14.06.2014 Assignment Writeup done for the Coursera course “Practical Machine Learning” of “Data Science” Specialization track. (Last Updated 22.06.2014)

Machine Learning for HAR (Human Activity Recognition)

Summary: This report is part of the course **Practical Machine Learning** offered by **Johns Hopkins University** on **Coursera**. Having data on personal acitivity of people (here weight lifting exercise) it is possible to predict the manner they used to exercise. To know about the dataset and variables visit the site and look for paragraph **Weight Lifting Exercises Dataset**.

The files were downloaded from the course web site (Practical Machine Learning: Course Project: Writeup):

- Training Set
- Test Set

The report consists of three parts:

- Data Processing and Filtering
- Variable Selection
- Model Building

Data Processing and Filtering

Loading the needed Libraries

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.0.3
```

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

```
## Warning: package 'lattice' was built under R version 3.0.3
```

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

```
## Warning: package 'ggplot2' was built under R version 3.0.3
```

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.0.3
```

```
## randomForest 4.6-7
## Type rfNews() to see new features/changes/bug fixes.
```

```
library(ggplot2)
```

Loading the Data

```
TrainingSet <- read.csv("pml-training.csv", header=T, sep=",")
TestingSet <- read.csv("pml-testing.csv", header=T, sep=",")
```

Check training and test set for missing Data

```
TestCount <- apply(is.na(TestingSet), 2, sum)
TestRatio <- 100*TestCount/nrow(TestingSet)
TestResults <- data.frame(cbind(TestCount, TestRatio))
TestNames <- rownames(TestResults[TestResults$TestRatio>90,])
head(TestNames)
```

```
## [1] "kurtosis_roll_belt"   "kurtosis_picth_belt"  "kurtosis_yaw_belt"
## [4] "skewness_roll_belt"   "skewness_roll_belt.1" "skewness_yaw_belt"
```

Based on the analysis, we use only predictors with more than 90% completeness.

```
TrainCount <- apply(is.na(TrainingSet), 2, sum)
TrainRatio <- 100*TrainCount/nrow(TrainingSet)
TrainResults <- data.frame(cbind(TrainCount, TrainRatio))
TrainNames <- rownames(TrainResults[TrainResults$TrainRatio>95,])
Names <- union(TrainNames, TestNames)
head(Names)
```

```
## [1] "max_roll_belt"        "max_picth_belt"       "min_roll_belt"
## [4] "min_pitch_belt"       "amplitude_roll_belt"  "amplitude_pitch_belt"
```

Based on the analysis, we use only predictors with more than 95% completeness.

Subset of Variables

```
TrainVars <- names(TrainingSet) %in% Names
TestVars <- names(TestingSet) %in% Names

NewTrainSet <- TrainingSet[!TrainVars]
NewTestSet <- TestingSet[!TestVars]
```

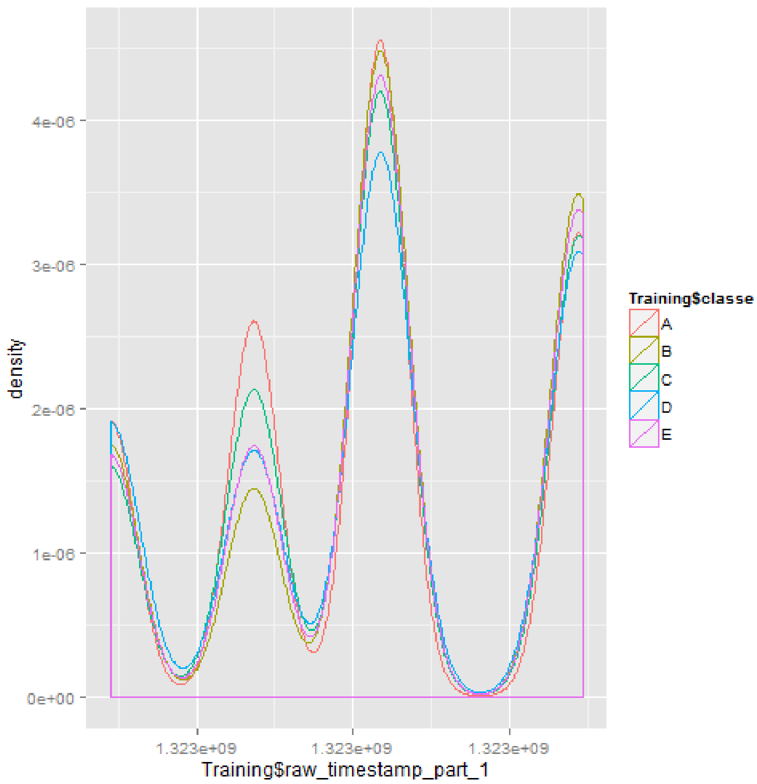
Remove variables with a value near zero

```
nzvTrain <- nearZeroVar(NewTrainSet, saveMetrics = FALSE)
nzvTest <- nearZeroVar(NewTestSet, saveMetrics = FALSE)
Training <- NewTrainSet[,-nzvTrain]
Testing <- NewTestSet[,-nzvTest]
```

