Excercise Quality Prediction Model

K Chandrasekaran(Raj)

January 22, 2016

Introduction In this report, we will examine data set obtained from https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv. This data set have classe response variable that indicate how well excercise is performed. This is Multilevel classification to be predicted with predictors. We will explore the data set, prepare the data set to drop predictors with no value or variances.Next we will examine remaining predictors to see which are the interested ones. This feature selection may need pre-processing. We will select models, tune paramters & compare the models to conclude.

Exploring Data Set After loading data set, we can see there are 160 variables & 19622 rows. We can see first few are user names & time stamps that may not be needed for prediction of classe response variable. Removing any predictor with NA, we reduced number of predictors. Using nearZeroVar from caret package we can remove those predictors which may not be needed. This reduces the predictors to 53 variable including the response variable.Below set of R code does this clearing.

```r
suppressWarnings(library(caret))
```

```
## Loading required package: lattice

## Loading required package: ggplot2

## Note: the specification for S3 class "family" in package 'MatrixModels' seems equivalent to one from
```

```r
df <- read.csv("F:/Raj/DataScienceTools/Assignment/MachineLearn/pml-training.csv")
# Remove user name & time predictors
df <- df[,-(1:7)]
navar <- sapply(df,anyNA)
navar <- as.data.frame(navar)
navar <- subset.data.frame(navar, navar == "FALSE")
navar <- row.names(navar)
df <- subset(df, select = navar)
# Check for any near Zero variances
nz <- nearZeroVar(df, saveMetrics = TRUE)
nz <- subset.data.frame(nz, nz$zeroVar == "FALSE" & nz$nzv == "FALSE")
nz <- row.names(nz)
df <- subset(df, select = nz)
# Create train and test data
set.seed(1976)
trainIndex <- createDataPartition(df$classe, p=0.7,list = FALSE)
traindata <- df[trainIndex,]
testdata <- df[-trainIndex,]
```

Feature Selection 52 predictors are still huge to build model. We will need to reduce this. Let us check out which of these variables have high correlation. Those highly correlated variable is what we may need to build our models on.

```r
# Determine Highly Correlated predictors.
hc <- cor(traindata[,-53])
hc <- findCorrelation(hc, cutoff = 0.85, names = TRUE)
hc <- c(hc,"classe")
```

High Correlation variable that have atleast 85% correlation seems to be 10 excluding response variable classe. That is significant dimension reduction.

Response variables are distinguished as below. exactly according to the specification (Class A), throwing the elbows to the front (Class B), lifting the dumbbell only halfway (Class C), lowering the dumbbell only halfway (Class D) and throwing the hips to the front (Class E).

Let us see whether 10 is enough to predict with predict tree. compare it with predicting with all 52 variables.

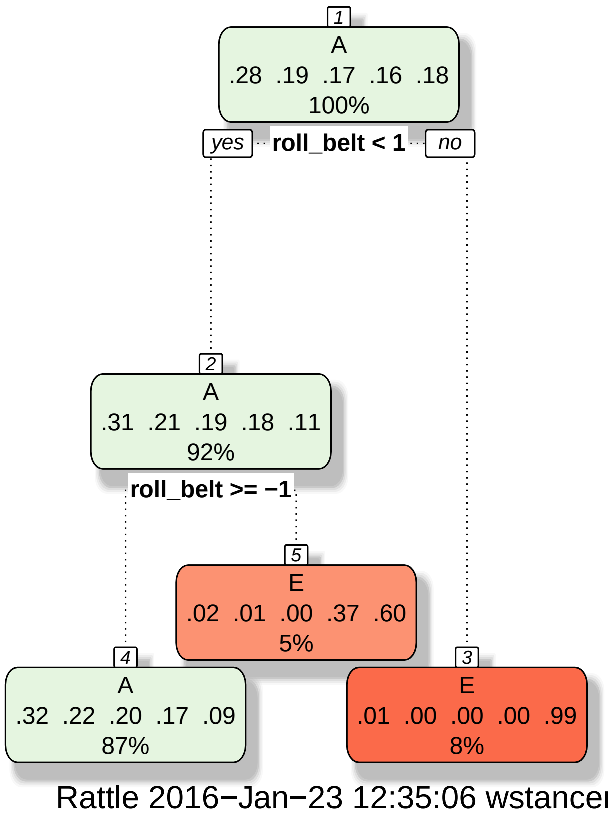
```r
suppressWarnings(library(rattle))
```

