Practical Machine Learning Course Project

Objective: The objective of this project is to undestand how well an individual is doing each type of exercise like running,sitting,walking and so. We have the data collected from the group of 6 individuals using devices such as Jawbone Up, Nike FuelBand, and Fitbit. We will find any patterns in their activity and try to predict the manner in which they did the exercise.

Report: We will predict the exercise manner by buidling the predictive model on the classe variable which categorises the activity in different classes like A,B,C,D and E.

Lets start by importing the train data. We will procced with following main step.

1. Import the train data to perform initial analysis.
2. Divide train data in train and test
3. Build a predictive model on the train data and use cross validation to avoid overfitting if needed.
4. Use the model to predict test data. Check the accuracy of model.
5. Apply the final model on the validation set.

PART 1 :Cleaning the data and subsetting it

```
# Import the data and replace blank records with NA's
training<-read.csv("./pml-training.csv", header=TRUE, na.strings = c("", " ","NA"))

# We will remove the unnecessary columns from the dataset and will use only numeric fields as predict
ors
training<-training[,-c(1,2,3,4,5,6,7)]

# We will remove NA columns for further clean up as keeping them will affect our prediction model
training<-training[, !apply(training, 2, function(x) any(is.na(x)))]

dim(training)
```

```
## [1] 19622    53
```

Next part is subsetting the training data further into train and test dataset.

```
#divide the training set into 60% train and 40% test
library(caret)
```

```
## Loading required package: lattice
## Loading required package: ggplot2
```

```
partition<-createDataPartition(training$classe,p=0.60,list=FALSE)
train<-training[partition,]
test<-training[-partition,]
```

PART 2 :Model building and testing accuracy

Now,we have the cleaned data so lets decide which prediction algorithm we want to use. We are trying to predict the variable calsse which is the categorical variable hence we will use classification trees.

Model 1: classification Tree

```
# Set seed to reproduce the results.Number is not important, just make sure you use the same number n
ext time to reproduce same results
set.seed(32332)
library(party)
```

```
## Loading required package: grid
## Loading required package: mvtnorm
## Loading required package: modeltools
## Loading required package: stats4
## Loading required package: strucchange
## Loading required package: zoo
##
## Attaching package: 'zoo'
##
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
##
## Loading required package: sandwich
```

```
model<-ctree(classe~.,data=train)
#plot(model)

result<-predict(model,test[,-53],type="response")

# Use confusion metrix to obtain accuracy
confusionMatrix(result,test$classe)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 2077  123   10   40   23
##          B   59 1219   62   56   40
##          C   31   80 1207   58   36
##          D   49   39   56 1104   44
##          E   16   57   33   28 1299
## 
## Overall Statistics
## 
##                Accuracy : 0.8802          
##                  95% CI : (0.8728, 0.8873)
##     No Information Rate : 0.2845          
##     P-Value [Acc > NIR] : < 2.2e-16       
## 
##                   Kappa : 0.8484          
##  Mcnemar's Test P-Value : 7.812e-07       
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9306   0.8030   0.8823   0.8585   0.9008
## Specificity            0.9651   0.9657   0.9684   0.9713   0.9791
## Pos Pred Value         0.9138   0.8489   0.8548   0.8545   0.9065
## Neg Pred Value         0.9722   0.9534   0.9750   0.9722   0.9777
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2647   0.1554   0.1538   0.1407   0.1656
## Detection Prevalence   0.2897   0.1830   0.1800   0.1647   0.1826
## Balanced Accuracy      0.9478   0.8844   0.9253   0.9149   0.9400
```

Accuracy of this model is not that great. let’s try randomForest classification tree.

