Data Analysis for Weight Lifting Excercise Data set

Abstarct

The goal of this project is to predict the manner in which the weight lifting exercise is done.To accomplish the goal of the project, first the relevent data ia downloaded and red. Then, the data is cleaned to remove measuremnets with largely missed or no measuremnets. Then, the cleaned data is partitioned to training and testing parts to cross validate different machine learning algorithms. Various machine learning algorithms such as random forest and decision tree aare implemented on the data set. The out of bag error is estimated using the validation data set. These steps are explained here:

reading data

first step is to read the data and acquired the required R pachages.

```
f_d <- getwd()
dt_training <- read.table(paste0(f_d,"/pml-training.csv"),sep = ",",header=T,na.strings=c("NA","#DIV/0!
dt_testing <- read.table(paste0(f_d,"/pml-testing.csv"),sep = ",",header=T,na.strings=c("NA","#DIV/0!",

summary(dt_training)

# required libraries
library(caret)
library(ggplot2)
library(caret)
library(randomForest)
library(e1071)
library(gbm)
library(survival)
library(splines)
library(plyr)
library(rpart)
library(rpart.plot)
library(rattle)
library(RColorBrewer)
```

cleaning data

The, clean the data with removing the measurements with high number of NA's or near zero variance.

```
dt_training <- dt_training[,3:160]

v_na <- numeric()

for (i in 1:dim(dt_training)[2]) {

  if (sum(is.na(dt_training[,i]))>0.95*dim(dt_training)[1]) {v_na=c(v_na,i)}
}

dt_training <- dt_training[,-v_na]
dim(dt_training)
```

```
## [1] 19622    58
```

```
#dt_training <- dt_tr[complete.cases(dt_training),]

v_zv <- nearZeroVar(dt_training, saveMetrics = TRUE)

dt_training <- dt_training[, v_zv$nzv==FALSE]

dt_training$classe = factor(dt_training$classe)

dim(dt_training)
```

```
## [1] 19622    57
```

```
dt_testing <- dt_testing[, 3:160]
dt_testing <- dt_testing[, -v_na]
dt_testing <- dt_testing[, v_zv$nzv==FALSE]

dim(dt_testing)
```

```
## [1] 20 57
```

partitioning data

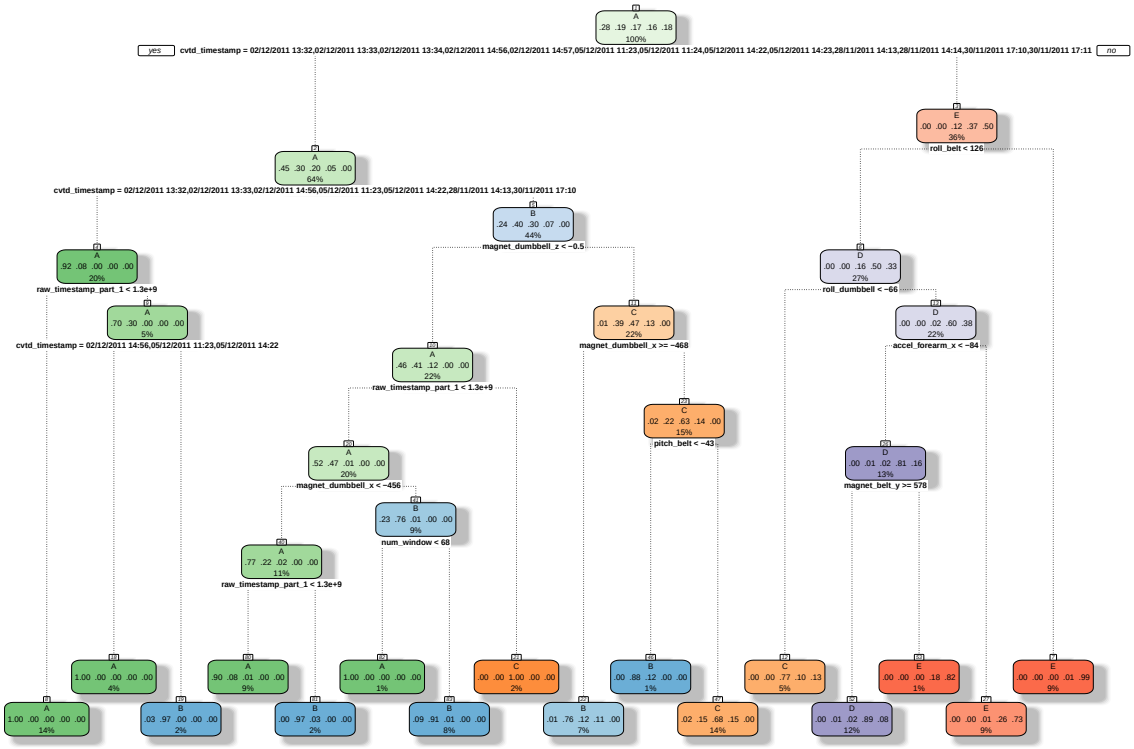
The, the clead data is partitioned to two sets one for training and one for validation to estimated the error.

```
inTrain = createDataPartition(dt_training$classe, p = .6)[[1]]

dt_training1 =dt_training[ inTrain,]

dt_cv1 = dt_training[-inTrain,]
```