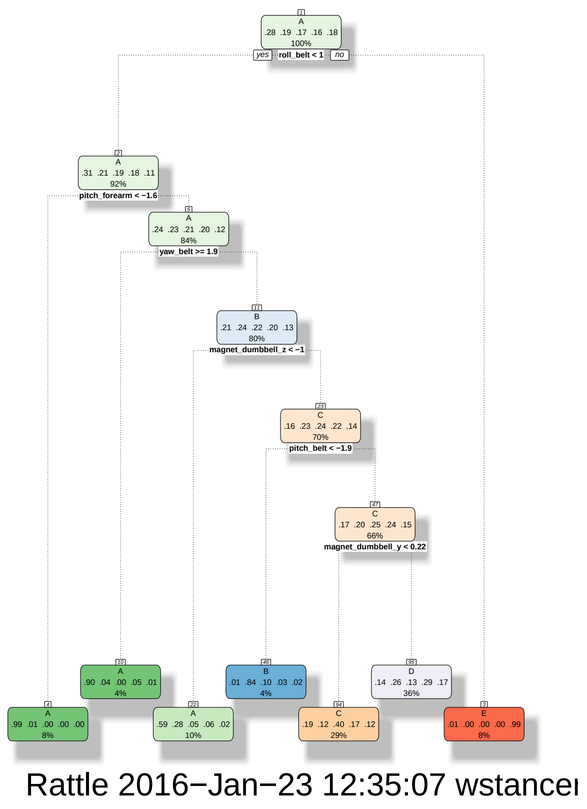
```
## Rattle: A free graphical interface for data mining with R.
## Version 4.0.5 Copyright (c) 2006-2015 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```r
# Select 85% Correlated variables from training data set/
tdata <- subset(traindata, select = hc)
#Build rpart model using only 85% correlated predictors & all predictors
modelrpart <- train(classe ~., data=tdata, method="rpart",preProcess=c("center", "scale"))
```

```
## Loading required package: rpart
```

```r
modelrpartall <- train(classe ~., data=traindata, method="rpart", preProcess=c("center", "scale"))
# Plot the tree using the model
par(mfrow=c(1,2))
fancyRpartPlot(modelrpart$finalModel)
fancyRpartPlot(modelrpartall$finalModel)
```





Above tree clearly says 10 variables are not enough. Even when using all variables, we cannot see predicting all the classes with this model. We can see the logic to this as why would highly correlated variable have some sort of connection with response variable. rpart model shows important variables are just 14 as given below

```
par(mfrow=c(1,1))
# Important predictors using rpart model.
varImp(modelrpartall)
```

```
## rpart variable importance
##
##   only 20 most important variables shown (out of 52)
##
##                   Overall
## roll_belt          100.00
## magnet_dumbbell_y   75.88
## pitch_forearm       70.81
## roll_dumbbell       51.46
## roll_forearm        37.33
## accel_belt_z        31.64
## magnet_belt_y       29.68
## pitch_belt          26.12
## total_accel_belt    25.66
## magnet_arm_x        20.08
## accel_arm_x         19.01
## yaw_belt            16.27
## magnet_dumbbell_z   15.52
## accel_forearm_x     12.35
## magnet_dumbbell_x   11.86
## accel_dumbbell_y    11.49
## accel_belt_x         0.00
## accel_arm_y          0.00
## accel_arm_z          0.00
## magnet_belt_z        0.00
```

