Practical Machine Learning

Snehadrita Das

2022-08-21

Executive Summary

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. 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, our goal was to use the data rom accelerometers on the belt, forearm, arm, and dumbell of 6 participants and predict the manner in which they did the exercise. We have considered *nested models* and let them compete for greater accuracy and lesser out of sample error. We have considered machine learning algorithms such as *Decision Trees, Random Forest and Gradient Boosted Trees*. Among the models, the model with Random Forest as its algorithm predicts with the highest accuracy of 97.4% and 2.6% of out of sample error.

Background and Data

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. 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. More information is available from the website here (http://groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset).

- Train Data (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv)
- Test Data (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv)

Getting Data

```
url1<-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
url2<-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

destfile1<-"C:/Users/Hp/Desktop/train.csv"
destfile2<-"C:/Users/Hp/Desktop/test.csv"

download.file(url1,destfile1)
download.file(url2,destfile2)

traindata<-read.csv("train.csv")  ## training data
testdata<-read.csv("test.csv")    ## test data

dim(traindata);dim(testdata)
```

```
[1] 19622   160
```

```
[1]  20 160
```

Library Dependency

```
library(corrplot)
library(caret)
library(rpart.plot)
library(randomForest)
library(gbm)
```

Data Processing

- We shall explore the training dataset for missing values and omit them. We shall omit the variables with loads of mussing values. To be noted that all the processing and working do be done in the training data set. The test data is only for prediction.
- We shall also omit the variables with very smaller (near zero) variance as they contribute little to the model and prediction procedure.

```
na<-rep(0,160)
for(i in 1:160){
     na[i]=sum(is.na(traindata[,i]))  ## looking for NA values and storing them in the vector
na
}
idx<-which(na==19216)                ## indices for the variables with mostly NA values
traindata<- traindata[,-c(1:7,idx)]  ## processing
near.zero<-nearZeroVar(traindata)     ## indices for the variables with mostly lesser varianc
es
final.traindata<-traindata[,-near.zero]  ## Processed training set

dim(final.traindata)
```

```
[1] 19622    53
```

Train set and Validation set

```
set.seed(17)
intrain<-createDataPartition(final.traindata$classe,p=0.7,list=FALSE)
train.set<-final.traindata[intrain,]   ## train set
validation<-final.traindata[-intrain,]  ## validation set

dim(train.set);dim(validation)
```

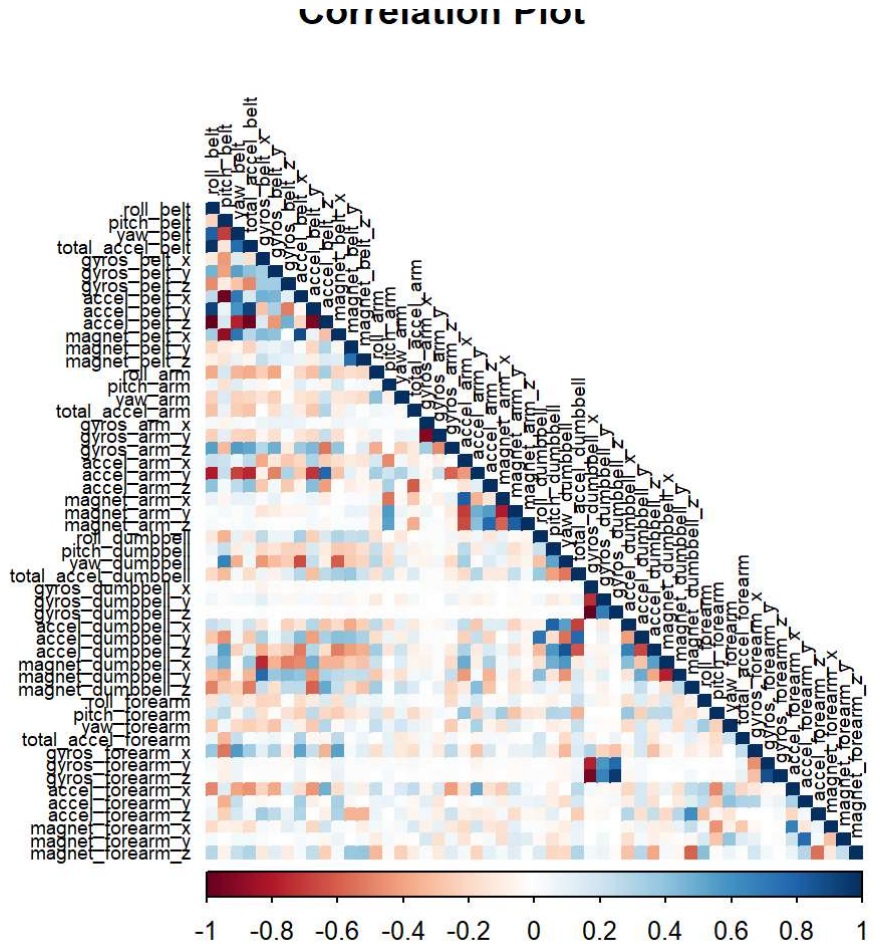
```
[1] 13737    53
```

```
[1] 5885   53
```

