PROBLEM STATEMENT

"Using devices such as Jawbone Up, Nike FuelBand, and Fitbit it is now possible to collect a large amount of data about personal activity relatively inexpensively. These type of devices are part of the quantified self movement ??? a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. One thing that people regularly do is quantify how much of a particular activity they do, but they rarely quantify how well they do it. In this project, your goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways."

What should be submitted.

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.

Data

The training data for this project are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

The test data are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

SOLUTION

Open the R tool and install the following packages by typing

Install.packages("caret")

Install.packages("randomForest")

Install.packages("e1071")

Load the Give Libraries using the following command.

Library(caret)

library(randomForest)

Library(e1071)

The training and Testing data set is available as links online.You can store the CSV Files as per the following command.

Urltrain = "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"

training = read.csv(url(Urltrain), na.strings=c("NA","#DIV/0!",""))

Urltest = "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

testing11 <- read.csv(url(Urltest), na.strings=c("NA","#DIV/0!",""))

Addtionally,you can manually download the CSV Files from the given link by using the:-

Getwd()

And setting to the Location where you have stored the .CSV Files.

There are many Columns which show high Variance, We get rid of them first by using the command:-

training.first <- training[, colSums(is.na(training)) == 0]

remove = c('X', 'user_name', 'raw_timestamp_part_1', 'raw_timestamp_part_2', 'cvtd_timestamp', 'new_window', 'num_window')

training.second <- training.first[, -which(names(training.first) %in% remove)]

We check the dimensions of the first and second row:-

> nrow(training.first)

[1] 19622

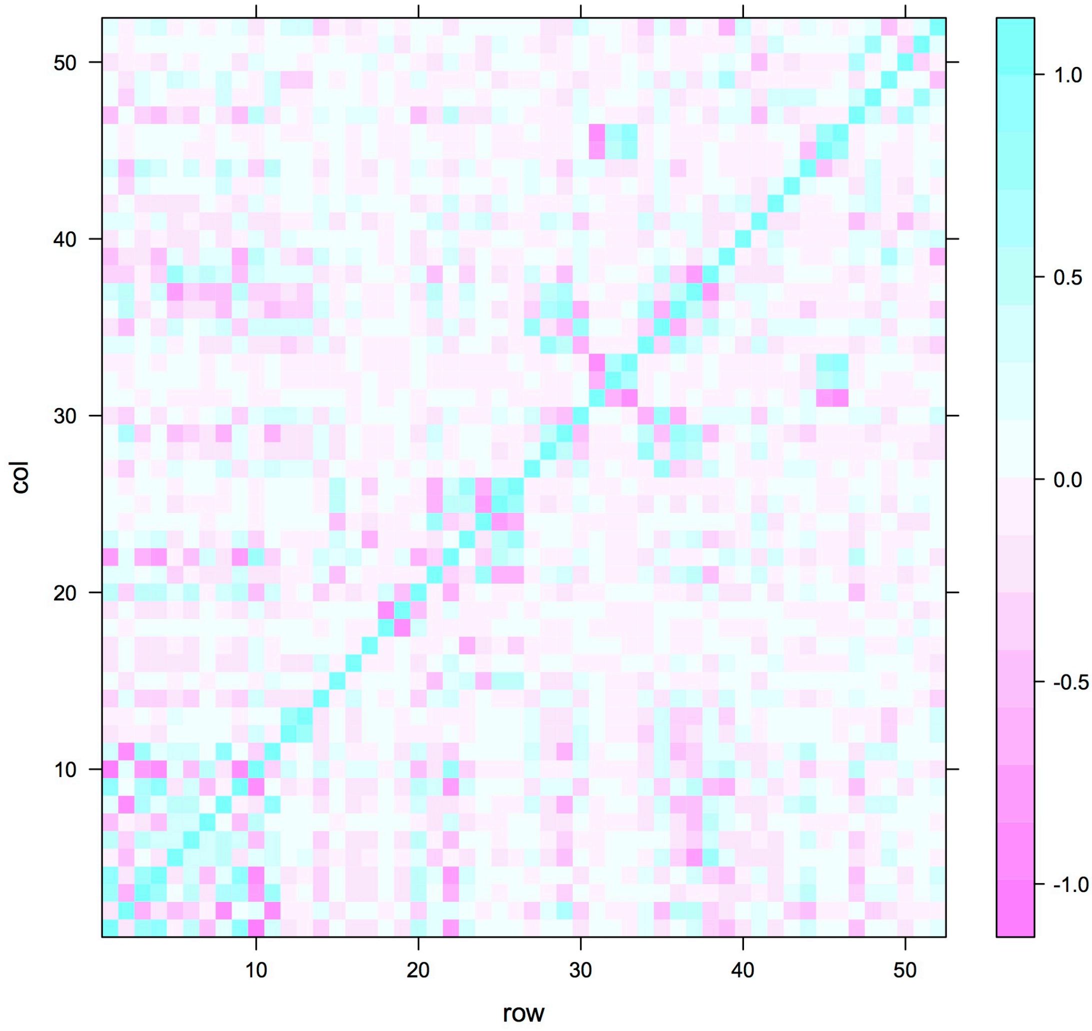
> nrow(training.second)

