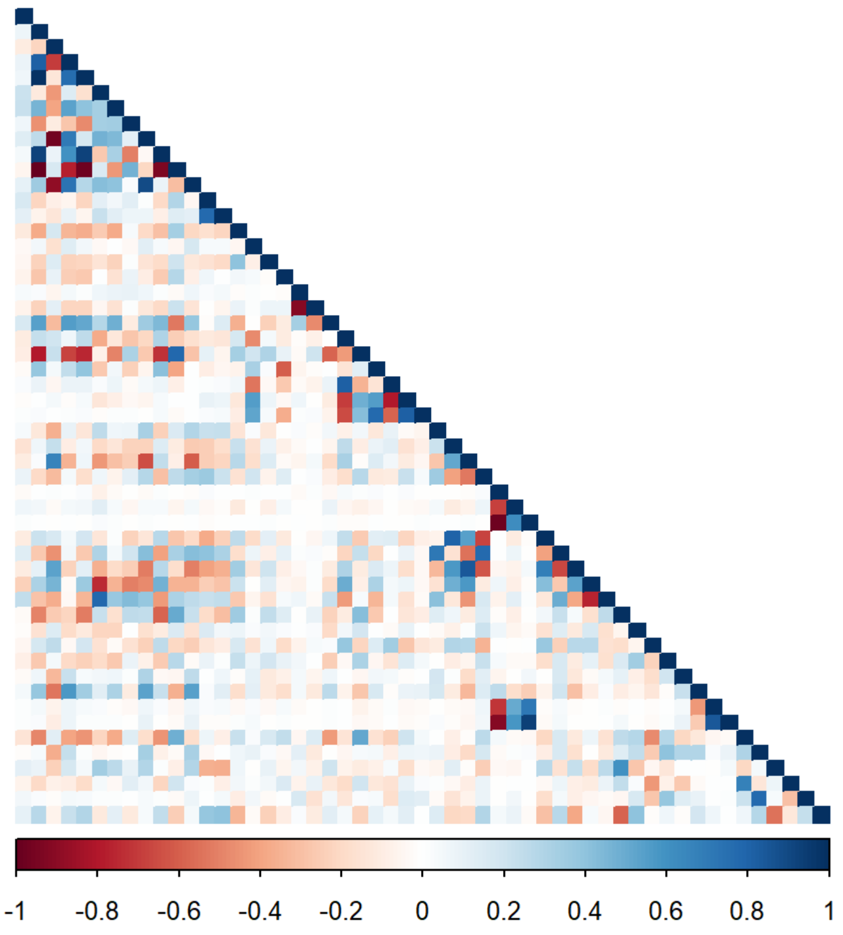
```
library(caret, quietly = TRUE)
nearZero <- nearZeroVar(training[, -88])
training <- training[, -nearZero]
```

This result in a training dataset with 54 variables, in which 53 are predictors and one target variable. The correlation analysis among the predictor variables indicated that some correlation among a small number of variables. We will proceed to split the dataset into train and set, and proceed to fit the model.

```
library(corrplot, quietly = TRUE)
```

```
## corrplot 0.84 loaded
```