Implementing ML algorithms

Two ML algorithms including random forest and decision tree are implemented.

```
set.seed(200)
mod1 <- rpart(classe ~ ., data=dt_training1, method="class")

fancyRpartPlot(mod1)
```



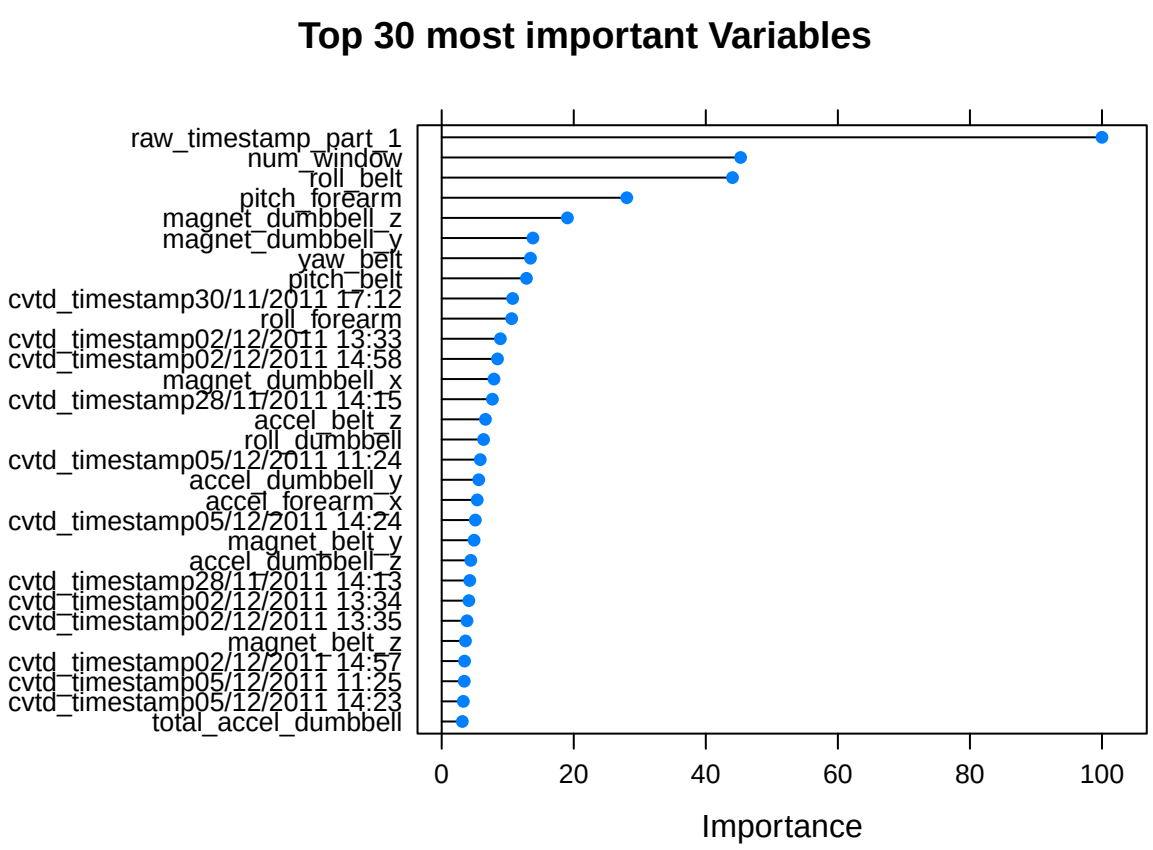
Rattle 2018−Dec−26 18:00:14 meymand.sajjad

```
confusionMatrix(predict(mod1,newdata=dt_cv1, type="class"),dt_cv1$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2146   66    5    3    0
##          B   65 1251   85   70    0
##          C   21  190 1257  186   58
##          D    0   11    8  816   87
##          E    0    0   13  211 1297
##
## Overall Statistics
##
##                Accuracy : 0.8625
##                  95% CI : (0.8547, 0.87)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.826
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9615   0.8241   0.9189   0.6345   0.8994
```

```
## Specificity            0.9868   0.9652   0.9298   0.9838   0.9650
## Pos Pred Value         0.9667   0.8504   0.7342   0.8850   0.8527
## Neg Pred Value         0.9847   0.9581   0.9819   0.9321   0.9771
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2735   0.1594   0.1602   0.1040   0.1653
## Detection Prevalence   0.2829   0.1875   0.2182   0.1175   0.1939
## Balanced Accuracy      0.9741   0.8947   0.9243   0.8092   0.9322
```

```
set.seed(200)
mod2 <- train(classe ~ ., data=dt_training1, method="rf")

v_IO <- varImp(mod2)

plot(v_IO, main = "Top 30 most important Variables", top = 30)
```




```
confusionMatrix(predict(mod2,newdata=dt_cv1),dt_cv1$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2232    2    0    0    0
##          B    0 1516    2    0    0
##          C    0    0 1366    0    0
##          D    0    0    0 1286    2
##          E    0    0    0    0 1440
##
```

```
## Overall Statistics
##
##                Accuracy : 0.9992
##                  95% CI : (0.9983, 0.9997)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.999
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9987   0.9985   1.0000   0.9986