Exploratory Data Analysis

- We shall see the association between the variables in the train set

```
corrplot(cor(train.set[,-53]),method = "color",type = "lower",
                  tl.cex = 0.6, tl.col = rgb(0, 0, 0),main="Correlation Plot")
```



Cross Validation

Since Cross Validation is a very powerful tool and a statistical method used to estimate the performance (or accuracy) of machine learning models and is used to protect against overfitting in a predictive model, particularly in a case where the amount of data may be limited, we shall be considering cross valiadtion as our train control.

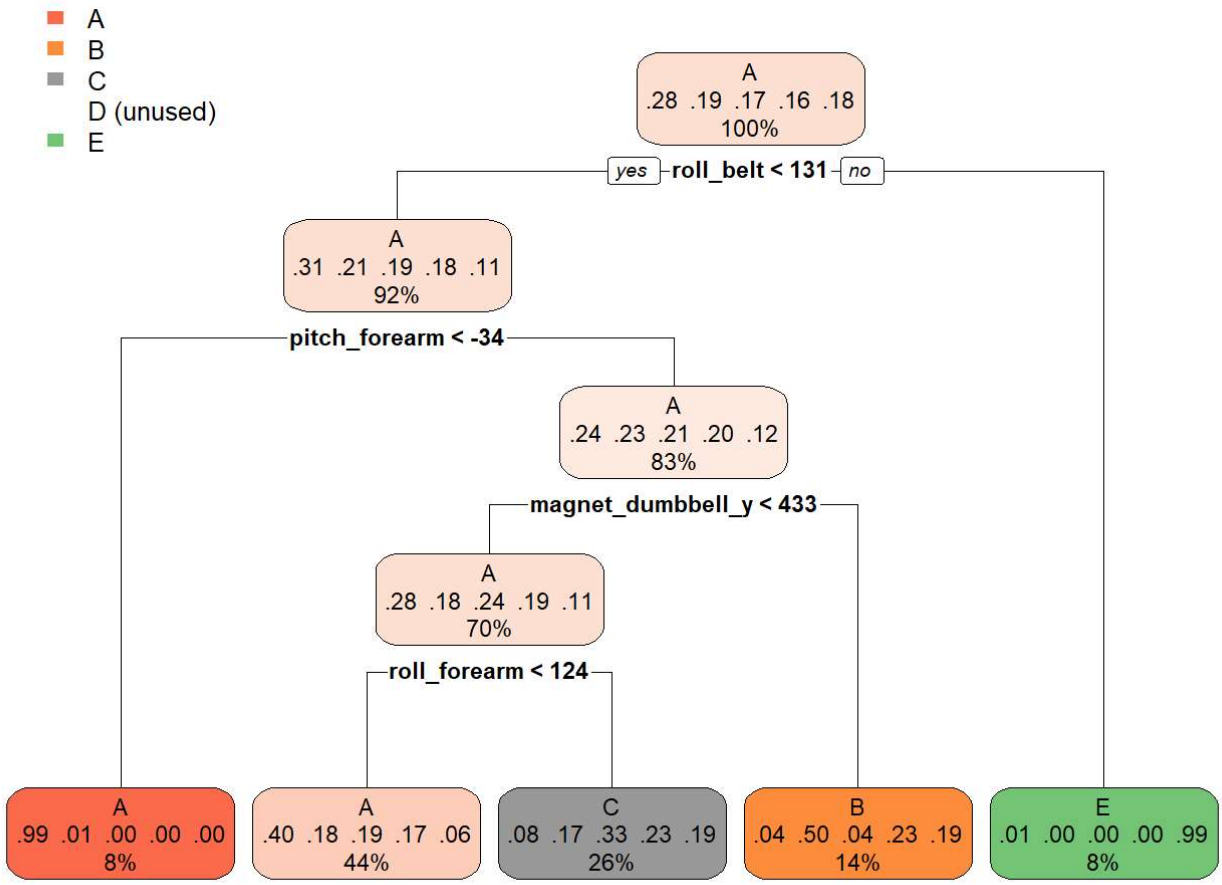
```
## cross validation
control<-trainControl(method = "cv",number = 3)
```

