Practical Machine Learning - Course Project

by Senthil Kumar V.

Background, Data & Goal

Using devices such as Jawbone Up, Nike FuelBand, and Fitbit it is now possible to collect a large amount of data about personal activity relatively inexpensively. These type of devices are part of the quantified self movement - a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. One thing that people regularly do is quantify how much of a particular activity they do, but they rarely quantify how well they do it. In this project, the goal is to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways.

The training data for this project are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv)

The test data are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv)

The goal of this project is to predict the manner in which they did the exercise. This is the “classe” variable in the training set. We may use any of the other variables to predict with. This report describes how the model was built, how it was cross validated, the expected out of sample error etc. This model was used to predict 20 different test cases and the results were submitted online separately.

Data Pre-processing

Load the training and testing data set and check their dimensions.

```
trainingDat <- read.csv("pml-training.csv", na.strings=c("NA", "", "NULL"))
dim(trainingDat)
```

```
## [1] 19622   160
```

```
testingDat <- read.csv("pml-testing.csv", na.strings=c("NA", "", "NULL"))
dim(testingDat)
```

```
## [1]  20 160
```

Ignore the variables with NA values as also the irrelevant variables like user name etc. from further analysis.

```
training.noNA <- trainingDat[ , colSums(is.na(trainingDat)) == 0]
dim(training.noNA)
```

```
## [1] 19622    60
```

```
unwanted <- c('X', 'user_name', 'raw_timestamp_part_1', 'raw_timestamp_part_2', 'cvtd_t
imestamp', 'new_window', 'num_window')
training.trim <- training.noNA[, -which(names(training.noNA) %in% unwanted)]
dim(training.trim)
```

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

Remove the variables with near zero variance.

```
library(caret)
```

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

```
zeroVar <- nearZeroVar(training.trim[sapply(training.trim, is.numeric)], saveMetrics =
TRUE)
training.nzv <- training.trim[,zeroVar[, 'nzv']==0]
dim(training.nzv)
```

