# ML Final Project

*Shuguang Ji*

*January 31, 2016*

## Objective

The goal of your project is to predict the manner in which they did the exercise. This is the "classe" variable in the training set. You may use any of the other variables to predict with. You should create a report describing how you built your model, how you used cross validation, what you think the expected out of sample error is, and why you made the choices you did. You will also use your prediction model to predict 20 different test cases.

### Step1 Install Packages

```
library(caret)
library(randomForest)
library(rpart)
library(rpart.plot)
library(RColorBrewer)
if (!require('rattle'))
{
  install.packages('rattle');
  library(rattle);
}
if (!require('e1071'))
{
  install.packages('e1071');
  library(e1071);
}

set.seed(1234)
```

### Step2 Load Data and Create Training and Testing Sets

In this analysis, 70% data are used for training and 30% are used for testing.

```
UrlTrain <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
UrlTest  <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

training <- read.csv(url(UrlTrain))
testing  <- read.csv(url(UrlTest))

inTrain <- createDataPartition(y=training$classe, p=0.7, list=FALSE)
myTraining <- training[inTrain, ]
myTesting <- training[-inTrain, ]
```

### Step3 Clean Data

The following codes are used to clean the data sets. Major cleaning process include: remove missing values, assign columns' names, dimension reduction, data transformation, and data coerce.

```
myDataNZV <- nearZeroVar(myTraining, saveMetrics=TRUE)
myNZVvars <- names(myTraining) %in% c("new_window", "kurtosis_roll_belt", "kurtosis_picth_belt",
                                      "kurtosis_yaw_belt", "skewness_roll_belt", "skewness_roll_belt.1",
                                      "max_yaw_belt", "min_yaw_belt", "amplitude_yaw_belt", "avg_roll_arm
                                      "var_roll_arm", "avg_pitch_arm", "stddev_pitch_arm", "var_pitch_arm
                                      "stddev_yaw_arm", "var_yaw_arm", "kurtosis_roll_arm", "kurtosis_pic
                                      "kurtosis_yaw_arm", "skewness_roll_arm", "skewness_pitch_arm", "skew
                                      "max_roll_arm", "min_roll_arm", "min_pitch_arm", "amplitude_roll_ar
                                      "kurtosis_roll_dumbbell", "kurtosis_picth_dumbbell", "kurtosis_yaw_
                                      "skewness_pitch_dumbbell", "skewness_yaw_dumbbell", "max_yaw_dumbbe
                                      "amplitude_yaw_dumbbell", "kurtosis_roll_forearm", "kurtosis_picth_
                                      "skewness_roll_forearm", "skewness_pitch_forearm", "skewness_yaw_fo
                                      "max_yaw_forearm", "min_roll_forearm", "min_yaw_forearm", "amplitud
                                      "amplitude_yaw_forearm", "avg_roll_forearm", "stddev_roll_forearm",
                                      "avg_pitch_forearm", "stddev_pitch_forearm", "var_pitch_forearm", "a
                                      "stddev_yaw_forearm", "var_yaw_forearm")
myTraining <- myTraining[!myNZVvars]
myTraining <- myTraining[c(-1)]
temp <- myTraining
for(i in 1:length(myTraining)) {
  if( sum( is.na( myTraining[, i] ) ) /nrow(myTraining) >= .6 ) {
    for(j in 1:length(temp)) {
      if( length( grep(names(myTraining[i]), names(temp)[j]) ) ==1)  {
        temp <- temp[ , -j]
      }
    }
  }
}

myTraining <- temp
rm(temp)

clean1 <- colnames(myTraining)
clean2 <- colnames(myTraining[, -58])
myTesting <- myTesting[clean1]
testing <- testing[clean2]

for (i in 1:length(testing) ) {
  for(j in 1:length(myTraining)) {
    if( length( grep(names(myTraining[i]), names(testing)[j]) ) ==1)  {
      class(testing[j]) <- class(myTraining[i])
    }
  }
}

testing <- rbind(myTraining[2, -58] , testing)
testing <- testing[-1,]
```

