Prediction Assignment

Vitawat Ngammuangpak

10/27/2017

The object of project want to predict the class of preformance by use data from accelerometers on belt, forearm, arm and dumbell of 6 participants.

1. Import data

After import data, we found 19,622 obs. in pml\_training data set and 20 obs. in pml\_testing data set. Both data set have 160 varaibles and found some varaible have "#DIV/0!" and a lot of missing value that need to clean.

## [1] "Importing training data.........."

## Warning: Missing column names filled in: 'X1' [1]

## Warning: 185 parsing failures.  
## row col expected actual  
## 2231 kurtosis\_roll\_arm a double #DIV/0!  
## 2231 skewness\_roll\_arm a double #DIV/0!  
## 2255 kurtosis\_roll\_arm a double #DIV/0!  
## 2255 skewness\_roll\_arm a double #DIV/0!  
## 2282 kurtosis\_roll\_arm a double #DIV/0!  
## .... ................. ........ .......  
## See problems(...) for more details.

## [1] 19622 160

## [1] "Importing testing data.........."

## Warning: Missing column names filled in: 'X1' [1]

## [1] 20 160

1. Data cleaning

First of all, for convenience to use data in the next time, we merge 2 data set together, then remove unuse varaible such as X1, user\_name, raw\_timestamp\_part\_1, raw\_timestamp\_part\_2, cvtd\_timestamp, new\_window and num\_window.

Due to the data value problem, we manage as follow

* Change value "#DIV/0!" to NA
* Delete varaible where contain all NA
* Delete varaible where contain all zero
* For varaible contain a lot of NA, we decide to remove varaible that have NA more than 50% of total case.
* For missing value, replace missing value with mean of that varaible

After clean the data, we have the new data set new\_pml\_training and new\_pml\_testing that ready to analysis.

# Change value "#DIV/0!" to NA  
total[total =="#DIV/0!"] <- NA   
  
# Remove varaible that contain all NA (NA column)  
total <- total[ ,colSums(is.na(total))<nrow(total)]   
  
# Remove varible that contain all zero (zero value column)  
total <- total[ ,colSums(total != 0, na.rm = TRUE) > 0]   
  
# Remove varible which contain NA more than 50 %  
total <- total[ ,colSums(is.na(total))/nrow(total) < 0.5]

1. Find Model

This project want to predict class of performance, used randonForest() and rpart() to find prediction model. Model selection will compare the accuracy value of each model. The step are as follows

* Devided new\_pml\_training to trainData 70% and testData set 30%, use createDataPartition() from caret package, defined p = 0.70
* Use trainData to train model by using randomforest() and rpart().("classe" is dependent varaible, all the rest varaible is predictors)
* Use each model predict "classe" on testData.
* Then use confusionMatrix() find accuracy.
* Select prediction model by compare accurracy value.

library(caret)  
set.seed(12345)  
trainIndex <- createDataPartition(y=new\_pml\_training$classe, p=0.70, list=FALSE)  
trainData <- new\_pml\_training[trainIndex,]  
testData <- new\_pml\_training[-trainIndex,]

library(rpart)  
set.seed(12345)  
model.rpart <- rpart(classe~., data= trainData)  
prediction.rpart <- predict(model.rpart, newdata= testData, type = "class")  
confus.rpart <- confusionMatrix(prediction.rpart, testData$classe)

## [1] "----------Result of rpart() model----------"

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction A B C D E  
## A 1498 196 69 106 25  
## B 42 669 85 86 92  
## C 43 136 739 129 131  
## D 33 85 98 553 44  
## E 58 53 35 90 790  
##   
## Overall Statistics  
##   
## Accuracy : 0.722   
## 95% CI : (0.7104, 0.7334)  
## No Information Rate : 0.2845   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.6467   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Statistics by Class:  
##   
## Class: A Class: B Class: C Class: D Class: E  
## Sensitivity 0.8949 0.5874 0.7203 0.57365 0.7301  
## Specificity 0.9060 0.9357 0.9097 0.94717 0.9509  
## Pos Pred Value 0.7909 0.6869 0.6273 0.68020 0.7700  
## Neg Pred Value 0.9559 0.9043 0.9390 0.91897 0.9399  
## Prevalence 0.2845 0.1935 0.1743 0.16381 0.1839  
## Detection Rate 0.2545 0.1137 0.1256 0.09397 0.1342  
## Detection Prevalence 0.3218 0.1655 0.2002 0.13815 0.1743  
## Balanced Accuracy 0.9004 0.7615 0.8150 0.76041 0.8405