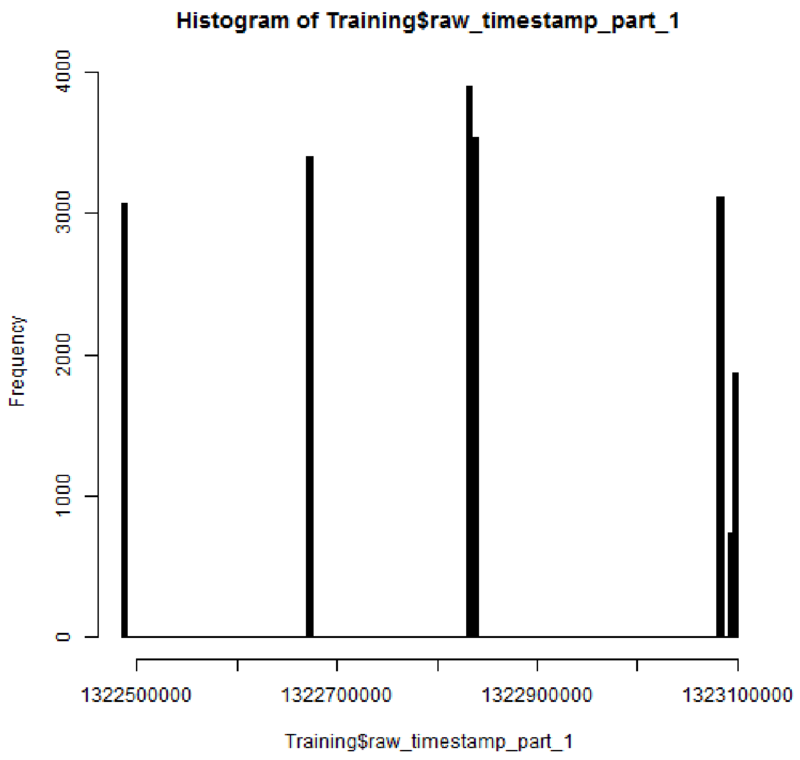
Variable Selection

Exploratory Data Analysis

```
qplot(Training$raw_timestamp_part_1, col=Training$classe, geom = "density")
```




```
hist(Training$raw_timestamp_part_1, breaks=200, col="black")
```



Combine Data

```
Training$Mark <- "Train"
Testing$Mark <- "Test"
colnames(Testing) <- colnames(Training)
Data <- rbind(Training, Testing)
```

```
## Warning: invalid factor level, NA generated
```

```
Data <- Data[,2:ncol(Data)]

abandonNameList <- c("raw_timestamp_part_2")
```

Preparation

```
Data$raw_timestamp_part_1 <- Data$raw_timestamp_part_1/100000
Combine <- cut(Data$raw_timestamp_part_1, c(min(Data$raw_timestamp_part_1), 13226,
13227,13229, max(Data$raw_timestamp_part_1)), order=T)
```

Steps needed for Random Forest

```
Data$RTSP1 <- Combine
abandonNameList <- union(abandonNameList, "raw_timestamp_part_1")
```