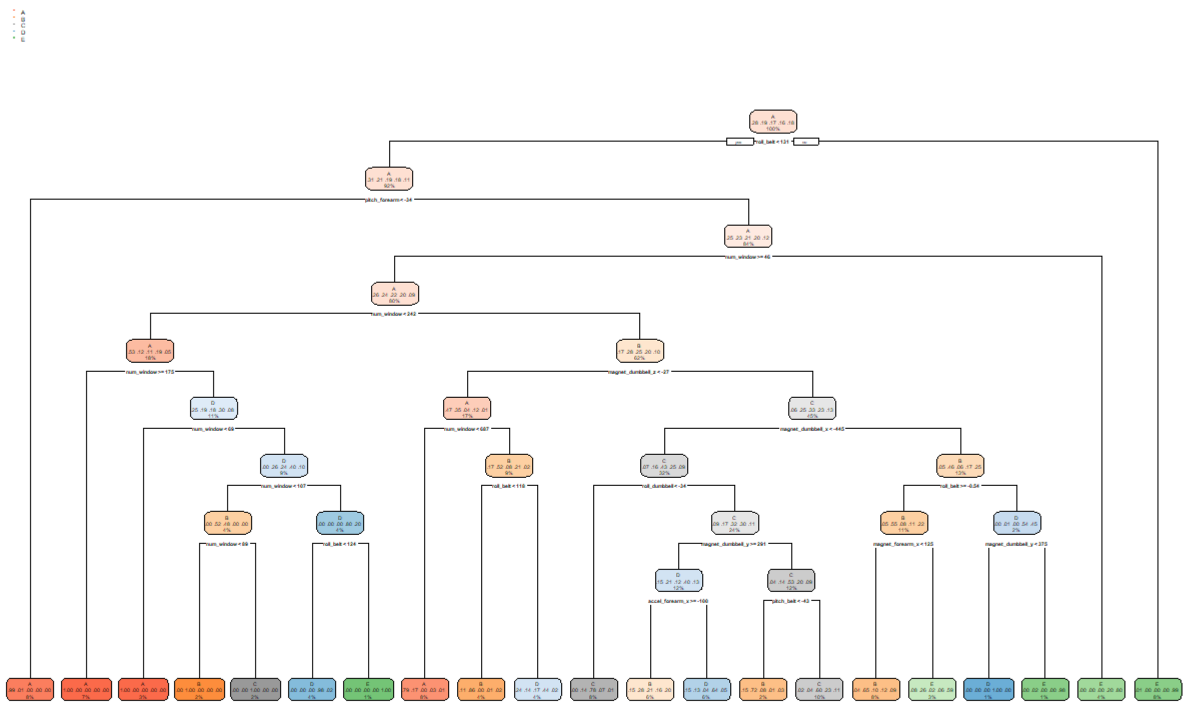
```
corr <- cor(training[, -54])
corrplot(corr, method = 'color', type='lower', tl.pos = 'n')
```



Decision Tree Model

Next, we will fit a decision tree model using rpart package.

```
library(caret, quietly = TRUE)
library(rpart, quietly = TRUE)
library(rpart.plot, quietly = TRUE)
intrain <- createDataPartition(training$classe, p = 0.7, list = FALSE)
train.df <- training[intrain, ]
test.df <- training[-intrain, ]
#lda <- train(pr.training, factor(training$classe), method='lda')
tree <- rpart(classe~., data=train.df, method = 'class')
rpart.plot(tree)
```



```
train.pred <- predict(tree, type='class')
#confusionMatrix(train.pred, factor(train.df$classe))
cat("Decision Tree Train accuracy: ", mean(train.pred == train.df$classe))
```

```
## Decision Tree Train accuracy:  0.7751329
```

```
test.pred <- predict(tree, test.df, type='class')
#confusionMatrix(test.pred, factor(test.df$classe))
cat("Decision Tree Test Accuracy: ", mean(test.pred == test.df$classe))
```

```
## Decision Tree Test Accuracy:  0.773322
```

```
#predict(tree, testing, type='class')
```

For this tree model, the train and test accuracy was around 80%. We will try other method to improve the accuracy.

Random Forest Model

```
library(randomForest, quietly = TRUE)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
rf <- randomForest(classe~., data=train.df, method='class')
cat("Random Forest Train Accuracy: ", mean(rf$predicted == rf$y))
```