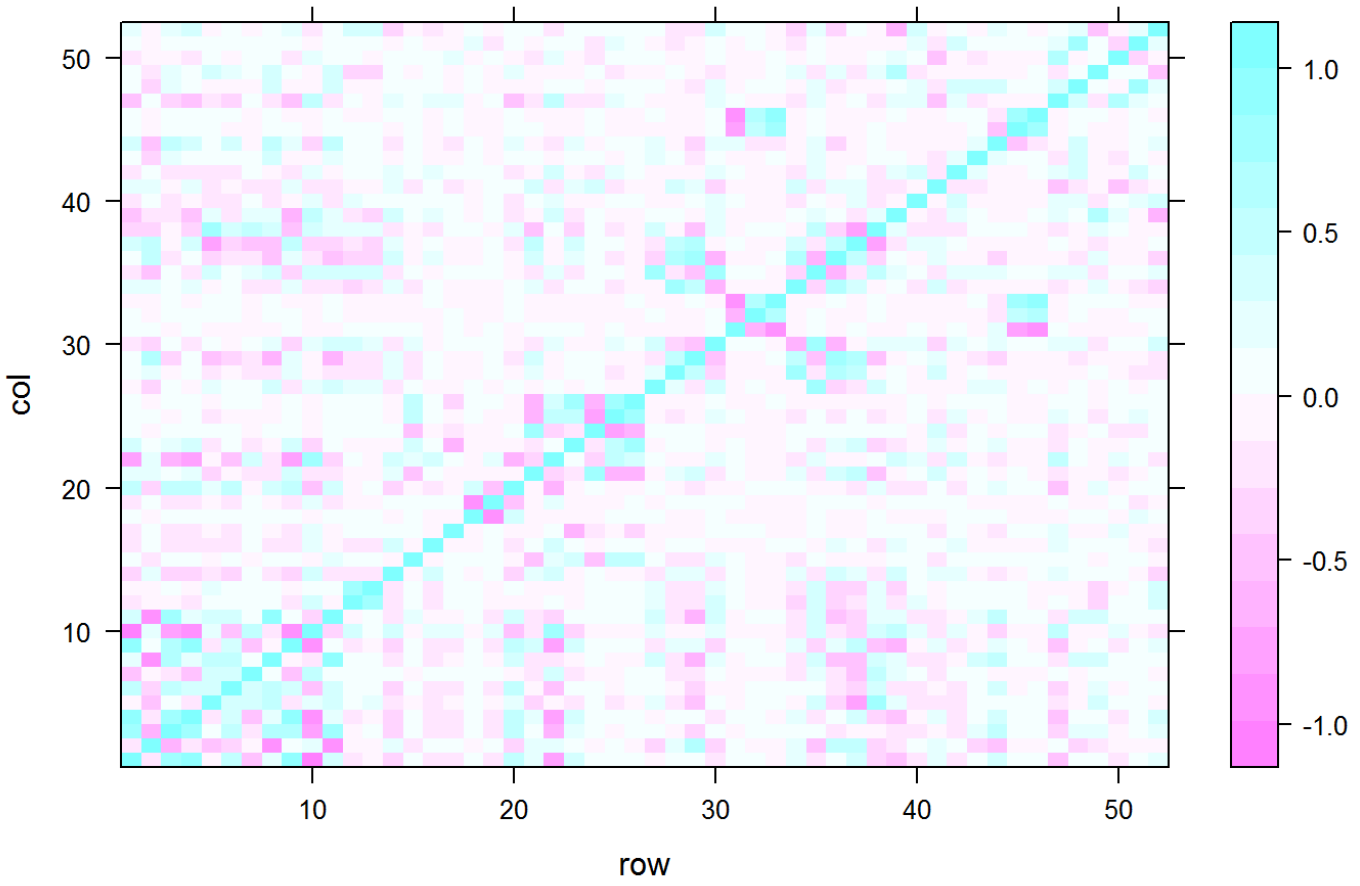
```
## [1] 19622    53
```

Remove the variables that are highly correlated with each other.

```
corrMat <- cor(na.omit(training.nzv[sapply(training.nzv, is.numeric)]))
dim(corrMat)
```

```
## [1] 52 52
```

```
corrDat <- expand.grid(row = 1:52, col = 1:52)
corrDat$correlation <- as.vector(corrMat)
levelplot(correlation ~ row+ col, corrDat)
```




```
rmcor <- findCorrelation(corrMat, cutoff = .90, verbose = TRUE)
```

```
training.fin <- training.nzv[,-rmcor]
dim(training.fin)
```

```
## [1] 19622    46
```

For cross validation sake, split the data into training and testing data sets.

```
train.list <- createDataPartition(y=training.fin$classe, p=0.6, list=FALSE)
train.dat <- training.fin[train.list,]
test.dat <- training.fin[-train.list,]
dim(train.dat)
```

```
## [1] 11776    46
```

```
dim(test.dat)
```

```
## [1] 7846   46
```