Categories (2): “roll_belt”

```
Data$rollBeltCut <- cut(Data$roll_belt,c(min(Data$roll_belt), 72, max(Data$roll_belt)))
abandonNameList <- union(abandonNameList, "roll_belt")
```

Categories (4): “pitch_belt”

```
Data$pitchBeltCut <- cut(Data$pitch_belt, c(min(Data$pitch_belt), -18, 12, 20,
max(Data$pitch_belt)))
abandonNameList <- union(abandonNameList, "pitch_belt")
```

Categories (3): “yaw_belt”

```
Data$yawBeltCut <- cut(Data$yaw_belt, c(min(Data$yaw_belt), -45, 90, max(Data$yaw_belt)))
abandonNameList <- union(abandonNameList, "yaw_belt")
```

```
abandonNameList <- union(abandonNameList, "yaw_belt")
```

Categories (2): "total_accel_belt"

```
Data$totalAccelBeltCut <- cut(Data$total_accel_belt, c(min(Data$total_accel_belt), 12,
max(Data$total_accel_belt)))
abandonNameList <- union(abandonNameList, "total_accel_belt")
```

Categories (2): "accel_belt_x"

```
Data$accelBeltxCut <- cut(Data$accel_belt_x, c(min(Data$accel_belt_x), 25,
max(Data$accel_belt_x)))
abandonNameList <- union(abandonNameList, "accel_belt_x")
```

Categories (1): "magnet_dumbbell_x"

```
Data$magnetDumbbellxCut <- cut(Data$magnet_dumbbell_x, c(min(Data$magnet_dumbbell_x), 0,
max(Data$magnet_dumbbell_x)))
abandonNameList <- union(abandonNameList, "magnet_dumbbell_x")
```

Categories (1): "magnet_dumbbell_y"

```
Data$magnetDumbbellyCut <- cut(Data$magnet_dumbbell_y, c(min(Data$magnet_dumbbell_y), -100,
max(Data$magnet_dumbbell_y)))
abandonNameList <- union(abandonNameList, "magnet_dumbbell_y")
```

Categories (1): "magnet_dumbbell_z"

```
Data$magnetDumbbellzCut <- cut(Data$magnet_dumbbell_z, c(min(Data$magnet_dumbbell_z), 200,
max(Data$magnet_dumbbell_z)))
abandonNameList <- union(abandonNameList, "magnet_dumbbell_z")
myvars <- names(Data) %in% abandonNameList
TotalData <- Data[!myvars]
```

Final Set for Model Building

```
training <- subset(TotalData, TotalData$Mark=="Train", select=-Mark)
training <- na.omit(training)
testing <- subset(TotalData, TotalData$Mark=="Test", select=-Mark)
PredictorNames <- names(training)[-48]
```

Model Building

Build Random Forest Model

```
Model <- randomForest(training[,PredictorNames],training[,"classe"], importance=T,
ntree=100)
```

Show the Random Forest Model

```
print(Model)
```

```
##
## Call:
##  randomForest(x = training[, PredictorNames], y = training[, "classe"],      ntree =
100, importance = T)
##                Type of random forest: classification
##                      Number of trees: 100
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.23%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 5575    1    1    0    0   0.0003586
## B    4 3789    1    0    0   0.0013179
## C    0   10 3403    8    0   0.0052616
## D    0    0   16 3198    1   0.0052877
## E    0    0    0    4 3598   0.0011105
```

Estimated Error

- The estimated error of the model is only 0.24%.
- I predicted all 20 examples correctly.

Cross-Validation

- I cite **Leo Breiman** and **Adele Cutler** from Berkeley: There is no need to carry out cross-validation as they say here:

> 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, as follows:
>
> Each tree is constructed using a different bootstrap sample from the original data. About one-third of the cases are left out of the bootstrap sample and not used in the construction of the kth tree.
>
> Put each case left out in the construction of the kth tree down the kth tree to get a classification. In this way, a test set classification is obtained for each case in about one-third of the trees. At the end of the run, take j to be the class that got most of the votes every time case n was oob. The proportion of times that j is not equal to the true class of n averaged over all cases is the oob error estimate. This has proven to be unbiased in many tests.