Model 2: Random Forest

```
set.seed(32332)
library(randomForest)
```

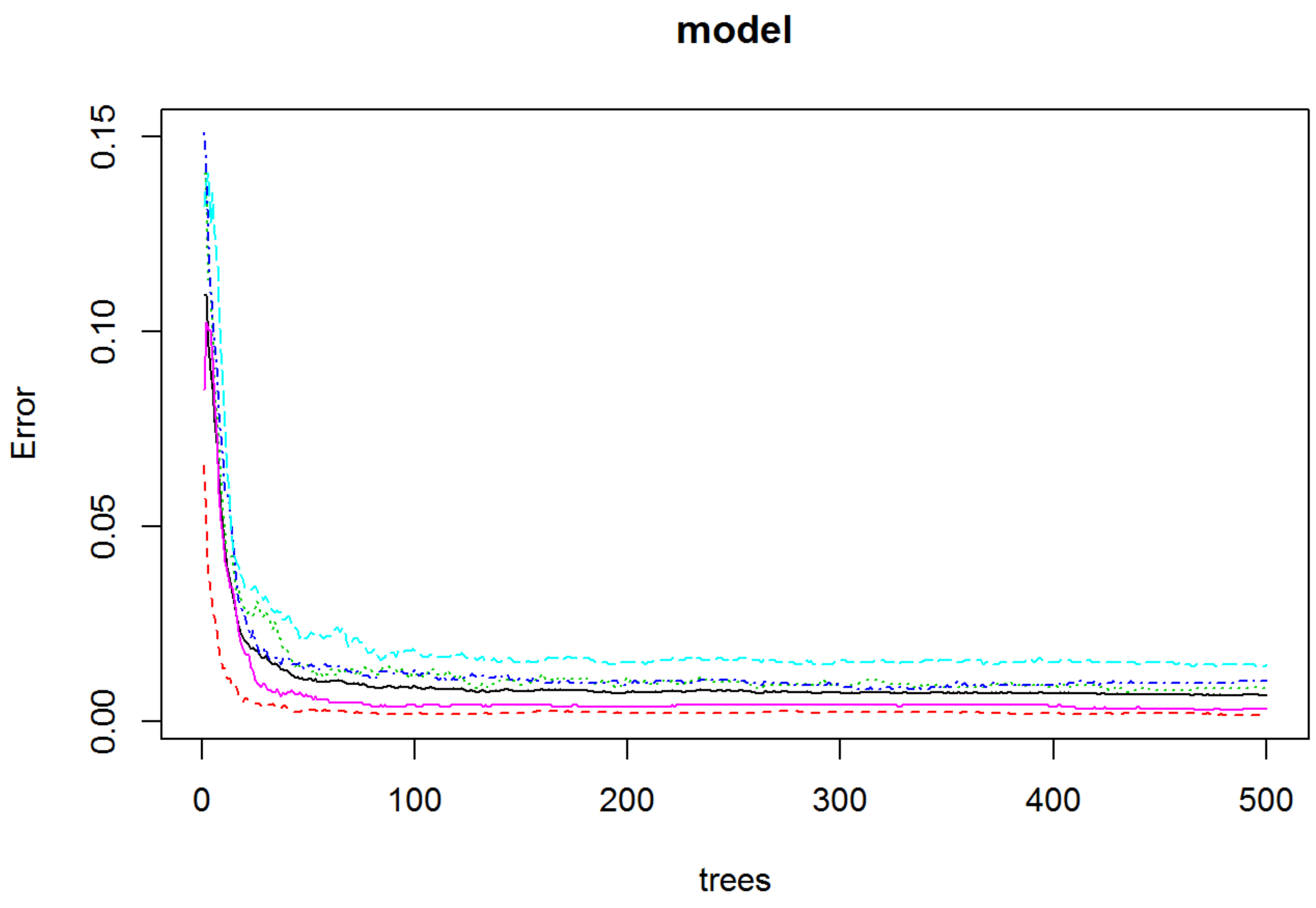
```
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

```
#Build random forest tree to predict classe variable with rest all variables as predictors
model<-randomForest(classe~.,data=train,importance=TRUE,ntree = 500)


model
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = train, importance = TRUE,      ntree = 500)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.68%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3343    4    0    0    1 0.001493429
## B   16 2260    3    0    0 0.008336990
## C    0   15 2033    6    0 0.010223953
## D    0    0   24 1902    4 0.014507772
## E    0    0    2    5 2158 0.003233256
```

