To Predict the Manner of Exercise

Keerthi Arun

August 20, 2015

Our aim is to create a prediction model that can be used to predict the manner in which someone did their exercise. pml-training csv file is used for creating the model. This file can be downloaded from here. The classe variable in this training set is a categorical variable quatifying the manner in which our six participants exercised. To Load the data used for creating the prediction model-

```
pml_training_set <- read.csv( "pml-training.csv")
```

Preprocessing the training data

We are only interested in those variables that can be used to make an accurate prediction. Starting by removing all those variables that shows near to zero variablity in thier observations.

```
library(caret)
```

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

```
near0 <- nearZeroVar(pml_training_set)
pml_training_set <- pml_training_set[,- near0]
summary(pml_training_set)
```

From the summary(Refer Appendix for summary(pml_training_set)), the column X is simply the index of datatable which can be omitted and so can be cvtd_timestamp, can be obtained from columns raw_timestamp_part_1 and raw_timestamp_part_2. we can also remove the column user_name.

```
rm_index <- which(names(pml_training_set)== 'X'|names(pml_training_set)== 'cvtd_timestamp'|names(pml_tra
pml_training_set <- pml_training_set[,- rm_index]
```

Further looking through the summary, you will notice that there are some categorical variables ,like max_roll_belt, max_picth_arm and many more, that have equally large number of NAs(=19216). On further investigation it is clear that all these NA values are for the same observations. If a variable has large number of missing values compared to its non-missing values, it is better to ignore it from prediction models to avoid incorrect predictions.

```
NA_index <- apply(pml_training_set,MARGIN = 2,function(x){sum(is.na(x))})
pml_training_set<- pml_training_set[,which(NA_index!=19216)]
```

Regression Modelling

Now let's divide pml_training_set into a training & testing so that later we can use testing dataset to crossvalidate our prediction model.

```
library(caret)
set.seed(100)
inTrain  <- createDataPartition(y = pml_training_set$classe,p = .7,list = FALSE)
training <- pml_training_set[inTrain,]
testing  <- pml_training_set[-inTrain,]
```

Considering non-linearity of variable values we are using regression trees for easier prediction. First, let's use Rpart package.

```
library(rpart)
modelfit1 <- train(form = classe~.,data = training, method ="rpart")
modelfit1$results
```

```
##           cp  Accuracy     Kappa AccuracySD    KappaSD
## 1 0.03850066 0.5389797 0.4043885 0.06128082 0.09799232
## 2 0.05719154 0.4467046 0.2560096 0.07220933 0.11816883
## 3 0.11728207 0.3377956 0.0798228 0.04070713 0.06113674
```

```
max(modelfit1$results$Accuracy)*100
```

```
## [1] 53.89797
```

Now, lets try the same with a different tree package - Random Forest package.

```
modelfit2 <- train(form = classe~.,data = training, method = "rf", trControl = trainControl(method = "c
```

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

```
modelfit2$results
```

```
##   mtry  Accuracy     Kappa   AccuracySD      KappaSD
## 1    2 0.9946859 0.9932779 0.0003332413 0.0004220048
## 2   28 0.9983257 0.9978823 0.0004545505 0.0005749243
## 3   55 0.9952683 0.9940146 0.0005491384 0.0006947912
```

```
max(modelfit2$results$Accuracy)*100
```

```
## [1] 99.83257
```

Though randomforest method is time consuming than the other, it clearly gives a much better accuracy and accuracy standard deviation. Now, let's test both models on testing dataset.

Crossvalidation

```
library(rpart)
pred1 <- predict(modelfit1,testing)
table(pred1,testing$classe)
```

```
##
## pred1    A    B    C    D    E
##     A 1500  512  504  413  113
##     B   26  360   35  168   78
##     C  118  267  487  334  242
##     D    0    0    0    0    0
##     E   30    0    0   49  649
```

From the table above, modelfit1 is not a good prediction model. NOt only there is a large number of cases predicted wrong, but also classe D is never predicted properly at all. Accuracy of this model calculated was 53.9,whereas modelfit2 gave an accuracy value equal to 99.8. Now, trying next model-

```
library(randomForest)
pred2 <- predict(modelfit2,testing)
table(pred2,testing$classe)
```

```
##
## pred2    A    B    C    D    E
##     A 1674    2    0    0    0
##     B    0 1137    0    0    0
##     C    0    0 1026    5    0
##     D    0    0    0  959    0
##     E    0    0    0    0 1082
```