Algorithms and Models

Decision Tree

Model

```
model.dt<-train(classe~.,data=train.set,method="rpart",trControl=control)
rpart.plot(model.dt$finalModel)
```



Prediction and Accuracy

```
dt.pred<-predict(model.dt,newdata = validation)  ## prediction
dt.mat<-confusionMatrix(dt.pred,factor(validation$classe))  ## accuracy
dt.mat
```

```
Confusion Matrix and Statistics

          Reference
Prediction    A    B    C    D    E
         A 1512  443  453  428  164
         B   28  398   36  169  153
         C  128  298  537  367  279
         D    0    0    0    0    0
         E    6    0    0    0  486

Overall Statistics

               Accuracy : 0.4984
                 95% CI : (0.4855, 0.5112)
    No Information Rate : 0.2845
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.3453

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: A Class: B Class: C Class: D Class: E
Sensitivity            0.9032  0.34943  0.52339   0.0000  0.44917
Specificity            0.6466  0.91867  0.77938   1.0000  0.99875
Pos Pred Value         0.5040  0.50765  0.33375      NaN  0.98780
Neg Pred Value         0.9438  0.85473  0.88564   0.8362  0.88949
Prevalence             0.2845  0.19354  0.17434   0.1638  0.18386
Detection Rate         0.2569  0.06763  0.09125   0.0000  0.08258
Detection Prevalence   0.5098  0.13322  0.27341   0.0000  0.08360
Balanced Accuracy      0.7749  0.63405  0.65139   0.5000  0.72396
```

```
dt.mat$overall
```

```
      Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
  4.983857e-01   3.452997e-01   4.855298e-01   5.112433e-01   2.844520e-01
AccuracyPValue  McnemarPValue
 1.945222e-261            NaN
```

Random Forest

Model

```
## random forest
model.rf=train(classe~.,data=train.set,method="rf",trControl=control,prox=TRUE,ntree=4)
model.rf$finalModel
```

```
Call:
 randomForest(x = x, y = y, ntree = 4, mtry = min(param$mtry,      ncol(x)), proximity = TRU
E)
               Type of random forest: classification
                     Number of trees: 4
No. of variables tried at each split: 27

        OOB estimate of  error rate: 5.87%
Confusion matrix:
     A    B    C    D    E class.error
A 3193   38   12   10    9  0.02115267
B   71 2048   38   35   31  0.07872245
C   17   66 1878   53   18  0.07578740
D   26   36   62 1760   23  0.07708443
E   18   50   30   35 2002  0.06229508
```