Model Building Using one of the boosting method that uses trees like gbm constructed model. modelgbm <- train(classe ~., data=traindata, method="gbm")

```
#Build model using gbm method
modelgbm <- train(classe ~., data=traindata, method="gbm", verbose=FALSE)
```

```
## Loading required package: gbm
```

```
## Warning: package 'gbm' was built under R version 3.2.3
```

```
## Loading required package: survival
```

```
##
## Attaching package: 'survival'
```

```
## The following object is masked from 'package:caret':
##
##     cluster
```

```
## Loading required package: splines
```

```
## Loading required package: parallel
```

```
## Loaded gbm 2.1.1
```

```
## Loading required package: plyr
```

The final values used for the model were n.trees = 150, interaction.depth = 3, shrinkage = 0.1 and n.minobsinnode = 10.

```
#print important variables
varImp(modelgbm)
```

```
## gbm variable importance
##
##   only 20 most important variables shown (out of 52)
##
##                   Overall
## roll_belt         100.000
## pitch_forearm      48.002
## yaw_belt           34.778
## magnet_dumbbell_z  31.769
## magnet_dumbbell_y  26.178
## magnet_belt_z      24.513
## roll_forearm       23.736
## accel_forearm_x    13.919
## roll_dumbbell      12.923
## gyros_belt_z       12.242
## pitch_belt         10.907
## gyros_dumbbell_y   10.313
## accel_dumbbell_y    7.979
## accel_forearm_z     7.722
## magnet_forearm_x    6.938
## accel_dumbbell_x    6.934
## yaw_arm             6.451
## magnet_forearm_z    6.406
## magnet_arm_z        5.891
## magnet_belt_y       5.137
```

Evaluating Model We have constructed 2 models using rpart & gbm methods. Let us compare the accuracy of these models on test data.

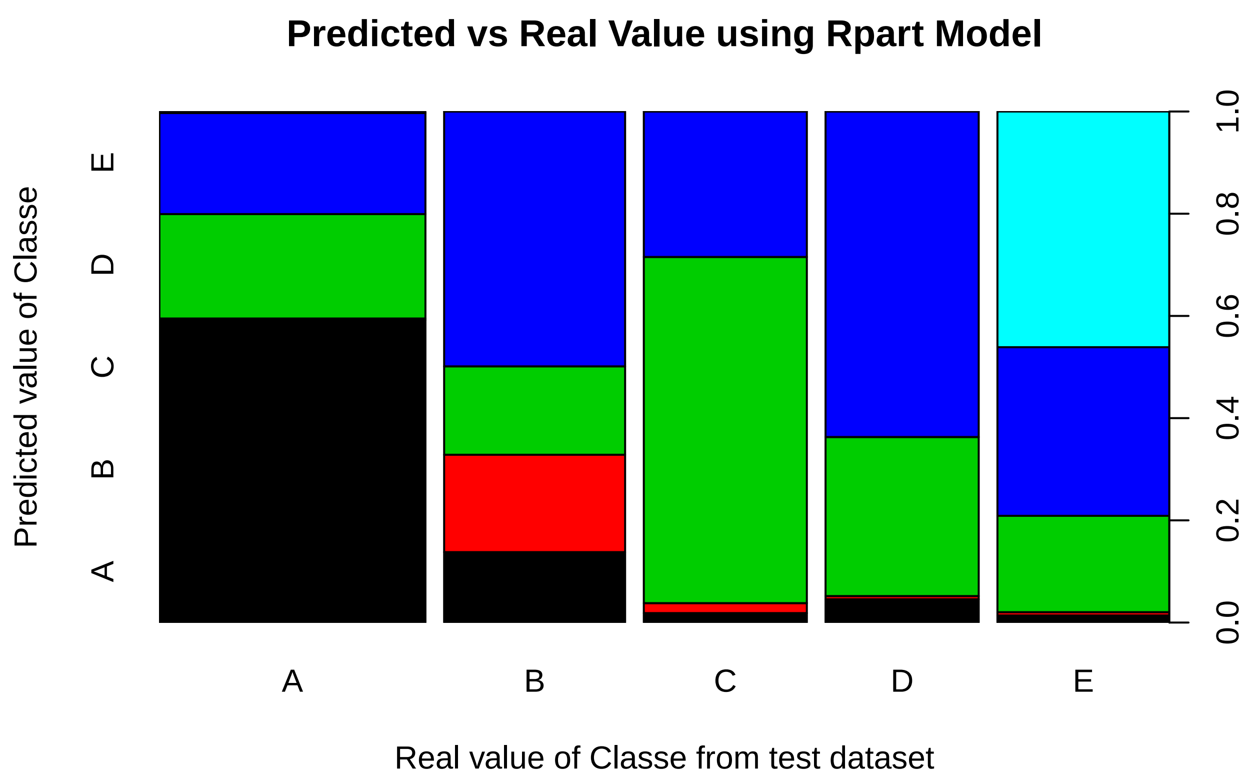
```
#Predict Classe for test dataset using the model objects
predictgbmt <- predict(modelgbm, testdata[,-53])
predictrpart <- predict(modelrpartall, testdata[,-53])
# Print out accuracy & other model measurements
confusionMatrix(testdata[,53], predictgbmt)
```