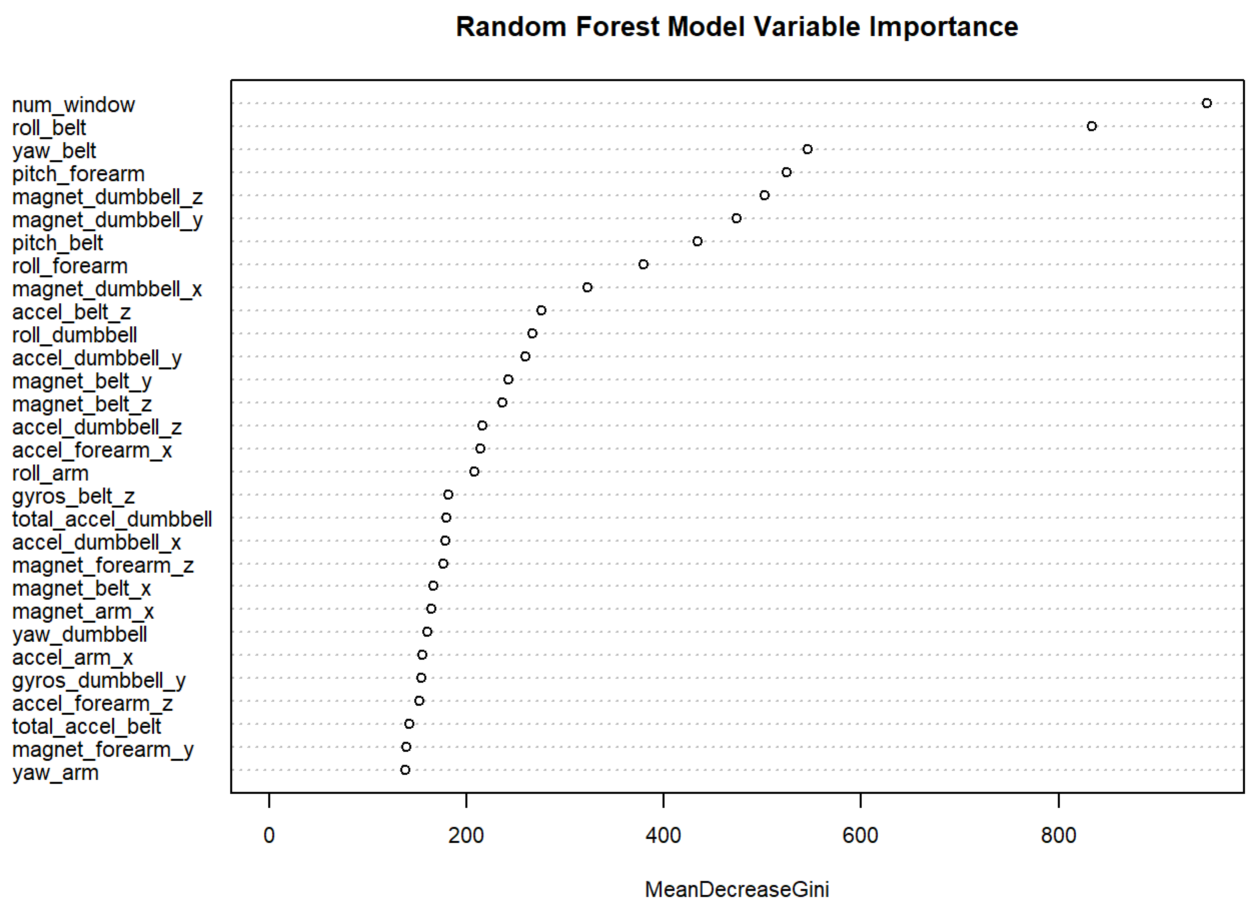
```
## Random Forest Train Accuracy:  0.9974521
```

```
rf.pred <- predict(rf, test.df, type = 'class')
cat('Randome Forest Test Accuracy: ', mean(rf.pred == test.df$classe))
```

```
## Randome Forest Test Accuracy:  0.9972812
```

The Random Forest model seems working well for both train and test data set, and this is our final model. The variable importance of this model are as follows:

```
varImpPlot(rf, main='Random Forest Model Variable Importance', cex=0.7)
```



```
predict(rf, testing, type='class')
```

```
##  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
```