R Notebook

Name: Gabriel Bentley

Date: 10/21/22

Dataset: HTRU2 Pulsar classification

https://archive.ics.uci.edu/ml/datasets/HTRU2

Data set information

Pulsar candidates collected during the HTRU survey. Pulsars are a type of star, of considerable scientific interest. Candidates must be classified in to pulsar and non-pulsar classes to aid discovery.

Attribute information

Each candidate is described by 8 continuous variables, and a single class variable. The first four are simple statistics obtained from the integrated pulse profile (folded profile). This is an array of continuous variables that describe a longitude-resolved version of the signal that has been averaged in both time and frequency (see [3] for more details). The remaining four variables are similarly obtained from the DM-SNR curve

1. Mean of the integrated profile.
2. Standard deviation of the integrated profile.
3. Excess kurtosis of the integrated profile.
4. Skewness of the integrated profile.
5. Mean of the DM-SNR curve.
6. Standard deviation of the DM-SNR curve.
7. Excess kurtosis of the DM-SNR curve.
8. Skewness of the DM-SNR curve.
9. Class [0 = NON-PULSAR, 1 = PULSAR]

Read in the data set and rename columns to readable names

```r
library(e1071)
library(MASS)
library(mccr)

df <- read.csv("HTRU_2.csv")


names(df)[1] = "IPMean"
names(df)[2] = "IPSTD"
names(df)[3] = "IPExcessKurtosis"
names(df)[4] = "IPSkewness"
names(df)[5] = "DMSNRmean"
names(df)[6] = "DMSNRSTD"
names(df)[7] = "DMSNRExcesskurtosis"
names(df)[8] = "DMSNRSkewness"
names(df)[9] = "Class"
```

```
df$Class <- as.factor(x = df$Class)
names(df)
```

```
## [1] "IPMean"               "IPSTD"                "IPExcessKurtosis"
## [4] "IPSkewness"           "DMSNRmean"            "DMSNRSTD"
## [7] "DMSNRExcesskurtosis" "DMSNRSkewness"        "Class"
```

split the data set into train, test, and valid

```
set.seed(5050)
i <- sample(1:nrow(df), nrow(df)*0.80, replace=FALSE)
train <- df[i,]
test <- df[-i,]
```

Explore the data set

```
names(train)
```

```
## [1] "IPMean"               "IPSTD"                "IPExcessKurtosis"
## [4] "IPSkewness"           "DMSNRmean"            "DMSNRSTD"
## [7] "DMSNRExcesskurtosis" "DMSNRSkewness"        "Class"
```

```
summary(train)
```

```
##      IPMean            IPSTD       IPExcessKurtosis     IPSkewness
##  Min.   :  5.812   Min.   :24.90   Min.   :-1.87601   Min.   :-1.7919
##  1st Qu.:100.992   1st Qu.:42.39   1st Qu.: 0.02787   1st Qu.:-0.1895
##  Median :115.055   Median :46.95   Median : 0.22431   Median : 0.1983
##  Mean   :111.056   Mean   :46.56   Mean   : 0.48075   Mean   : 1.7889
##  3rd Qu.:126.984   3rd Qu.:51.05   3rd Qu.: 0.47169   3rd Qu.: 0.9262
##  Max.   :192.617   Max.   :98.78   Max.   : 8.06952   Max.   :68.1016
##    DMSNRmean          DMSNRSTD      DMSNRExcesskurtosis DMSNRSkewness
##  Min.   :  0.2132   Min.   :  7.37   Min.   :-3.139      Min.   :  -1.977
##  1st Qu.:  1.9331   1st Qu.: 14.45   1st Qu.: 5.798      1st Qu.:  35.202
##  Median :  2.7952   Median : 18.42   Median : 8.448      Median :  83.133
##  Mean   : 12.6411   Mean   : 26.34   Mean   : 8.298      Mean   : 104.709
##  3rd Qu.:  5.4356   3rd Qu.: 28.34   3rd Qu.:10.678      3rd Qu.: 138.698
##  Max.   :222.4214   Max.   :110.64   Max.   :34.540      Max.   :1191.001
##  Class
##  0:13018
##  1: 1299
##
##
##
##
```

```
str(train)
```

```
## 'data.frame':    14317 obs. of  9 variables:
##  $ IPMean             : num  140 112 110 130 122 ...
##  $ IPSTD              : num  53.5 46.7 55.7 45.3 51 ...
##  $ IPExcessKurtosis   : num  -0.13537 0.28257 0.30815 -0.0498 -0.00142 ...
##  $ IPSkewness         : num  -0.47519 0.2329 -0.16747 -0.00578 -0.14505 ...
##  $ DMSNRmean          : num  1.68 2.52 2.79 3.55 23.25 ...
##  $ DMSNRSTD           : num  11.1 14.7 19.1 21.1 58 ...
##  $ DMSNRExcesskurtosis: num  13.24 9.57 8.03 7.61 2.27 ...
##  $ DMSNRSkewness      : num  249.87 118.63 70.95 64.46 3.56 ...
##  $ Class              : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
```

```
head(train, n = 20)
```