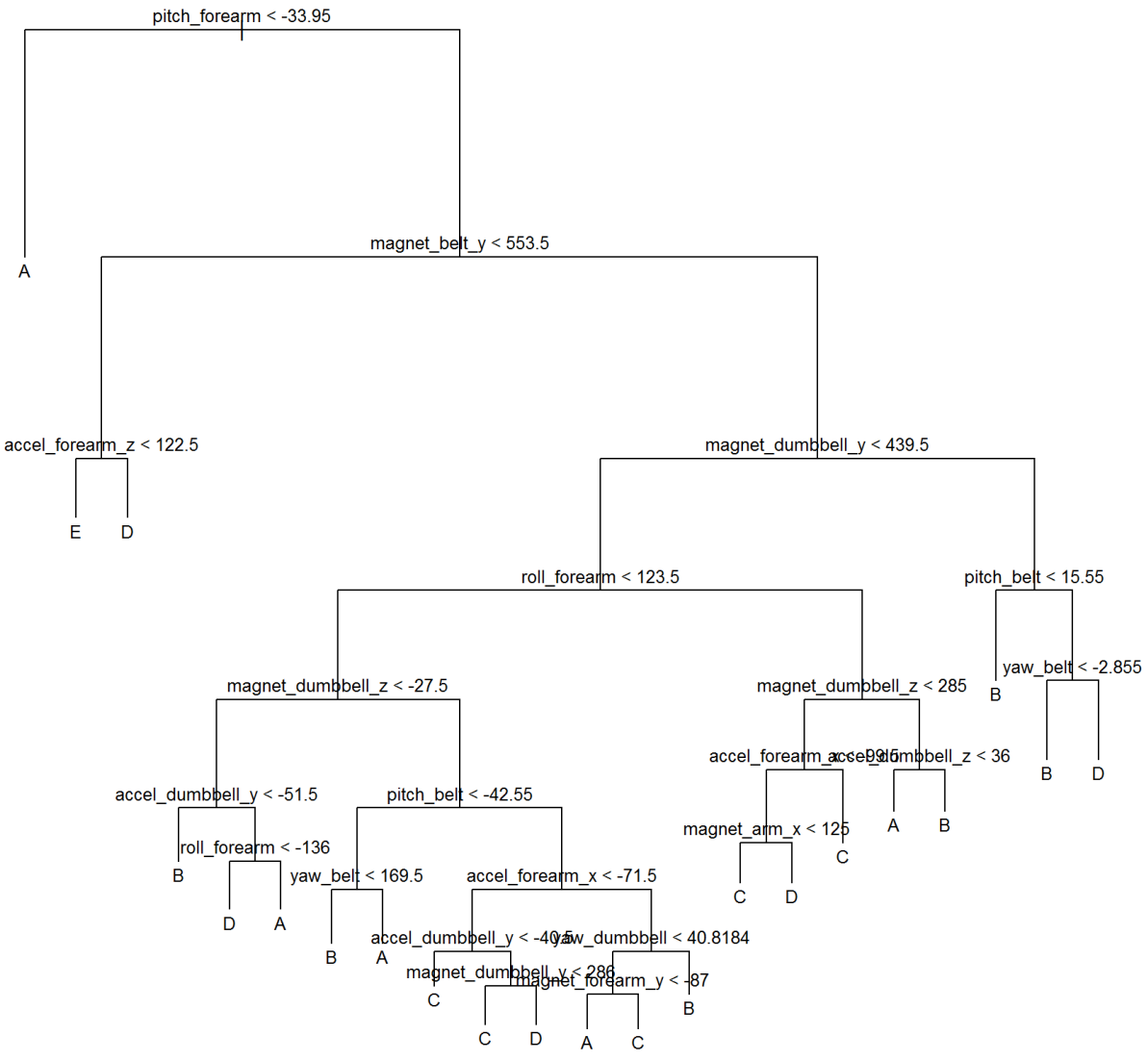
Data Analysis - Regression Tree & Random Forest models.

Fit a regression tree and plot it.

```
library(tree)
set.seed(123456)
tree.train <- tree(classe~.,data=train.dat)
summary(tree.train)
```

```
##
## Classification tree:
## tree(formula = classe ~ ., data = train.dat)
## Variables actually used in tree construction:
##  [1] "pitch_forearm"     "magnet_belt_y"     "accel_forearm_z"
##  [4] "magnet_dumbbell_y" "roll_forearm"      "magnet_dumbbell_z"
##  [7] "accel_dumbbell_y"  "pitch_belt"        "yaw_belt"
## [10] "accel_forearm_x"   "yaw_dumbbell"      "magnet_forearm_y"
## [13] "magnet_arm_x"      "accel_dumbbell_z"
## Number of terminal nodes:  22
## Residual mean deviance:  1.602 = 18830 / 11750
## Misclassification error rate: 0.3112 = 3665 / 11776
```

```
plot(tree.train)
text(tree.train,pretty=0, cex =.8)
```



Cross validate the above tree with the testing data.

```
tree.pred <- predict(tree.train,test.dat,type="class")
predMat <- with(test.dat,table(tree.pred,classe))
sum(diag(predMat))/sum(as.vector(predMat))
```

