Machine Learning Assignment

Prediction of the manner in which a human subject performed a dumbbell (1.25kg) exercise based on accelerometer measurements.

Synopsis

The goal of this project is to predict the manner in which a human subject did the dumbbell (1.25kg) exercise. We used th "classe" variable in the training set. We created a training and a test data to do the analysis. Our prediction model was build using random forest and classsification trees. We also used our model to predict 20 different test cases.

Data

The data for this project come from this source: Human Activity Recognition Website. Data was provided in two files the **`pml-training.csv`** and the **`pml-testing.csv`**. Our variable of interest was the `classe` (A, B, C, D or E) variable, our model aims to predict which class each human belonged.

Loading and Summary of the Data

After dwonloading the testing and training data files, both data sets had 160 variables. For our analysis only columns related to belt, arm or dumbbell measurements were kept as predictors. Additionally, any columns containing `NA` or `#DIV/0!` values were excluded as predictors and these constraints yielded 52 predictors.

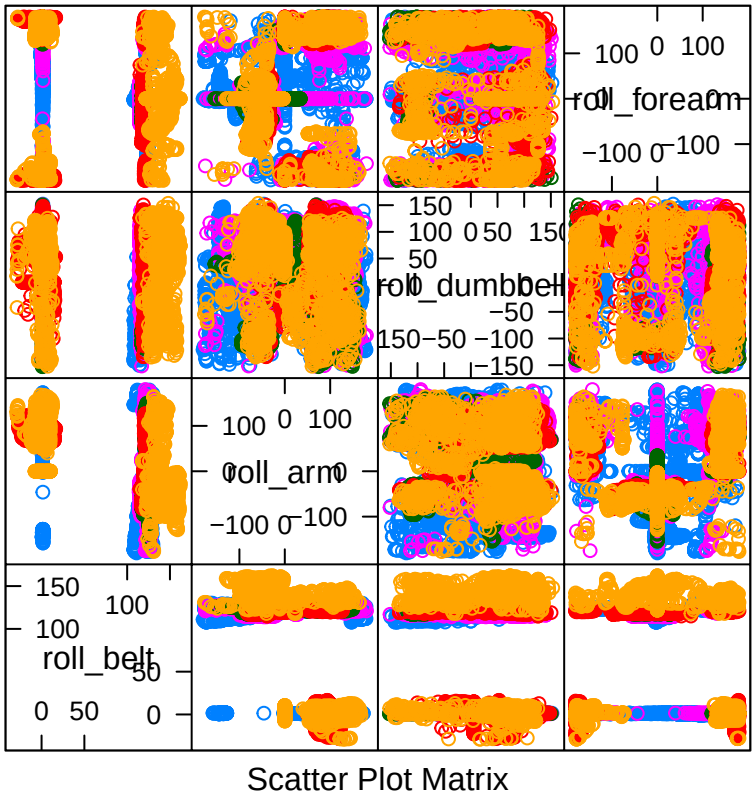
1. Transform Data

The training data set was then partitioned into `training_train` - 60% and `training_test` - 40% for building our model on using a . A feature plot was then prepared to observe for paterns and interdependence between predictors.

```
# dataframes containing only the intended predictors
pmlTrain <- pml_training[,colnames_predictors]
pmlTrain$classe <- pml_training$classe
pmlTest <-  pml_testing[,c(colnames_predictors)]
pmlTest$classe <- pml_testing$classe

#in training data create a training and testing data
inTrain <- createDataPartition(y=pmlTrain$classe,
                               p=0.6, list=FALSE)
pmlTrain_train <- pmlTrain[inTrain,]
pmlTest_test <- pmlTrain[-inTrain,]

#in training data create a training and testing data
inTrain <- createDataPartition(y=pmlTrain$classe,
                               p=0.6, list=FALSE)
pmlTrain_train <- pmlTrain[inTrain,]
pmlTest_test <- pmlTrain[-inTrain,]

#columns related to be used as predictors
```

```
is_Roll <- (substr(names(pmlTrain_train),1,4) == "roll")
is_Yaw <- (substr(names(pmlTrain_train),1,3) == "yaw")
is_Pitch <- (substr(names(pmlTrain_train),1,5) == "pitch")
```

```
fplot1 <- featurePlot(x=pmlTrain_train[,names(pmlTrain_train)[is_Roll]], y= pmlTrain_train$classe, plot=
print(fplot1)
```