As expected, this model is much more accurate. Out of sample error for the model is the percentage of number of correct predictions in total number of predictions made.

```
False_pred <- pred2 != testing$classe # Creating a logical vector for identifying incorrect predictions
sum(False_pred)/sum(table(pred2,testing$classe)) #Out of sample error for training set
```

```
## [1] 0.001189465
```

The above value is specific to that data used. To find **expected out of sample error**, let's use confusionMatrix function. This function gives associated statistical values using which we can calculate expected out of sample error.

```
cm <- confusionMatrix(data = pred2,reference = testing$classe)
cm$byClass
```

```
##          Sensitivity Specificity Pos Pred Value Neg Pred Value Prevalence
## Class: A   1.0000000   0.9995251      0.9988067      1.0000000  0.2844520
## Class: B   0.9982441   1.0000000      1.0000000      0.9995788  0.1935429
## Class: C   1.0000000   0.9989710      0.9951503      1.0000000  0.1743415
## Class: D   0.9948133   1.0000000      1.0000000      0.9989850  0.1638063
## Class: E   1.0000000   1.0000000      1.0000000      1.0000000  0.1838573
##          Detection Rate Detection Prevalence Balanced Accuracy
```

```
## Class: A      0.2844520           0.2847918         0.9997625
## Class: B      0.1932031           0.1932031         0.9991220
## Class: C      0.1743415           0.1751912         0.9994855
## Class: D      0.1629567           0.1629567         0.9974066
## Class: E      0.1838573           0.1838573         1.0000000
```

cm$byClass object has eight columns and five rows. Balanced Accuracy is the the expected accuracy of the model. Therefore inaccuracy of each classe can be calculated by subtracting accuracy from 1. Now sum of all inaccuracy values divided by total number of inaccuracy values gives the 'Expected Out of Sample Error'.

```
sum(1 - cm$byClass[,8])/nrow(cm$byClass) # Expected Out of Sample Error
```

```
## [1] 0.0008446613
```