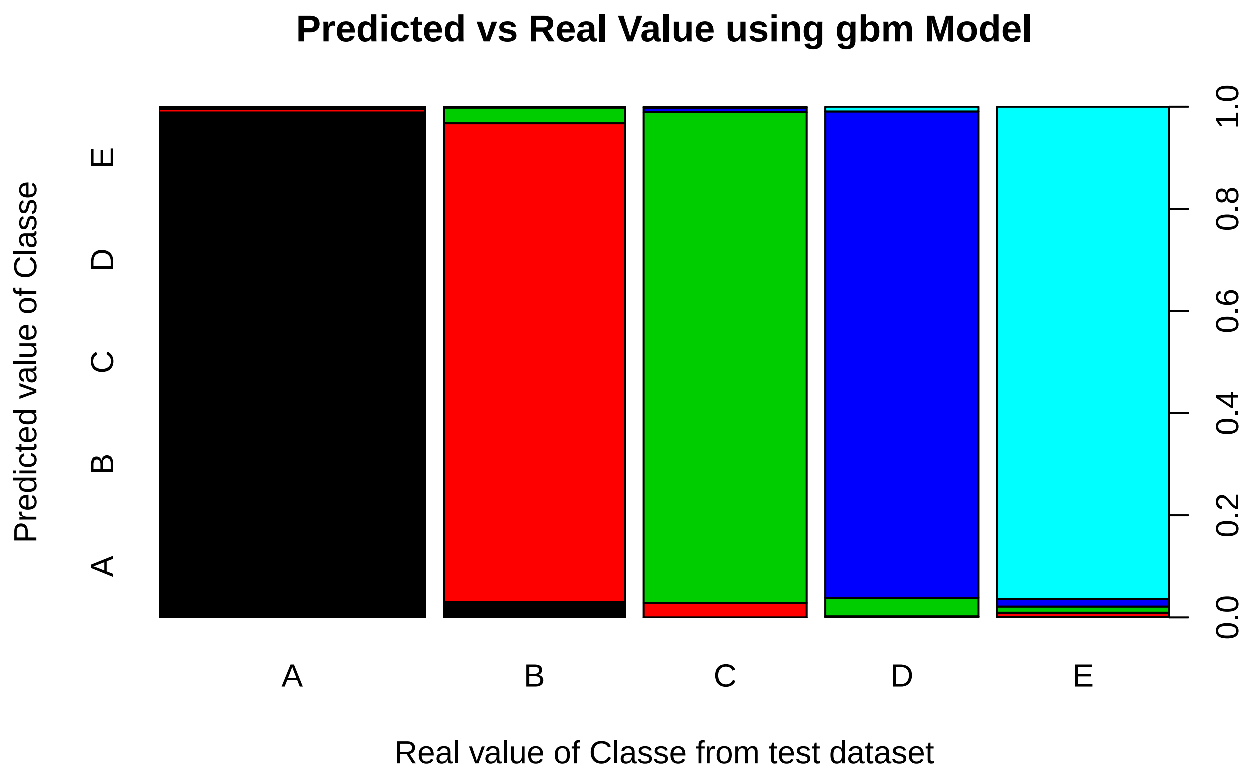
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1655   11    2    5    1
##          B   34 1068   35    1    1
##          C    0   29  986    9    2
##          D    1    1   35  918    9
##          E    1    9   13   16 1043
##
## Overall Statistics
##
##                Accuracy : 0.9635
##                  95% CI : (0.9584, 0.9681)
##     No Information Rate : 0.2873
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9538
##  Mcnemar's Test P-Value : 4.479e-07
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9787   0.9553   0.9206   0.9673   0.9877
## Specificity            0.9955   0.9851   0.9917   0.9907   0.9919
## Pos Pred Value         0.9886   0.9377   0.9610   0.9523   0.9640
## Neg Pred Value         0.9915   0.9895   0.9825   0.9937   0.9973
## Prevalence             0.2873   0.1900   0.1820   0.1613   0.1794
## Detection Rate         0.2812   0.1815   0.1675   0.1560   0.1772
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9871   0.9702   0.9562   0.9790   0.9898
```