Prediction and Accuracy

```
rf.pred<-predict(model.rf,newdata = validation)
rf.mat<-confusionMatrix(rf.pred,factor(validation$classe))
rf.mat
```

```
Confusion Matrix and Statistics

          Reference
Prediction    A    B    C    D    E
         A 1666   39    1    0    0
         B    5 1080   27    3    8
         C    2   10  984   12    3
         D    1    5   13  941    6
         E    0    5    1    8 1065

Overall Statistics

               Accuracy : 0.9747
                 95% CI : (0.9703, 0.9785)
    No Information Rate : 0.2845
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.968

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: A Class: B Class: C Class: D Class: E
Sensitivity            0.9952   0.9482   0.9591   0.9761   0.9843
Specificity            0.9905   0.9909   0.9944   0.9949   0.9971
Pos Pred Value         0.9766   0.9617   0.9733   0.9741   0.9870
Neg Pred Value         0.9981   0.9876   0.9914   0.9953   0.9965
Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
Detection Rate         0.2831   0.1835   0.1672   0.1599   0.1810
Detection Prevalence   0.2899   0.1908   0.1718   0.1641   0.1833
Balanced Accuracy      0.9929   0.9696   0.9768   0.9855   0.9907
```

```
rf.mat$overall
```

```
      Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
     0.9746814      0.9679535      0.9703399      0.9785432      0.2844520
AccuracyPValue  McnemarPValue
     0.0000000            NaN
```

Gradient Boosted Trees

Model

```
model.gbm<-train(classe~.,data=train.set,method="gbm",trControl=control,verbose=FALSE)
print(model.gbm$finalModel)
```

```
A gradient boosted model with multinomial loss function.
150 iterations were performed.
There were 52 predictors of which 51 had non-zero influence.
```

Prediction and Accuracy

```
gbm.pred<-predict(model.gbm,newdata = validation)
gbm.mat<-confusionMatrix(gbm.pred,factor(validation$classe))
gbm.mat
```

```
Confusion Matrix and Statistics

          Reference
Prediction    A    B    C    D    E
         A 1648   49    0    0    1
         B   13 1054   23    9   14
         C    9   34  988   31   11
         D    1    1   11  912   12
         E    3    1    4   12 1044

Overall Statistics

               Accuracy : 0.9594
                 95% CI : (0.954, 0.9643)
    No Information Rate : 0.2845
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9486

 Mcnemar's Test P-Value : 5.089e-10

Statistics by Class:

                     Class: A Class: B Class: C Class: D Class: E
Sensitivity            0.9845   0.9254   0.9630   0.9461   0.9649
Specificity            0.9881   0.9876   0.9825   0.9949   0.9958
Pos Pred Value         0.9706   0.9470   0.9208   0.9733   0.9812
Neg Pred Value         0.9938   0.9822   0.9921   0.9895   0.9921
Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
Detection Rate         0.2800   0.1791   0.1679   0.1550   0.1774
Detection Prevalence   0.2885   0.1891   0.1823   0.1592   0.1808
Balanced Accuracy      0.9863   0.9565   0.9727   0.9705   0.9804
```

```
gbm.mat$overall
```

```
      Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
  9.593883e-01   9.486067e-01   9.540274e-01   9.642874e-01   2.844520e-01
AccuracyPValue  McnemarPValue
  0.000000e+00   5.088867e-10
```

Comparisons

```
      Algorithm  Accuracy Out.of.sample.Error
1 Decision Tree 0.4983857          0.50161427
2 Random Forest 0.9746814          0.02531861
3           GBM 0.9593883          0.04061172
```

- ***We see that the model with Random Forest algorithm predicts with 97.4% accuracy and hence we shall use this model to perform prediction o our test data***

Prediction on Test data

```
predict(model.rf,testdata)
```

```
 [1] B A B A A E D B A A B C B A E E A B A B
Levels: A B C D E
```

