PML Course Project -- Weight Lifting Exercises Prodiction

Yuling Tu

January 6, 2018

Synopsis

In this project, the goal is to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants and to predict the manner in which they did the exercise.

The data for this project come from http://groupware.les.inf.puc-rio.br/har. Special thanks to the following authors for being so generous in sharing their data for public usage.

Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H. Qualitative Activity Recognition of Weight Lifting Exercises. Proceedings of 4th International Conference in Cooperation with SIGCHI (Augmented Human '13) . Stuttgart, Germany: ACM SIGCHI, 2013.

Data Explortory and Cleaning

Briefly analyzing the data structure, columns and rows. There are 160 columns for both training and testing data set. But, lots of columns have NA data.

```
# read data
trainset <- read.csv(file ="C:/Users/l1ylt01/Documents/R/coursera/data/pml-training.csv")
testset <- read.csv(file ="C:/Users/l1ylt01/Documents/R/coursera/data/pml-testing.csv")
dim(trainset)

## [1] 19622   160

dim(testset)

## [1]  20 160

#str(trainset) -- won't show to reduce page size
```

First, partiction the training data into a training and testing data set. 60/40 partition ratio is used in this case.
Next, data cleaning is performed as below.

1) Lots of columns has NA data and NA data will affect the model training. So, we will remove any column or variable has more tha 90% of NA data.
2) If the numeric is closed to 0, it will skew the model prediction. nearZeroVar function is used.

3) Remove the data elements which are not related to outcome we try to predict, such as X, user_name, timestamp (3 columns), new_window, num_window.

```
# load library
library(pgmm); library(e1071);

library(rpart); library(gbm);

library(caret);library(randomForest);

# seperate trainset into training and testing set
intrain<- createDataPartition(trainset$classe, p=0.6, list=FALSE)
training<- trainset[intrain, ]
testing <- trainset[-intrain, ]
dim(training) ; dim(testing)

## [1] 11776   160

## [1] 7846  160

#remove 90% NA variables
na <- sapply(training, function(x) mean(is.na(x))) > 0.90
training <- training[, na==FALSE]
testing  <- testing[, na==FALSE]

#remove near zero variables
nzv <- nearZeroVar(training)
training <- training[, -nzv]
testing  <- testing[, -nzv]

# remove non-variable colmns
training <- training[, -(1:7)]
testing  <- testing[, -(1:7)]

dim(training)

## [1] 11776    52

dim(testing)

## [1] 7846   52
```

Model Selection

After data cleaning process above, only 51 variables remain along with the outcome (classe).

Then, the following session is to identify which model has the highest accuracy against training data. The following four models are used.
1) Classification Tree (rpart)
2) Random Forest (rf)

3) Generalized Boosted Regression (gbm)
4) Linear discriminant analysis (lda) -- just for fun