[1] 19622

There are many Columns which show high Variance,We get rid of them first by using the command:-

```
> zV= nearZeroVar(training.dere[sapply(training.second, is.numeric)], saveMetrics =
TRUE)

> training.nonzerovar = training.second[,zeroVar[, 'nzv']==0]

> dim(training.nonzerovar)

[1] 19622    53

> cM <- cor(na.omit(training.nonzerovar[sapply(training.nonzerovar, is.numeric)]))

> dim(cM)

[1] 52 52

cDF <- expand.grid(row = 1:52, col = 1:52)

cDF$correlation <- as.vector(cM)

> cDF <- expand.grid(row = 1:52, col = 1:52)

> cDF$correlation <- as.vector(cM)
```

- levelplot(correlation ~ row+ col, cDF)




```
> rcor = findCorrelation(corrMatrix, cutoff = .87, verbose = TRUE)
Compare row 10  and column  1 with corr  0.992
  Means:  0.27 vs 0.168 so flagging column 10
Compare row 1  and column  9 with corr  0.925
  Means:  0.25 vs 0.164 so flagging column 1
Compare row 9  and column  4 with corr  0.928
  Means:  0.233 vs 0.161 so flagging column 9
Compare row 8  and column  2 with corr  0.966
  Means:  0.245 vs 0.157 so flagging column 8
Compare row 2  and column  11 with corr  0.884
```

Means: 0.228 vs 0.154 so flagging column 2

Compare row 19 and column 18 with corr 0.918

Means: 0.09 vs 0.154 so flagging column 18

Compare row 46 and column 31 with corr 0.914

Means: 0.101 vs 0.158 so flagging column 31

Compare row 46 and column 33 with corr 0.933

Means: 0.082 vs 0.161 so flagging column 33

All correlations <= 0.87

```
> training.decor = training.nonzerovar[,-rcor]
> dim(training.decor)
[1] 19622    45
```

We now split our Training Data into 2 Sets with a 65% split to our modified training set and the remaining 35% which stays in the testing set.

```
> inTrain <- createDataPartition(y=training.decor$classe, p=0.65, list=FALSE)
> train <- training.decor[inTrain,];
>  test <- training.decor[-inTrain,]
> dim(training);
[1] 13737    46
> dim(testing)
[1] 5885   46
```

```
set.seed(999)
rf.training=randomForest(classe~.,data=train,ntree=100, importance=TRUE)
rf.training
y=varImpPlot(rf.training,)
> y=varImpPlot(rf.training,)
> y
                    MeanDecreaseAccuracy MeanDecreaseGini
yaw_belt                       32.029426        724.83447
total_accel_belt               12.604981        255.56995
```