## Specificity            0.9996   0.9997   1.0000   0.9997   1.0000
## Pos Pred Value         0.9991   0.9987   1.0000   0.9984   1.0000
## Neg Pred Value         1.0000   0.9997   0.9997   1.0000   0.9997
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1932   0.1741   0.1639   0.1835
## Detection Prevalence   0.2847   0.1935   0.1741   0.1642   0.1835
## Balanced Accuracy      0.9998   0.9992   0.9993   0.9998   0.9993
```

```
mod1$finalModel
```

```
## NULL
```

```
mod2$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 38
##
##         OOB estimate of  error rate: 0.1%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3348    0    0    0    0 0.0000000000
## B    1 2275    3    0    0 0.0017551558
## C    0    2 2051    1    0 0.0014605648
## D    0    0    2 1926    2 0.0020725389
## E    0    0    0    1 2164 0.0004618938
```

Error calculation

The accuracy level of decision tree is about 87% and for random forest is about 99%. As the random forest algorithm's accuracy level suggest, this method work very well for predicting the outcome of the validation set.

predictions

Here is the predict outcome levels on the original Testing data set using Random Forest algorithm that generated the lowest error rate.

```
predict_testset <- predict.train(mod2, dt_testing)
predict_testset
```

```
##  [1] 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

The Random Forest method yielded better results. The Confusion Matrix achieved 99.9% accuracy. The Out of Sample Error achieved 99.7449 %. Since Random forests works well when there is a large number of inputs, especially when the interactions between variables are unknow and also can handle unscaled variables and categorical variables, Random Forest is selected to be used for the final predictions.