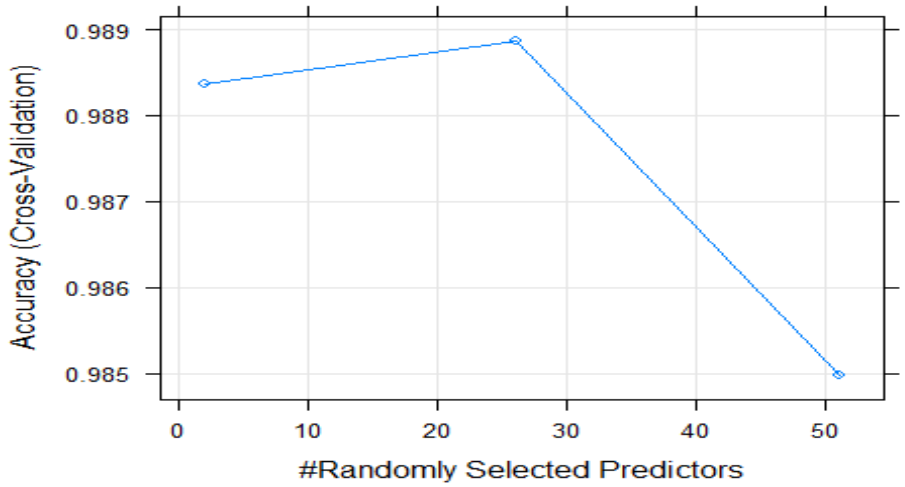
Each model was trained with 5-fold cross-validation to select optimal tuning parameters for the model.

```
# set seed
set.seed(433)
# train 4 models
rpart <- train(classe ~ .,data = training, method = "rpart",
trControl=trainControl(method="cv", number=5, verboseIter=F))
rf <- train(classe ~ .,data = training, method = "rf",
trControl=trainControl(method="cv", number=5, verboseIter=F))
gbm <- train(classe ~ .,data = training,method = "gbm",
trControl=trainControl(method="cv", number=5, verboseIter=F), verbose =
FALSE)
lda <- train(classe ~ .,data = training, method = "lda",
trControl=trainControl(method="cv", number=5, verboseIter=F))

# accuracy & out-of-sample error
rpartcm <- confusionMatrix(predict(rpart, testing), testing$classe)
rfcm <- confusionMatrix(predict(rf, testing), testing$classe)
gbmcm <- confusionMatrix(predict(gbm, testing), testing$classe)
ldacm <- confusionMatrix(predict(lda, testing), testing$classe)

accuracycompare <- data.frame(
  Model = c('RPART', 'RF', 'GBM', 'LDA'),
  Accuracy = rbind(rpartcm$overall[1], rfcm$overall[1], gbmcm$overall[1],
ldacm$overall[1])
)

accuracycompare

##   Model  Accuracy
## 1 RPART 0.5694621
## 2    RF 0.9926077
## 3   GBM 0.9571756
## 4   LDA 0.6881213

rfac <- rfcm$overall[1]
rfoos <- 1-rfcm$overall[1]
```

After comparing four different models, the Random Forest has the highest accuracy rate of 99.26% and the out-of-sample error is 0.74%.

So, Random Forest is picked as the final model to predict the test data set.

```
plot(rf)
```



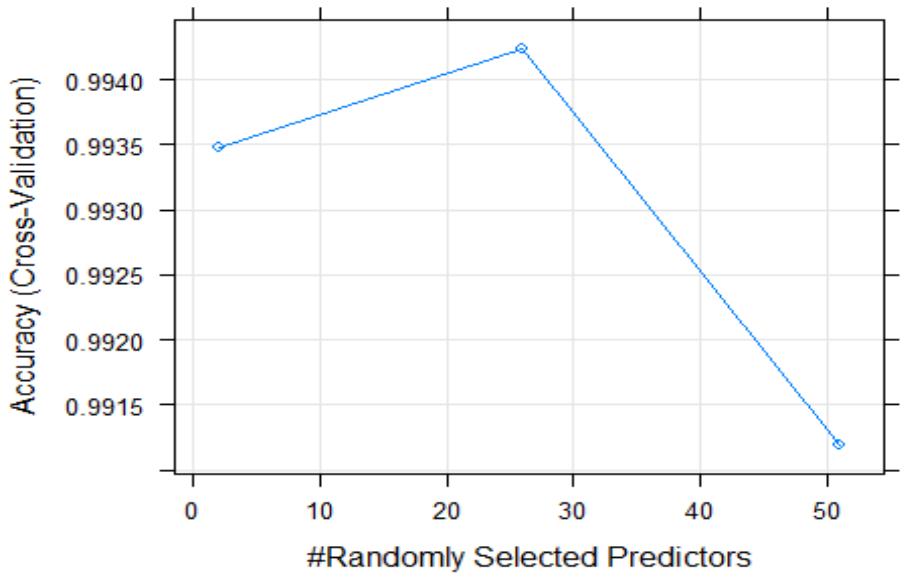
However, the previous Random Forest model was trained with 60/40 data partition. The Random Forest model plot did show the accuracy change through 5-fold cross-validation and the ratio is slight high.