gyros_belt_x	15.593406	86.91657
gyros_belt_y	8.756169	101.48180
gyros_belt_z	19.055566	340.55547
magnet_belt_x	17.916008	215.04822
magnet_belt_y	16.679845	408.57591
magnet_belt_z	15.244762	328.98098
roll_arm	20.024840	236.10620
pitch_arm	10.497100	144.78818
yaw_arm	15.096575	204.52550
total_accel_arm	11.077107	80.05923
gyros_arm_y	20.526429	123.59521
gyros_arm_z	15.786235	61.49579
accel_arm_x	10.504261	184.80547
accel_arm_y	13.453446	135.70542
accel_arm_z	13.766164	109.37582
magnet_arm_x	8.604656	188.95669
magnet_arm_y	9.299032	168.88421
magnet_arm_z	15.245923	147.37576
roll_dumbbell	15.598563	326.60878
pitch_dumbbell	8.315829	135.90273
yaw_dumbbell	13.627680	194.76769
total_accel_dumbbell	12.948539	205.95248
gyros_dumbbell_y	13.828084	223.95488
accel_dumbbell_x	13.039806	189.44484
accel_dumbbell_y	17.225338	286.71295
accel_dumbbell_z	16.209837	244.10784
magnet_dumbbell_x	14.924324	337.38854
magnet_dumbbell_y	20.257029	493.11831
magnet_dumbbell_z	28.686235	572.07015
roll_forearm	14.688614	425.69150
pitch_forearm	19.962761	566.22108

```
yaw_forearm             14.481442   142.72084
total_accel_forearm     14.186970    88.99483
gyros_forearm_x         15.437910    70.97954
gyros_forearm_y         20.587655   115.92990
gyros_forearm_z         17.605637    77.98011
accel_forearm_x         15.020910   226.22867
accel_forearm_y         12.898990   119.38238
accel_forearm_z         17.083710   211.92138
magnet_forearm_x        10.898452   164.74667
magnet_forearm_y        13.269938   189.39069
magnet_forearm_z        23.045708   227.39173

tree.pred=predict(rf.training,test,type="class")
predMatrix = with(testing,table(tree.pred,classe))
> confusionMatrix(tree.pred,test$classe)
Confusion Matrix and Statistics

          Reference
Prediction    A    B    C    D    E
         A 1949    5    0    0    0
         B    4 1317   10    0    0
         C    0    6 1186   18    0
         D    0    0    1 1104    2
         E    0    0    0    3 1260

Overall Statistics

               Accuracy : 0.9929
                 95% CI : (0.9906, 0.9947)
    No Information Rate : 0.2845
    P-Value [Acc > NIR] : < 2.2e-16
```

Kappa : 0.991

Mcnemar's Test P-Value : NA

Statistics by Class:

	Class: A	Class: B	Class: C	Class: D	Class: E
Sensitivity	0.9980	0.9917	0.9908	0.9813	0.9984
Specificity	0.9990	0.9975	0.9958	0.9995	0.9995
Pos Pred Value	0.9974	0.9895	0.9802	0.9973	0.9976
Neg Pred Value	0.9992	0.9980	0.9981	0.9964	0.9996
Prevalence	0.2845	0.1934	0.1744	0.1639	0.1838
Detection Rate	0.2839	0.1918	0.1728	0.1608	0.1835
Detection Prevalence	0.2846	0.1939	0.1763	0.1613	0.1840
Balanced Accuracy	0.9985	0.9946	0.9933	0.9904	0.9989

Random Forests give us Highly Accurate results.Hence,we go in for this for testing our Test Data.

Now,we use our Random Forest Model to test our Test set predictors.

TESTING DATA PREDICTIONS

```
> Testsetpredictors <- predict(rf.training, testing11)
> Testsetpredictors
 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
 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
```