Appendix

summary(pml_training_set)

```
##  raw_timestamp_part_1 raw_timestamp_part_2   num_window
##  Min.   :1.322e+09    Min.   :   294       Min.   :  1.0
##  1st Qu.:1.323e+09    1st Qu.:252912       1st Qu.:222.0
##  Median :1.323e+09    Median :496380       Median :424.0
##  Mean   :1.323e+09    Mean   :500656       Mean   :430.6
##  3rd Qu.:1.323e+09    3rd Qu.:751891       3rd Qu.:644.0
##  Max.   :1.323e+09    Max.   :998801       Max.   :864.0
##    roll_belt        pitch_belt          yaw_belt       total_accel_belt
##  Min.   :-28.90   Min.   :-55.8000   Min.   :-180.00   Min.   : 0.00
##  1st Qu.:  1.10   1st Qu.:  1.7600   1st Qu.: -88.30   1st Qu.: 3.00
##  Median :113.00   Median :  5.2800   Median : -13.00   Median :17.00
##  Mean   : 64.41   Mean   :  0.3053   Mean   : -11.21   Mean   :11.31
##  3rd Qu.:123.00   3rd Qu.: 14.9000   3rd Qu.:  12.90   3rd Qu.:18.00
##  Max.   :162.00   Max.   : 60.3000   Max.   : 179.00   Max.   :29.00
##   gyros_belt_x        gyros_belt_y       gyros_belt_z
##  Min.   :-1.040000   Min.   :-0.64000   Min.   :-1.4600
##  1st Qu.:-0.030000   1st Qu.: 0.00000   1st Qu.:-0.2000
##  Median : 0.030000   Median : 0.02000   Median :-0.1000
##  Mean   :-0.005592   Mean   : 0.03959   Mean   :-0.1305
##  3rd Qu.: 0.110000   3rd Qu.: 0.11000   3rd Qu.:-0.0200
##  Max.   : 2.220000   Max.   : 0.64000   Max.   : 1.6200
##   accel_belt_x       accel_belt_y     accel_belt_z     magnet_belt_x
##  Min.   :-120.000   Min.   :-69.00   Min.   :-275.00   Min.   :-52.0
##  1st Qu.: -21.000   1st Qu.:  3.00   1st Qu.:-162.00   1st Qu.:  9.0
##  Median : -15.000   Median : 35.00   Median :-152.00   Median : 35.0
##  Mean   :  -5.595   Mean   : 30.15   Mean   : -72.59   Mean   : 55.6
##  3rd Qu.:  -5.000   3rd Qu.: 61.00   3rd Qu.:  27.00   3rd Qu.: 59.0
##  Max.   :  85.000   Max.   :164.00   Max.   : 105.00   Max.   :485.0
##  magnet_belt_y   magnet_belt_z       roll_arm         pitch_arm
##  Min.   :354.0   Min.   :-623.0   Min.   :-180.00   Min.   :-88.800
##  1st Qu.:581.0   1st Qu.:-375.0   1st Qu.: -31.77   1st Qu.:-25.900
##  Median :601.0   Median :-320.0   Median :   0.00   Median :  0.000
##  Mean   :593.7   Mean   :-345.5   Mean   :  17.83   Mean   : -4.612
##  3rd Qu.:610.0   3rd Qu.:-306.0   3rd Qu.:  77.30   3rd Qu.: 11.200
##  Max.   :673.0   Max.   : 293.0   Max.   : 180.00   Max.   : 88.500
```

```
##     yaw_arm          total_accel_arm  gyros_arm_x        gyros_arm_y
##  Min.   :-180.0000   Min.   : 1.00   Min.   :-6.37000   Min.   :-3.4400
##  1st Qu.: -43.1000   1st Qu.:17.00   1st Qu.:-1.33000   1st Qu.:-0.8000
##  Median :   0.0000   Median :27.00   Median : 0.08000   Median :-0.2400
##  Mean   :  -0.6188   Mean   :25.51   Mean   : 0.04277   Mean   :-0.2571
##  3rd Qu.:  45.8750   3rd Qu.:33.00   3rd Qu.: 1.57000   3rd Qu.: 0.1400
##  Max.   : 180.0000   Max.   :66.00   Max.   : 4.87000   Max.   : 2.8400
##   gyros_arm_z       accel_arm_x       accel_arm_y      accel_arm_z
##  Min.   :-2.3300   Min.   :-404.00   Min.   :-318.0   Min.   :-636.00
##  1st Qu.:-0.0700   1st Qu.:-242.00   1st Qu.: -54.0   1st Qu.:-143.00
##  Median : 0.2300   Median : -44.00   Median :  14.0   Median : -47.00
##  Mean   : 0.2695   Mean   : -60.24   Mean   :  32.6   Mean   : -71.25
##  3rd Qu.: 0.7200   3rd Qu.:  84.00   3rd Qu.: 139.0   3rd Qu.:  23.00
##  Max.   : 3.0200   Max.   : 437.00   Max.   : 308.0   Max.   : 292.00
##   magnet_arm_x     magnet_arm_y     magnet_arm_z    roll_dumbbell
##  Min.   :-584.0   Min.   :-392.0   Min.   :-597.0   Min.   :-153.71
##  1st Qu.:-300.0   1st Qu.:  -9.0   1st Qu.: 131.2   1st Qu.: -18.49
##  Median : 289.0   Median : 202.0   Median : 444.0   Median :  48.17
##  Mean   : 191.7   Mean   : 156.6   Mean   : 306.5   Mean   :  23.84
##  3rd Qu.: 637.0   3rd Qu.: 323.0   3rd Qu.: 545.0   3rd Qu.:  67.61
##  Max.   : 782.0   Max.   : 583.0   Max.   : 694.0   Max.   : 153.55
##  pitch_dumbbell     yaw_dumbbell      total_accel_dumbbell
##  Min.   :-149.59   Min.   :-150.871   Min.   : 0.00
##  1st Qu.: -40.89   1st Qu.: -77.644   1st Qu.: 4.00
##  Median : -20.96   Median :  -3.324   Median :10.00
##  Mean   : -10.78   Mean   :   1.674   Mean   :13.72
##  3rd Qu.:  17.50   3rd Qu.:  79.643   3rd Qu.:19.00
##  Max.   : 149.40   Max.   : 154.952   Max.   :58.00
##  gyros_dumbbell_x    gyros_dumbbell_y   gyros_dumbbell_z
##  Min.   :-204.0000   Min.   :-2.10000   Min.   : -2.380
##  1st Qu.:  -0.0300   1st Qu.:-0.14000   1st Qu.: -0.310
##  Median :   0.1300   Median : 0.03000   Median : -0.130
##  Mean   :   0.1611   Mean   : 0.04606   Mean   : -0.129
##  3rd Qu.:   0.3500   3rd Qu.: 0.21000   3rd Qu.:  0.030
##  Max.   :   2.2200   Max.   :52.00000   Max.   :317.000
##  accel_dumbbell_x  accel_dumbbell_y  accel_dumbbell_z  magnet_dumbbell_x
##  Min.   :-419.00   Min.   :-189.00   Min.   :-334.00   Min.   :-643.0
##  1st Qu.: -50.00   1st Qu.:  -8.00   1st Qu.:-142.00   1st Qu.:-535.0
##  Median :  -8.00   Median :  41.50   Median :  -1.00   Median :-479.0
##  Mean   : -28.62   Mean   :  52.63   Mean   : -38.32   Mean   :-328.5
##  3rd Qu.:  11.00   3rd Qu.: 111.00   3rd Qu.:  38.00   3rd Qu.:-304.0
##  Max.   : 235.00   Max.   : 315.00   Max.   : 318.00   Max.   : 592.0
##  magnet_dumbbell_y magnet_dumbbell_z  roll_forearm       pitch_forearm
##  Min.   :-3600     Min.   :-262.00   Min.   :-180.0000   Min.   :-72.50
##  1st Qu.:  231     1st Qu.: -45.00   1st Qu.:  -0.7375   1st Qu.:  0.00
##  Median :  311     Median :  13.00   Median :  21.7000   Median :  9.24
##  Mean   :  221     Mean   :  46.05   Mean   :  33.8265   Mean   : 10.71
##  3rd Qu.:  390     3rd Qu.:  95.00   3rd Qu.: 140.0000   3rd Qu.: 28.40
##  Max.   :  633     Max.   : 452.00   Max.   : 180.0000   Max.   : 89.80
##   yaw_forearm      total_accel_forearm gyros_forearm_x
##  Min.   :-180.00   Min.   :  0.00      Min.   :-22.000
##  1st Qu.: -68.60   1st Qu.: 29.00      1st Qu.: -0.220
##  Median :   0.00   Median : 36.00      Median :  0.050
##  Mean   :  19.21   Mean   : 34.72      Mean   :  0.158
```

```
##  3rd Qu.: 110.00   3rd Qu.: 41.00      3rd Qu.:  0.560
##  Max.   : 180.00   Max.   :108.00      Max.   :  3.970
##  gyros_forearm_y     gyros_forearm_z    accel_forearm_x   accel_forearm_y
##  Min.   : -7.02000   Min.   : -8.0900   Min.   :-498.00   Min.   :-632.0
##  1st Qu.: -1.46000   1st Qu.: -0.1800   1st Qu.:-178.00   1st Qu.:  57.0
##  Median :  0.03000   Median :  0.0800   Median : -57.00   Median : 201.0
##  Mean   :  0.07517   Mean   :  0.1512   Mean   : -61.65   Mean   : 163.7
##  3rd Qu.:  1.62000   3rd Qu.:  0.4900   3rd Qu.:  76.00   3rd Qu.: 312.0
##  Max.   :311.00000   Max.   :231.0000   Max.   : 477.00   Max.   : 923.0
##  accel_forearm_z   magnet_forearm_x  magnet_forearm_y magnet_forearm_z
##  Min.   :-446.00   Min.   :-1280.0   Min.   :-896.0   Min.   :-973.0
##  1st Qu.:-182.00   1st Qu.: -616.0   1st Qu.:   2.0   1st Qu.: 191.0
##  Median : -39.00   Median : -378.0   Median : 591.0   Median : 511.0
##  Mean   : -55.29   Mean   : -312.6   Mean   : 380.1   Mean   : 393.6
##  3rd Qu.:  26.00   3rd Qu.:  -73.0   3rd Qu.: 737.0   3rd Qu.: 653.0
##  Max.   : 291.00   Max.   :  672.0   Max.   :1480.0   Max.   :1090.0
##  classe
##  A:5580
##  B:3797
##  C:3422
##  D:3216
##  E:3607
##
```