Final Model Retrain

So, will re-train the Random Forest model by using the whole training data set. And, the same data cleaning process need to be run through the original training and testing data set.

```
#remove 90% NA variables
na <- sapply(trainset, function(x) mean(is.na(x))) > 0.90
trainfinal <- trainset[, na==FALSE]
testfinal <- testset[, na==FALSE]

#remove near zero variables
nzv <- nearZeroVar(trainfinal)
trainfinal <- trainfinal[, -nzv]
testfinal  <- testfinal[, -nzv]

# remove non-variable colmns
trainfinal<- trainfinal[, -(1:7)]
testfinal  <- testfinal[, -(1:7)]

dim(trainfinal)

## [1] 19622    52

dim(testfinal)

## [1] 20 52
```

```
# fit the model and make sure importance is TRUE for later usage
rffinal<- train(classe ~ .,data = trainfinal, method = "rf",
trControl=trainControl(method="cv", number=5, verboseIter=F) ,importance =
TRUE)
plot(rffinal)
```



The model accuracy change ratio is less than the previous Random Forest model using 60% of training data. This final Random Forest model would prodict the testing data better.

Prediction Results

```
predfinal <- predict(rffinal, testfinal)
# print out the final predict for Test data set
predictionresults <- data.frame(
  problem_id=testfinal$problem_id,
  predicted=predfinal
)
predictionresults
```

```
##    problem_id predicted
## 1           1         B
## 2           2         A
## 3           3         B
## 4           4         A
## 5           5         A
## 6           6         E
## 7           7         D
## 8           8         B
## 9           9         A
```

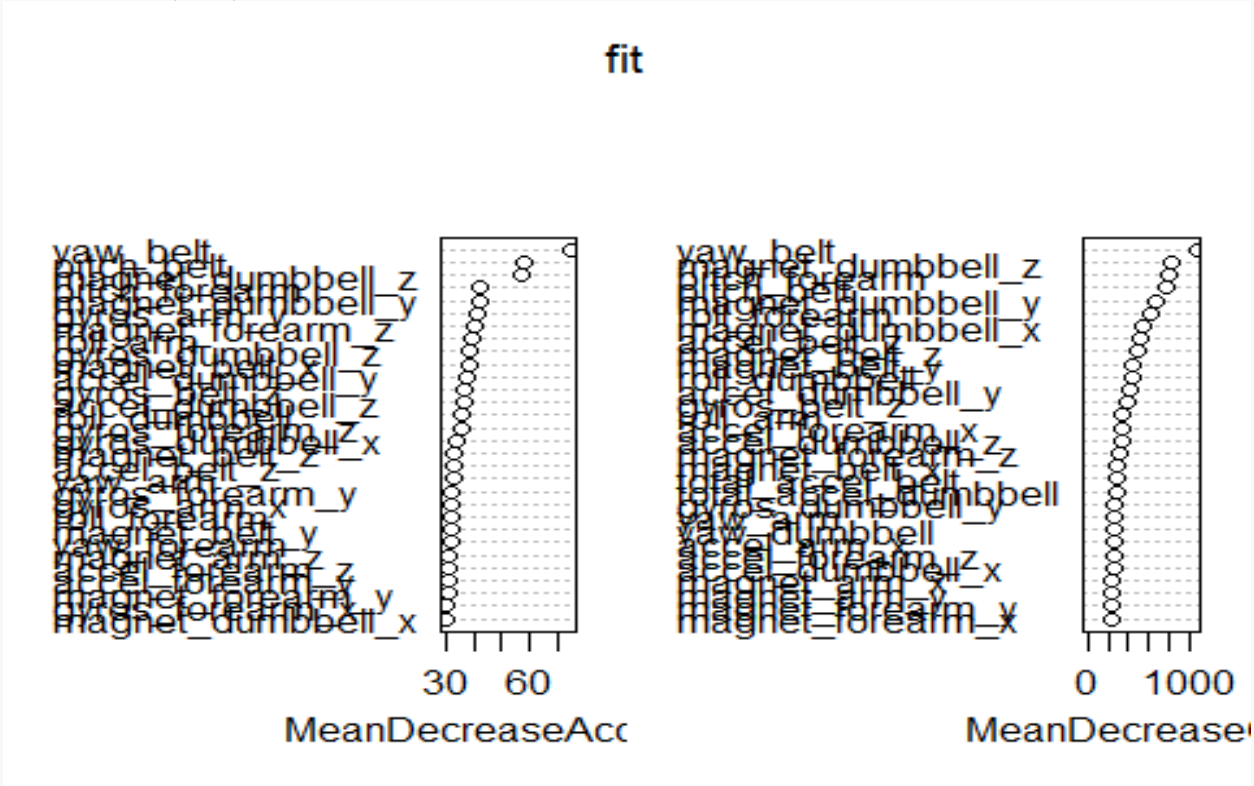
```
## 10          10          A
## 11          11          B
## 12          12          C
## 13          13          B
## 14          14          A
## 15          15          E
## 16          16          E
## 17          17          A
## 18          18          B
## 19          19          B
## 20          20          B
```