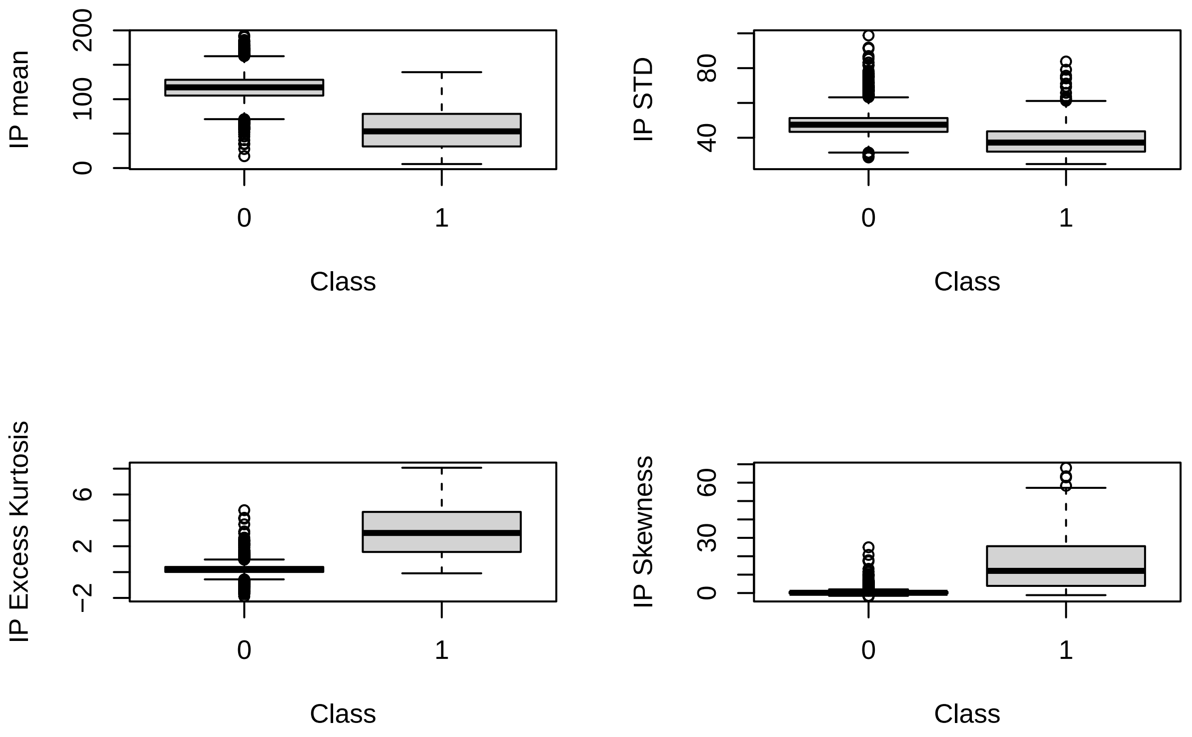
```
##         IPMean    IPSTD IPExcessKurtosis   IPSkewness  DMSNRmean DMSNRSTD
## 3421  139.7031 53.53071     -0.135372880 -0.475185250   1.675585 11.06329
## 13595 112.3047 46.73890      0.282571349  0.232895256   2.524247 14.67628
## 16717 109.8750 55.66305      0.308153932 -0.167467369   2.792642 19.06157
## 14475 130.3125 45.25427     -0.049795135 -0.005775005   3.549331 21.14412
## 2409  121.9141 50.96307     -0.001420464 -0.145053089  23.245819 58.04228
## 14749 108.0469 56.68276      0.486172280 -0.278363449   2.693144 16.75673
## 2129  105.0234 49.66929      0.216725221 -0.095071098   1.994983 13.65307
## 11709 117.3906 47.24822      0.336201110  0.120427217   2.479933 19.67687
## 5008  117.9609 54.53631      0.042907526 -0.470556202   2.087793 15.22336
## 2257  124.4141 49.67166      0.233147455 -0.007355501   2.065217 17.55995
## 10133 174.4453 54.56058     -0.967856050  0.153688727  88.736622 93.30741
## 10981 140.5391 50.28620     -0.381446574 -0.248177818 127.956522 76.14583
## 15115 121.3359 47.25682      0.232933200 -0.106721627   2.407191 20.00683
## 8075  132.4766 48.43552     -0.254866985 -0.413368210   1.688963 13.57551
## 8337  123.7812 51.49514     -0.016499516 -0.189112215   2.865385 21.94340
## 7606  115.6484 57.50361      0.398611080 -0.340352463  52.783445 69.81927
## 2017  125.7734 48.12877     -0.167786773 -0.115305503   3.638796 25.58156
## 1022  120.2969 45.27354      0.122069871  0.033778110   3.924749 22.85371
## 9979  159.3281 55.45101     -0.696563295 -0.088413471 168.069398 82.06931
## 1794  130.7031 51.84071     -0.127221522 -0.275435121   5.358696 27.22173
##       DMSNRExcesskurtosis DMSNRSkewness Class
## 3421           13.2441638   249.8742273     0
## 13595           9.5681179   118.6343753     0
## 16717           8.0342736    70.9539162     0
## 14475           7.6053810    64.4604129     0
## 2409            2.2660604     3.5636064     0
## 14749           8.7790844    91.7005879     0
## 2129           11.0978825   160.7824000     0
## 11709           9.0645640    87.6977922     0
## 5008            9.6738169   112.0748913     0
## 2257            9.3766258    94.0265727     0
## 10133           0.3386946    -1.6194894     0
## 10981          -0.5537850    -0.8480723     0
## 15115           9.1781865    88.5974695     0
## 8075           10.9207674   146.0245159     0
## 8337            8.5006663    75.8727589     0
## 7606            1.0246872    -0.1627685     1
## 2017            7.5435163    58.2367928     0
## 1022            7.3832008    59.5033961     0
## 9979           -0.8884102    -0.5767806     0
## 1794            5.5843948    32.3056860     0
```

```
colSums(is.na(train))
```