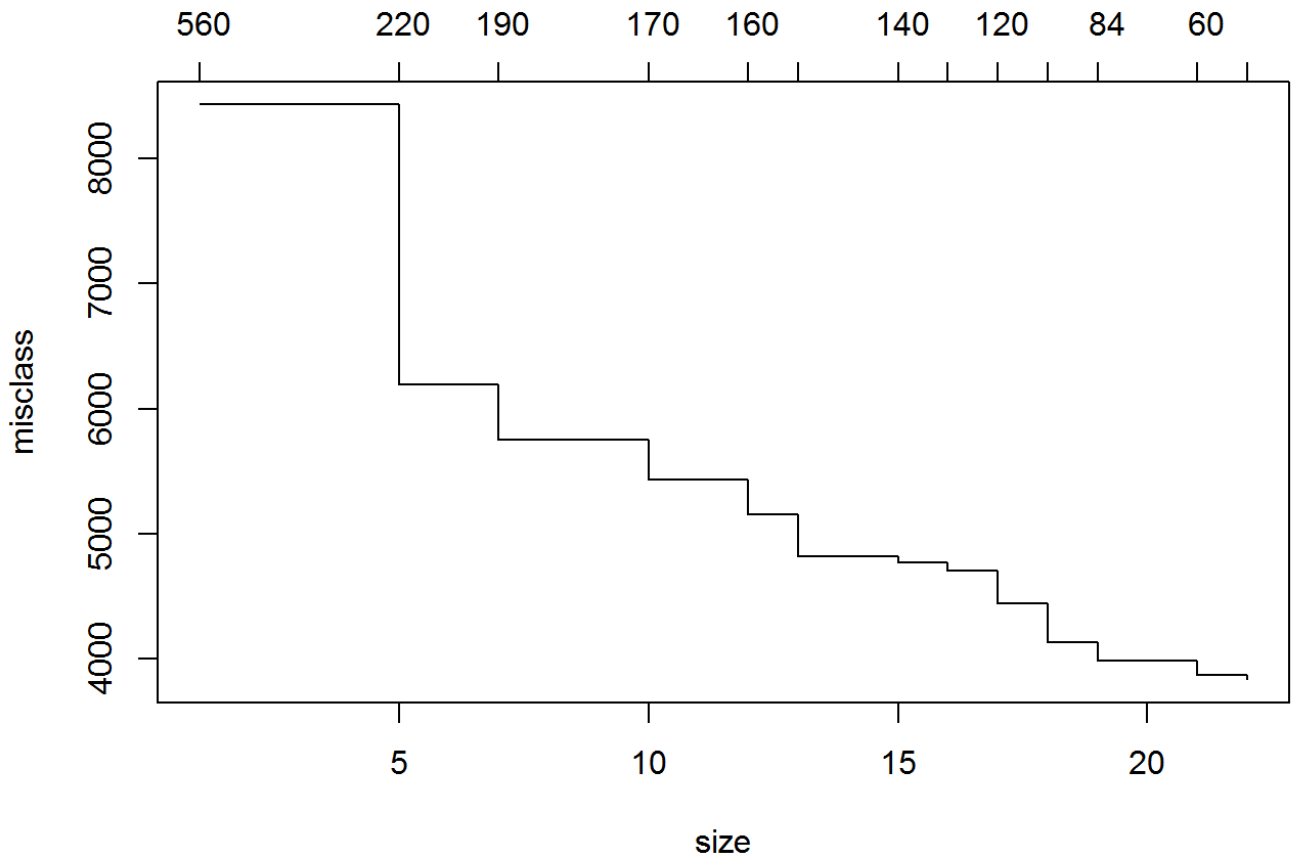
```
## [1] 0.6769054
```

The above tree probably has too much of variance. Hence the tree will be pruned using cross validation as detailed below.

```
cv.train <- cv.tree(tree.train,FUN=prune.misclass)
cv.train
```

```
## $size
##  [1] 22 21 19 18 17 16 15 13 12 10  7  5  1
##
## $dev
##  [1] 3839 3875 3987 4136 4448 4707 4775 4824 5156 5435 5749 6193 8428
##
## $k
##  [1]     -Inf  60.0000  84.5000 107.0000 123.0000 132.0000 141.0000
##  [8] 145.0000 157.0000 170.0000 191.3333 223.5000 555.7500
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

```
plot(cv.train)
```




```
prune.train <- prune.misclass(tree.train,best=15)

tree.pred <- predict(prune.train,test.dat,type="class")
predMat <- with(test.dat,table(tree.pred,classe))
sum(diag(predMat))/sum(as.vector(predMat))
```

```
## [1] 0.6176396
```

The above result shows that the model has become simpler due to pruning while retaining almost earlier accuracy. A shallower tree is easier to interpret but often has a lower accuracy. To improve the accuracy we will construct random forest model wherein many individual trees are built and averaged out to reduce the variance and enhance the accuracy.