Randon Forest Package

After research, there are more plot functions to plot the model with RandomForest package instead of caret package. So, we will use RandomForest package to cross check that random forest is indeed the good model for this data set.

Variable importance

```
# evaluate importance variables
fit<-randomForest(classe~., data=trainfinal,
trControl=trainControl(method="cv", number=5, verboseIter=F),
importance=TRUE)
# Variable Importance plot
varImpPlot(fit)
```



The first graph (MeanDecreaseAccuracy) shows that if a variable is assigned values by random permutation by how much will the MSE increase. So in this case if yaw_belt is randomly permuted, the MSE will increase by 75% on an average. Higher the value, higher the variable importance.

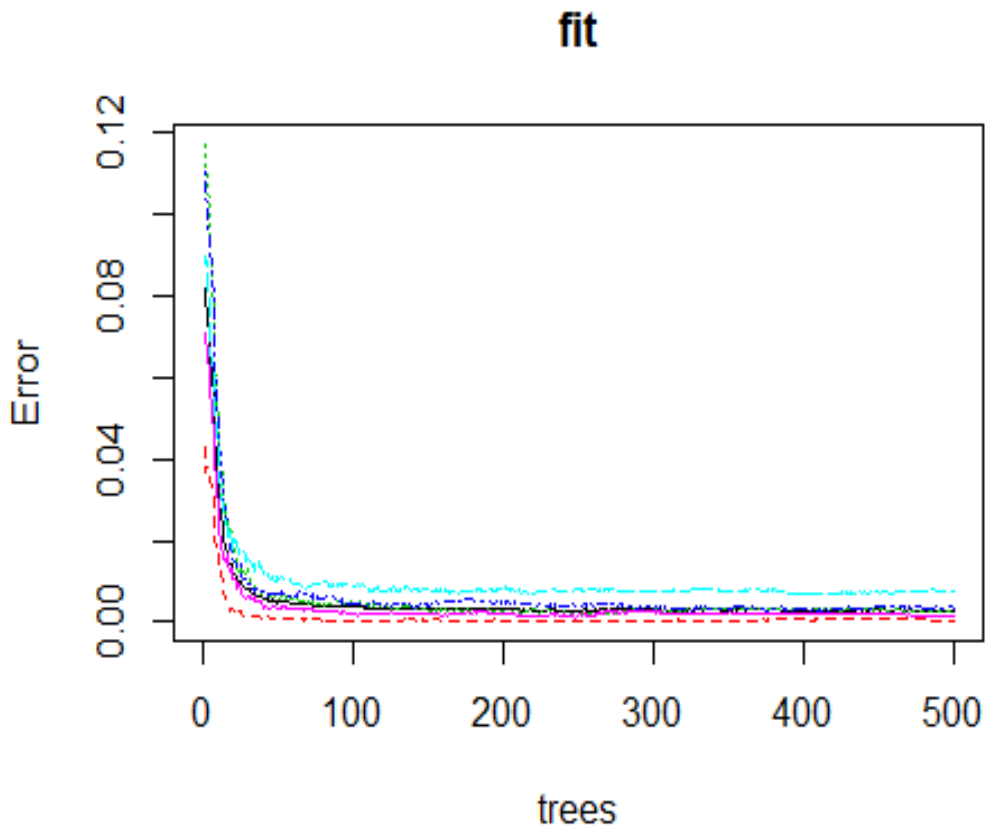
On the other hand, Node purity is measured by Gini Index which is the the difference between RSS before and after the split on that variable.