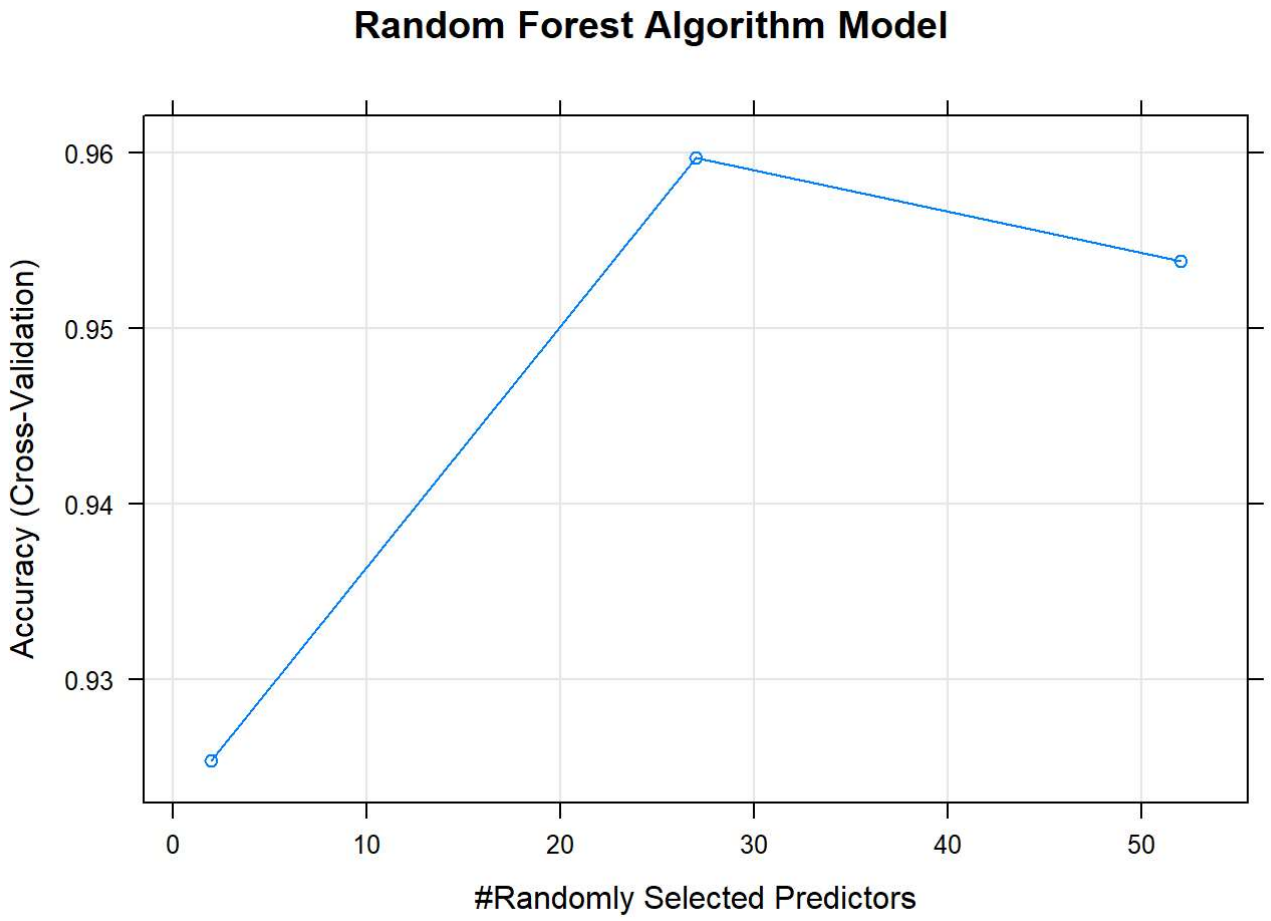
Appendix

- Github (https://github.com/busybee21/Practical-Machine-Learning)