```
confusionMatrix(testdata[,53], predictrpart)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 991   5 342 331   5
##          B 157 217 197 568   0
##          C  19  20 695 292   0
##          D  44   6 300 614   0
##          E  15   7 204 357 499
##
## Overall Statistics
##
##                Accuracy : 0.5125
##                  95% CI : (0.4996, 0.5253)
##     No Information Rate : 0.3674
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3944
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8083  0.85098   0.3999   0.2840  0.99008
## Specificity            0.8534  0.83623   0.9202   0.9060  0.89166
## Pos Pred Value         0.5920  0.19052   0.6774   0.6369  0.46118
## Neg Pred Value         0.9442  0.99199   0.7853   0.6854  0.99896
## Prevalence             0.2083  0.04333   0.2953   0.3674  0.08564
## Detection Rate         0.1684  0.03687   0.1181   0.1043  0.08479
## Detection Prevalence   0.2845  0.19354   0.1743   0.1638  0.18386
## Balanced Accuracy      0.8309  0.84361   0.6600   0.5950  0.94087
```

Selecting Model Clearly rpart model accuracy results are pathetic. Good thing we did not build gbm method model using important variable derived from rpart method. rpart model miserably fails to detect classD. gbm method model has done well here with much higher accuracy rate. Below plot compares the predicted classe from rpart & gbm model versus real data in test data set.


Predicted vs Real Value using Rpart Model
Predicted value of Classe
A
B
C
D
E
Real value of Classe from test dataset
A
B
C
D
E
0.0
0.2
0.4
0.6
0.8
1.0



Conclusion Let us apply the gbm model to provided new test data that has 20 rows & predict which classe they belong to.

```
#Read the new data set
tdf <- read.csv("F:/Raj/DataScienceTools/Assignment/MachineLearn/pml-testing.csv")
# Predict the classe variable for new data set using gbm method object
predictgbmt2 <- predict(modelgbm, tdf)
#print out the predicted results.
table(predictgbmt2)
```

```
## predictgbmt2
## A B C D E
## 7 8 1 1 3
```

```
predictgbmt2
```

```
##  [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
```