Since the concept of criteria of variable importance is different in two cases, you have different rankings for different variables.

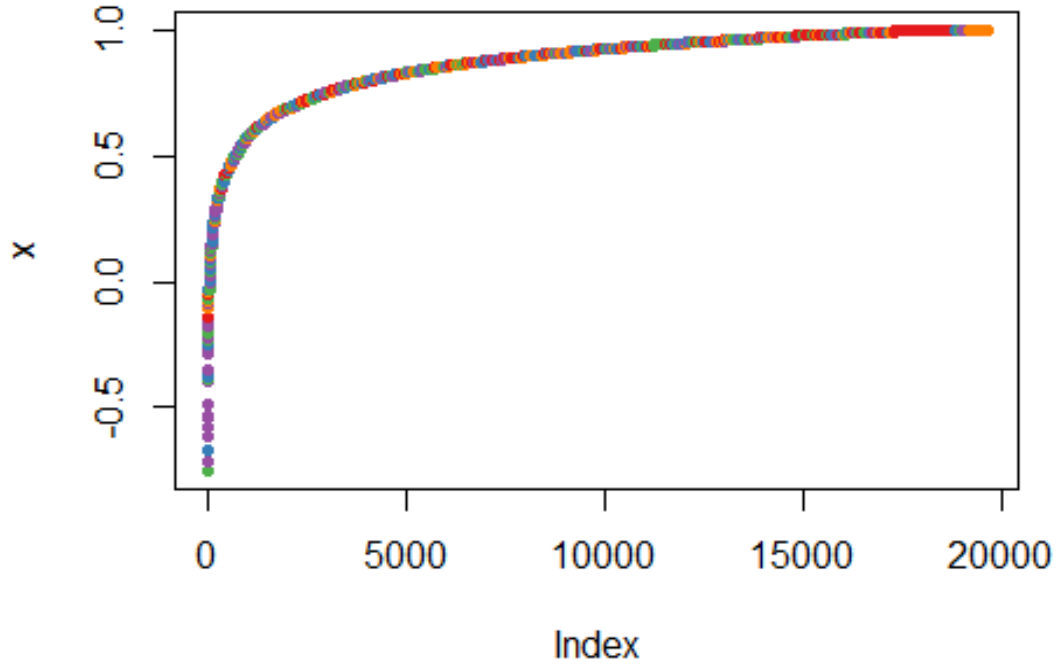
There is no fixed criterion to select the "best" measure of variable importance it depends on the problem you have at hand.

This varImpPlot forms a part of prediction power of your Random Forest Model. If you drop the top variable from your model, it's prediction power will greatly reduce.

```
plot(fit)
```




```
plot(margin(fit,trainfinal$classe))
```



The first plot indicates the error for your different classes (colored) and out-of-bag samples (black) over the amount of trees. Classes are in the same order as the results you get from print(fit). The error seems to be drop to close to 0 around 30 ~ 40 trees for this model.

n ensemble classification, you mostly do a majority vote from all models in the ensemble, with the class voted most becoming the finally predicted class. The margin thereby indicates the ratio of votes having been done for the correct class for all samples. 1 indicates that for one sample, all votes of the ensemble were correct, while e.g. 0 indicates a draw between the correct and the next best classes. Therefore, values >0 mean that the majority was right, hence this sample ended up as being predicted correct - whilst all values <0 mean that the majority was wrong, hence the sample ended up as being classified wrong. Again, colors indicate classes, so in the example above you see that nearly all A & E samples got classified correctly, but some B, C & D are classified wrong. However, the curve and the density around 0 are still indicated Random Forest is the right model to this data set.