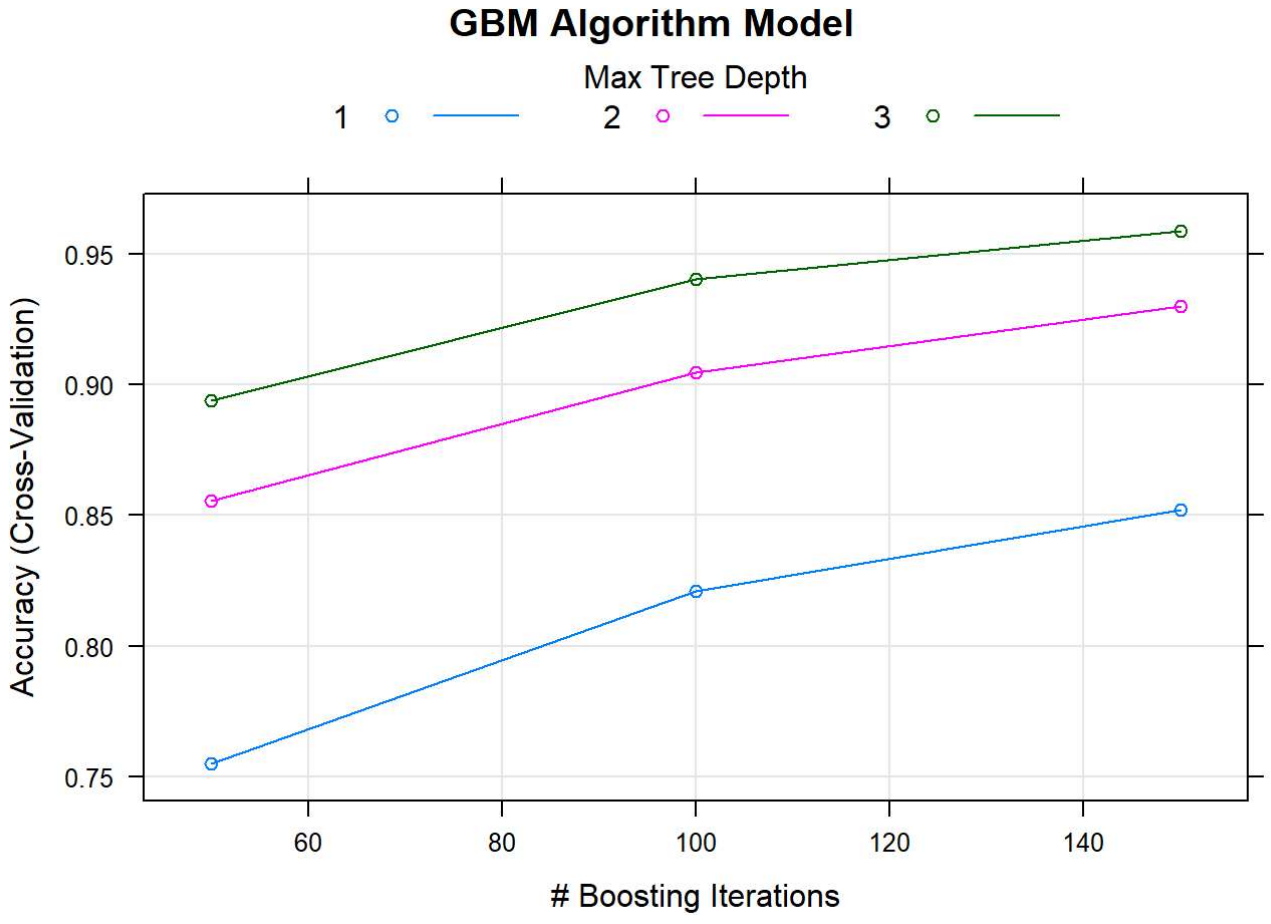
The Rmd file and the html file of this project can be found in the above Github repository.

Model visualization

```
par(mfrow=c(1,2))
plot(model.rf,main="Random Forest Algorithm Model")
```




```
plot(model.gbm,main="GBM Algorithm Model")
```



Code Chunk for the Comparison section

```
Algorithm=c("Decision Tree","Random Forest","GBM")
Accuracy=c(dt.mat$overall[1],rf.mat$overall[1],gbm.mat$overall[1])
Out.of.sample.Error=c(1-dt.mat$overall[1],1-rf.mat$overall[1],1-gbm.mat$overall[1])
viz=data.frame(Algorithm,Accuracy,Out.of.sample.Error)
viz
```