```
#Plotting the model shows reduced error rate for each class as number of nodes increases
plot(model)
```



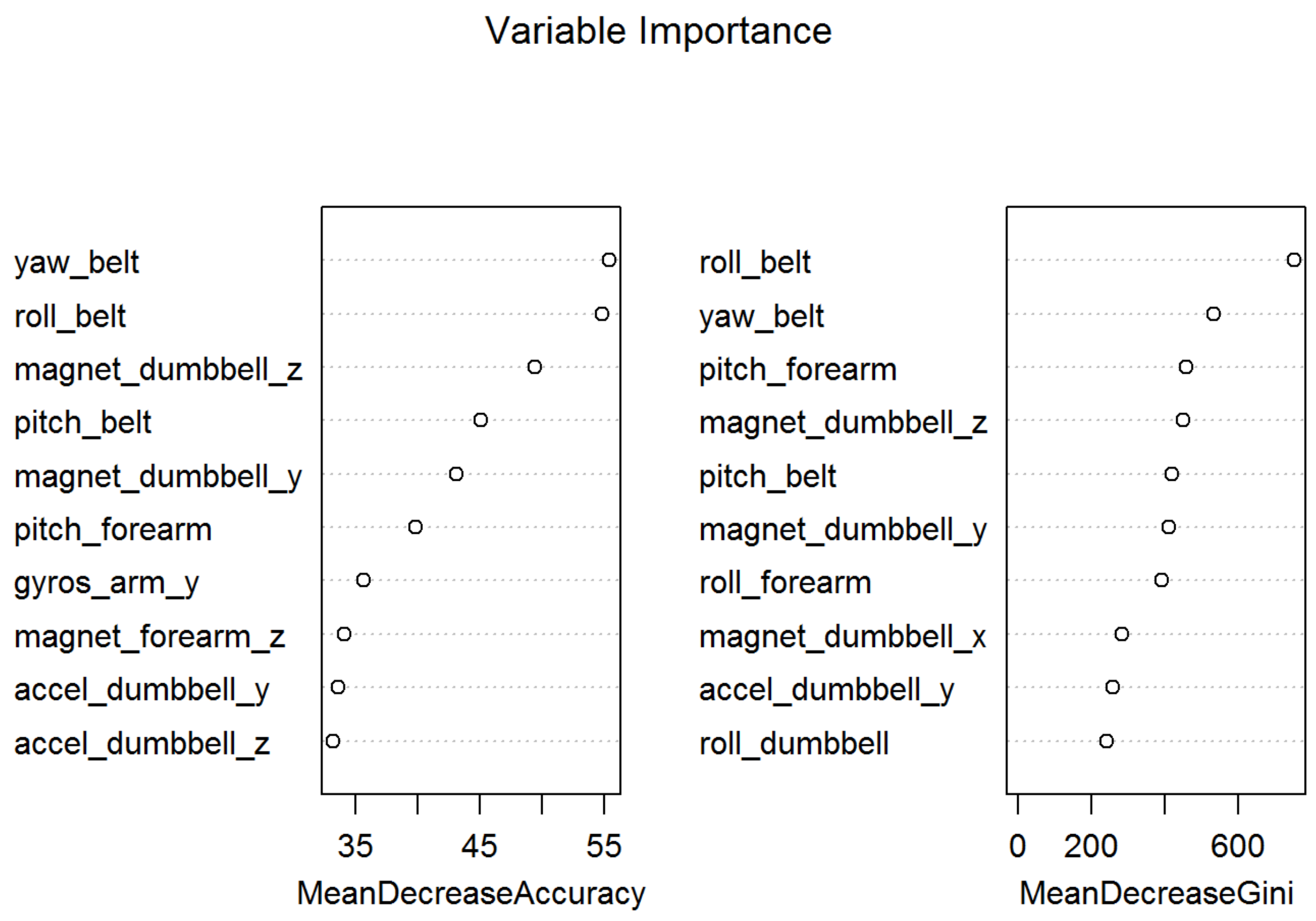
```
#Predict using the model on test data set
result<-predict(model,test[,-53],type="response")
```

In random forests, there is no need for cross-validation or a separate test set to get an unbiased estimate of the test set error. It is estimated internally, during the run. When the training set for the current tree is drawn by sampling with replacement, about one-third of the cases are left out of the sample. This oob (out-of-bag) data is used to get a running unbiased estimate of the classification error as trees are added to the forest.

```
# Use confusion metrix to obtain accuracy
confusionMatrix(result,test$classe)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 2230    4    0    0    0
##          B    2 1510   22    0    0
##          C    0    4 1346   17    4
##          D    0    0    0 1269    3
##          E    0    0    0    0 1435
## 
## Overall Statistics
## 
##                Accuracy : 0.9929
##                  95% CI : (0.9907, 0.9946)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.991
##  Mcnemar's Test P-Value : NA
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9991   0.9947   0.9839   0.9868   0.9951
## Specificity            0.9993   0.9962   0.9961   0.9995   1.0000
## Pos Pred Value         0.9982   0.9844   0.9818   0.9976   1.0000
## Neg Pred Value         0.9996   0.9987   0.9966   0.9974   0.9989
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2842   0.1925   0.1716   0.1617   0.1829
## Detection Prevalence   0.2847   0.1955   0.1747   0.1621   0.1829
## Balanced Accuracy      0.9992   0.9955   0.9900   0.9932   0.9976
```

```
#Plot to indicate top 10 important variables for the model
varImpPlot(model,sort=TRUE, n.var=10,main="Variable Importance")
```



Here estimate of test error (out of sample error) is 0.63%,very less and Accuracy of random forest is better than classification tree, so clearly we have the winner.

PART 3: Testing the predictions on validation set

```
# Import the validation data and replace blank records with NA's
validation<-read.csv("./pml-testing.csv", header=TRUE, na.strings = c("", " ","NA"))

# Repeat the same cleaning stpes we used on training data
validation<-validation[,-c(1,2,3,4,5,6,7)]
validation<-validation[, !apply(validation, 2, function(x) any(is.na(x)))]
dim(validation)
```

```
## [1] 20 53
```

```
final.model<-predict(model,validation)
final.model
```

```
##  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
```

Conclusion: Random forest is better at predicting values for categorical variables.