-1.pdf

2. Building the classification tree model

We created a classification tree usinf the `rpart` method in the `carat` package.

```
#create a classification tree
modFit1 <- train(classe ~ .,method="rpart",data=pmlTrain_train)
```

```
## Loading required package: rpart
```

```
print(modFit1$finalModel)
```

```
## n= 11776
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
```

```
##  1) root 11776 8428 A (0.28 0.19 0.17 0.16 0.18)
##    2) roll_belt< 130.5 10774 7437 A (0.31 0.21 0.19 0.18 0.11)
##      4) pitch_forearm< -34.55 931    3 A (1 0.0032 0 0 0) *
##      5) pitch_forearm>=-34.55 9843 7434 A (0.24 0.23 0.21 0.2 0.12)
##       10) magnet_dumbbell_y< 436.5 8280 5923 A (0.28 0.18 0.24 0.19 0.11)
##         20) roll_forearm< 122.5 5103 2986 A (0.41 0.18 0.18 0.16 0.059) *
##         21) roll_forearm>=122.5 3177 2132 C (0.076 0.18 0.33 0.23 0.18) *
##       11) magnet_dumbbell_y>=436.5 1563  770 B (0.033 0.51 0.044 0.23 0.18) *
##    3) roll_belt>=130.5 1002   11 E (0.011 0 0 0 0.99) *
```

The misclassification error rate observed was 0.5 This value is close to 0.5 then `no purity` thus we decided to use a random forest model The prediction table was

```
#kable(tab1, format = "markdown")
tab1 <- table(pmlTrain_train$classe,pred1)
print(tab1)
```

```
##    pred1
##        A    B    C    D    E
##   A 3045   52  240    0   11
##   B  914  793  572    0    0
##   C  941   68 1045    0    0
##   D  833  364  733    0    0
##   E  301  286  587    0  991
```