```
require(randomForest)
```

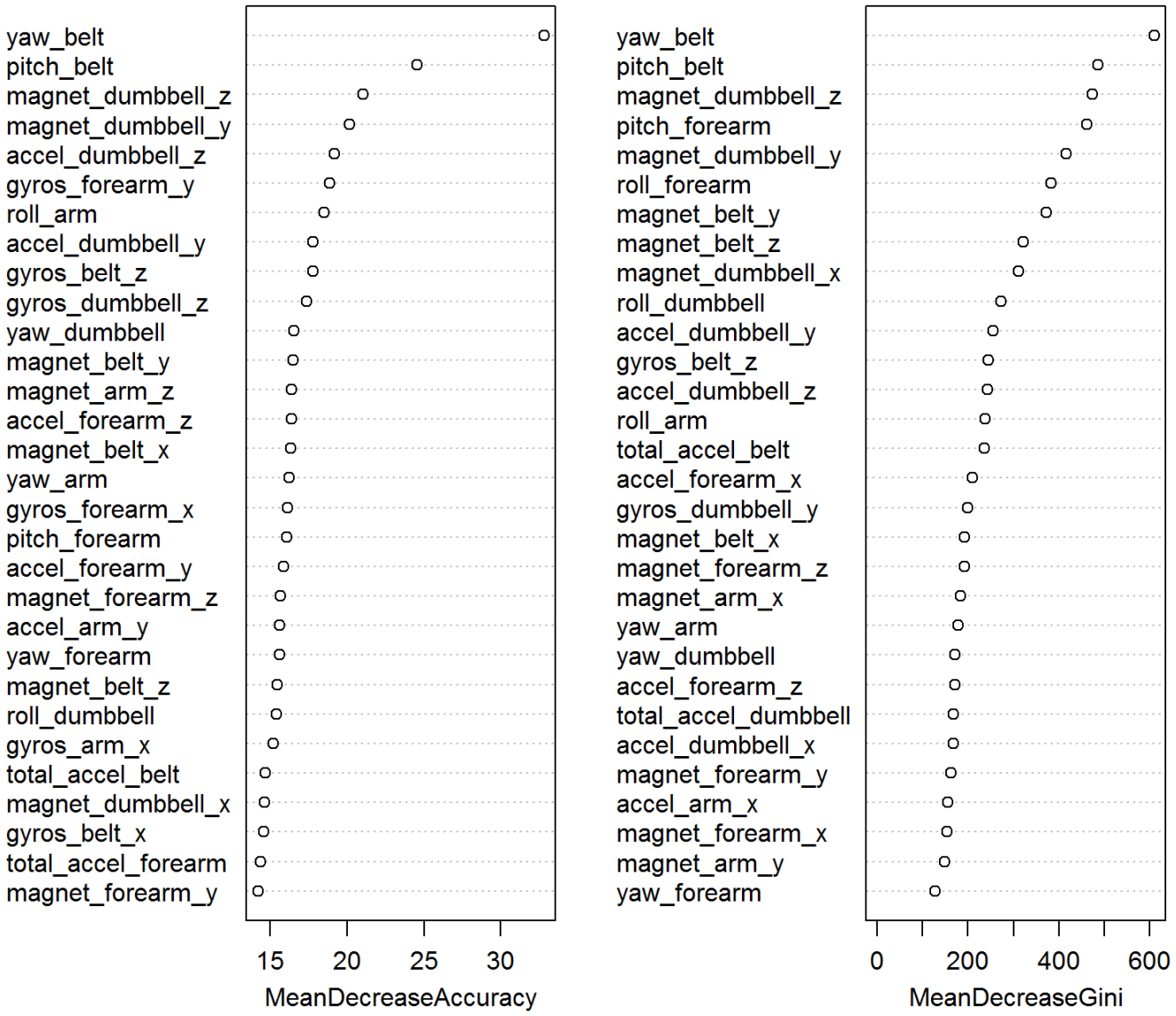
```
## Loading required package: randomForest
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

```
set.seed(123456)
rf.train <- randomForest(classe~.,data=train.dat, ntree=100, importance=TRUE)
rf.train
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = train.dat, ntree = 100,      importance =
 TRUE)
##                Type of random forest: classification
##                      Number of trees: 100
## No. of variables tried at each split: 6
##
##         OOB estimate of  error rate: 0.94%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3344    4    0    0    0 0.001194743
## B   20 2245   13    0    1 0.014918824
## C    0   27 2021    6    0 0.016066212
## D    0    0   25 1901    4 0.015025907
## E    0    1    3    7 2154 0.005080831
```

The variables of high importance are shown in the plots below.

```
varImpPlot(rf.train,)
```



Now we apply this random forest model on the test data and compute the out of sample accuracy.

```
tree.pred <- predict(rf.train,test.dat,type="class")
predMat <- with(test.dat,table(tree.pred,classe))
sum(diag(predMat))/sum(as.vector(predMat))
```

```
## [1] 0.9922253
```

The above result shows that the random forest model is quite accurate. The predictions from this model were submitted online for the prediction assignment as illustrated below.

```
answers <- predict(rf.train, testingDat)
answers
```

```
##  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
```