library(randomForest)  
set.seed(12345)  
model.rf <- randomForest(classe~.,data= trainData)  
prediction.rf <- predict(model.rf, newdata = testData)  
confus.rf <- confusionMatrix(prediction.rf, testData$classe)

## [1] "----------Result of randonForest() model----------"

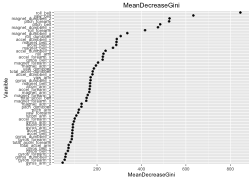
## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction A B C D E  
## A 1673 9 0 0 0  
## B 1 1127 10 0 0  
## C 0 3 1016 13 0  
## D 0 0 0 951 5  
## E 0 0 0 0 1077  
##   
## Overall Statistics  
##   
## Accuracy : 0.993   
## 95% CI : (0.9906, 0.995)  
## No Information Rate : 0.2845   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.9912   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: A Class: B Class: C Class: D Class: E  
## Sensitivity 0.9994 0.9895 0.9903 0.9865 0.9954  
## Specificity 0.9979 0.9977 0.9967 0.9990 1.0000  
## Pos Pred Value 0.9946 0.9903 0.9845 0.9948 1.0000  
## Neg Pred Value 0.9998 0.9975 0.9979 0.9974 0.9990  
## Prevalence 0.2845 0.1935 0.1743 0.1638 0.1839  
## Detection Rate 0.2843 0.1915 0.1726 0.1616 0.1830  
## Detection Prevalence 0.2858 0.1934 0.1754 0.1624 0.1830  
## Balanced Accuracy 0.9986 0.9936 0.9935 0.9927 0.9977

From the result, show the randomForest() accuracy is 0.9932 while rpart() accuracy is 0.7720. RandomForest() model is more accurate than rpart() model. So we decide to use model from ramdomForest() to predict.

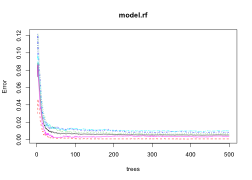
However, the accuracy and sensitively value of randomForest() model can calculate as follows

Total accuracy = (1673+1127+1017+950+1077)/(total case = 5885) = 0.9932 (99.32%)  
Sensitively class A = (1673)/(1673+1) = 0.9994 (99.94%)  
Sensitively class B = (1127)/(1127+9+3) = 0.9895 (98.95%)  
Sensitively class C = (1016)/(1016+10) = 0.9903 (99.03%)  
Sensitively class D = (951)/(950+13) = 0.9865 (98.65%)  
Sensitively class E = (1077)/(1077+5) = 0.9954 (99.54%)

MeanDecreaseGini represents how each variable is useful/important for prediction. Comparing the MeanDecreasGini values of randonForest() model from high to low, found roll\_belt is most useful/important varaible for prediction, followed by yaw\_belt, magnet\_dumbell\_z, pitch\_forearm while gyros\_arm\_z is the least useful/importance.



Consider the error rate (OOB:out of bag, class A, class B, Class C, class D and class E) for prediction on the different ntree. We found when number of tree is increase, each error rate will be decrease and close to zero . So if define suitable number of tree , the model will get low error rate and high accurate to predict.



1. Prediction

Use randomForest() model to predict 20 value on pml\_testing

## [1] "----- Predict 20 value use ramdomForest() -----"

prediction.rf.20 <- predict(model.rf, newdata = new\_pml\_testing)  
prediction.rf.20

## 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