The classification tree produced was

Conclusion: with default settings `rpart` model seems not satisfactory

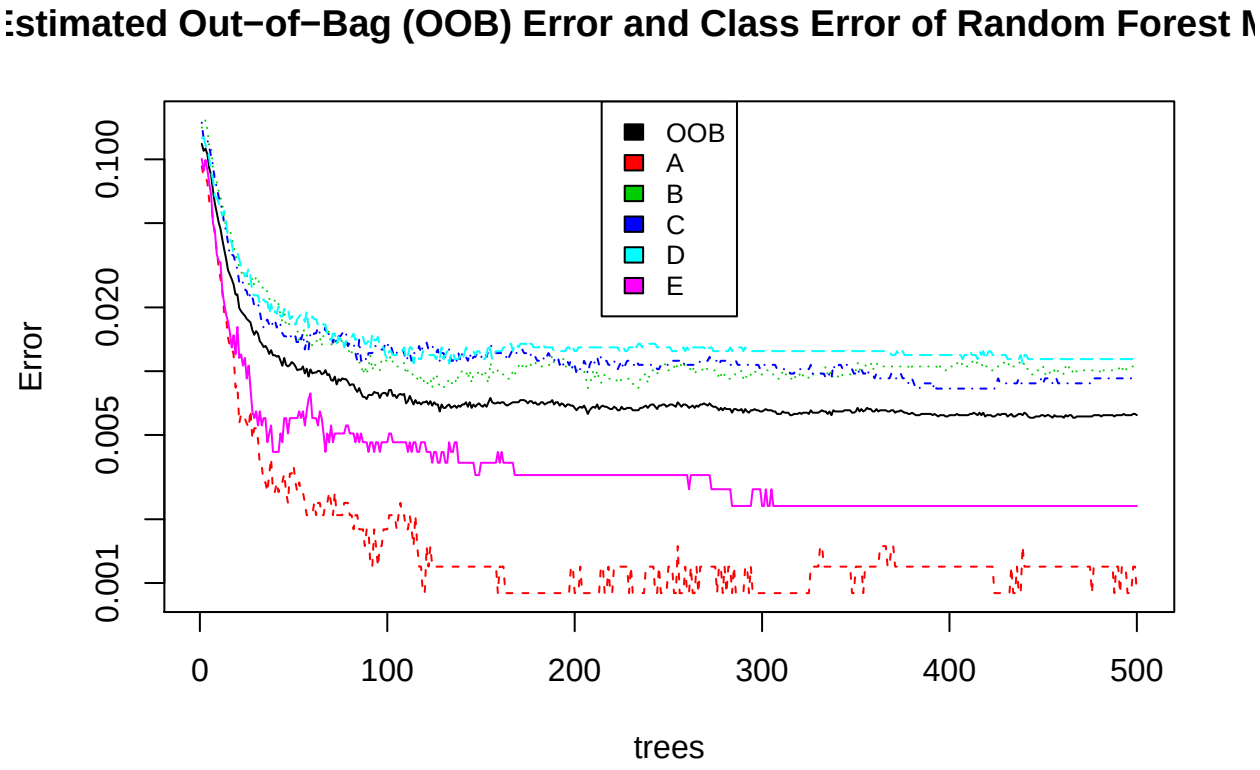
3. Building the random forest model

After the classification tree model, we decided to build a random forest model. The generic plot using model from randomForest is as shown below

```
# create a Random Forest model
modFit2 <- randomForest(classe ~ ., data=pmlTrain_train)
# display model fit results
modFit2
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = pmlTrain_train)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.62%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3345    3    0    0    0 0.0008960573
## B   19 2255    4    0    1 0.0105309346
## C    0   17 2035    2    0 0.0092502434
## D    0    0   21 1908    1 0.0113989637
## E    0    0    0    5 2160 0.0023094688
```

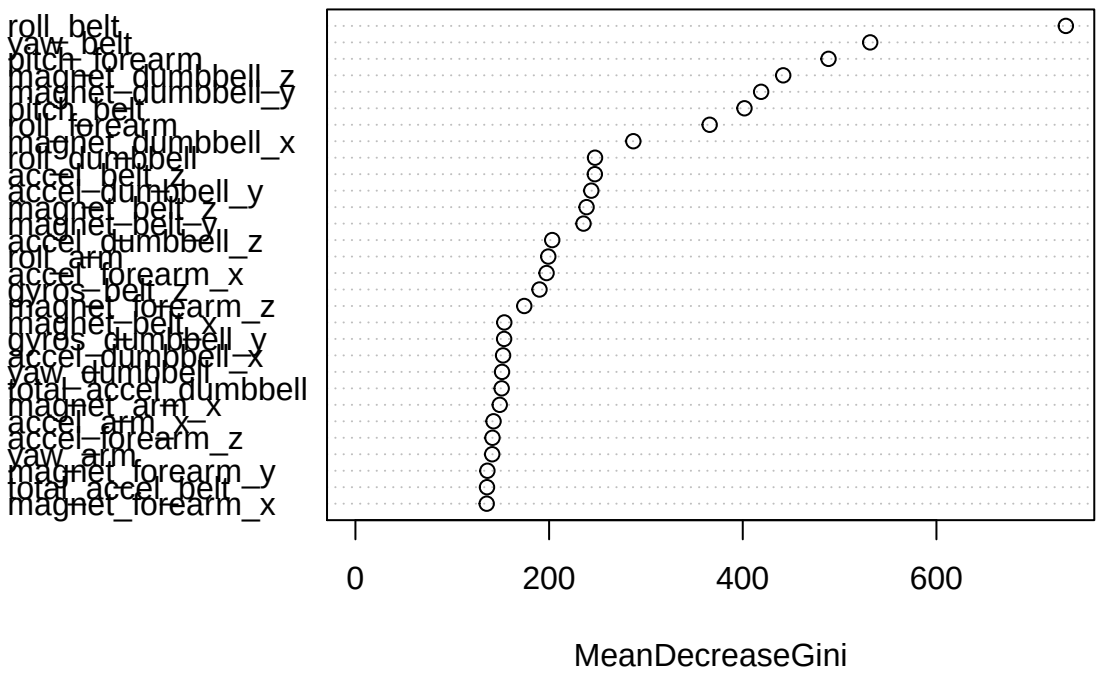
```
# Generic plot using model from randomForest
plot(modFit2, log="y",
     main="Estimated Out-of-Bag (OOB) Error and Class Error of Random Forest Model")
legend("top", colnames(modFit2$err.rate), col=1:6, cex=0.8, fill=1:6)
```



-1.pdf

We also did a dot plot for the variables of importance

```
# Dotchart of variable importance as measured by randomForest
varImpPlot(modFit2, main="Variable Importance in the Random Forest Model")
```



-1.pdf

```
#plot1 <- varImp(modFit2, scale = FALSE)
#plot(plot1, top = 20)
```

The random forest model had convincing results. The model had a misclassification error rate of 0.0039511 and the out of sample error rate of 0.0039511.

```
#kable( tab2 , format = "markdown")
tab2 <- table(pmlTest_test$classe,pred2)
print(tab2)
```

```
##    pred2
##        A    B    C    D    E
##   A 2230    2    0    0    0
##   B    0 1517    1    0    0
##   C    0    5 1360    3    0
##   D    0    0   13 1273    0
##   E    0    0    0    7 1435
```

Writing the ouputs

```
# write prediction answers to files
pml_write_files = function(x){
  n = length(x)
  for(i in 1:n){
```

```
    filename = paste0("problem_id_", i, ".txt")
    write.table(x[i], file=filename, quote=FALSE, row.names=FALSE, col.names=FALSE)
  }
}
pml_write_files(pred)
```