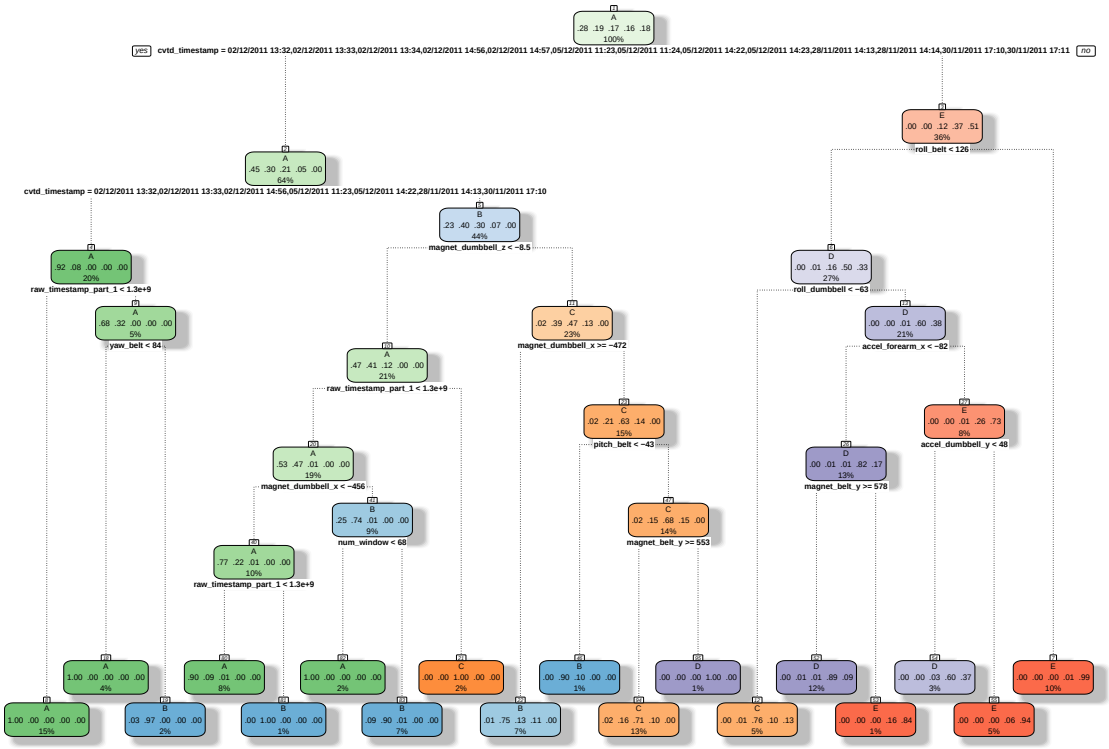
## Step4 Marchine Learning

In this step, the cleaned data sets will be used to train the models and do predict.

### Decision Tree Model

Following plot is the results from running decision tree model on traning data set.

```
mod.dt <- rpart(classe ~ ., data=myTraining, method="class")
fancyRpartPlot(mod.dt)
```




Rattle 2016−Feb−01 00:20:41 User

The accuracy of decision tree model is shown below.

```
predictions.dt <- predict(mod.dt, myTesting, type = "class")
confusionMatrix(predictions.dt, myTesting$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1606   53    3    0    0
##          B   40  947   71   50    0
##          C   28  130  930  108   49
##          D    0    9   22  752  133
##          E    0    0    0   54  900
```

```
##
## Overall Statistics
##
##                Accuracy : 0.8726
##                  95% CI : (0.8638, 0.881)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8389
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9594   0.8314   0.9064   0.7801   0.8318
## Specificity            0.9867   0.9661   0.9352   0.9667   0.9888
## Pos Pred Value         0.9663   0.8547   0.7470   0.8210   0.9434
## Neg Pred Value         0.9839   0.9598   0.9793   0.9573   0.9631
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2729   0.1609   0.1580   0.1278   0.1529
## Detection Prevalence   0.2824   0.1883   0.2116   0.1556   0.1621
## Balanced Accuracy      0.9730   0.8988   0.9208   0.8734   0.9103
```

### Random Forest Model

Following plot is the results from running random forest model on traning data set.

```
mod.rf <- randomForest(classe ~. , data=myTraining)
```

```
predictions.rf <- predict(mod.rf, myTesting, type = "class")
confusionMatrix(predictions.rf, myTesting$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    2    0    0    0
##          B    0 1137    1    0    0
##          C    0    0 1023    2    0
##          D    0    0    2  962    0
##          E    0    0    0    0 1082
##
## Overall Statistics
##
##                Accuracy : 0.9988
##                  95% CI : (0.9976, 0.9995)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9985
##  Mcnemar's Test P-Value : NA
##
```

```
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9982   0.9971   0.9979   1.0000
## Specificity            0.9995   0.9998   0.9996   0.9996   1.0000
## Pos Pred Value         0.9988   0.9991   0.9980   0.9979   1.0000
## Neg Pred Value         1.0000   0.9996   0.9994   0.9996   1.0000
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1932   0.1738   0.1635   0.1839
## Detection Prevalence   0.2848   0.1934   0.1742   0.1638   0.1839
## Balanced Accuracy      0.9998   0.9990   0.9983   0.9988   1.0000
```

By comparing the performance of decision tree model and random forest model, we notice that random forest model outperformed. Thus, random forest model is selected to do predcit fro 20 test cases.

### 20 Test Cases

The results of applying randome forest algorithm to the 20 test cases are show below.

```
predictions.20 <- predict(mod.rf, testing, type = "class")
predictions.20
```

```
##  2 31  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21
##  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
```