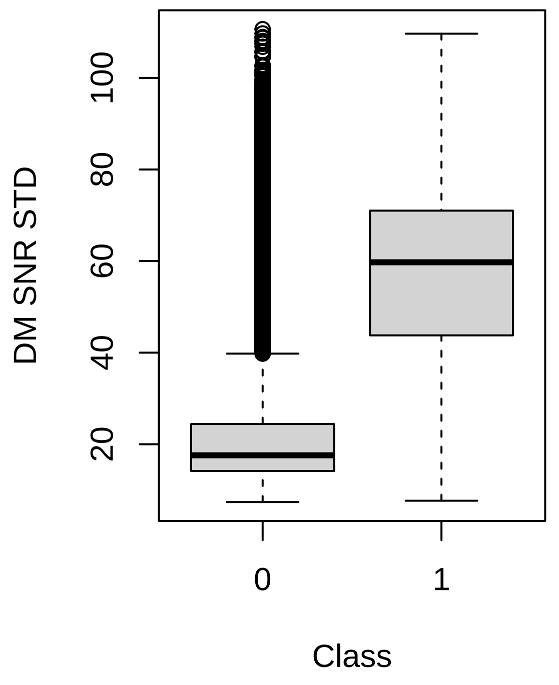
```
##              IPMean               IPSTD    IPExcessKurtosis          IPSkewness
##                   0                   0                   0                   0
##           DMSNRmean            DMSNRSTD DMSNRExcesskurtosis       DMSNRSkewness
##                   0                   0                   0                   0
##               Class
##                    0
```

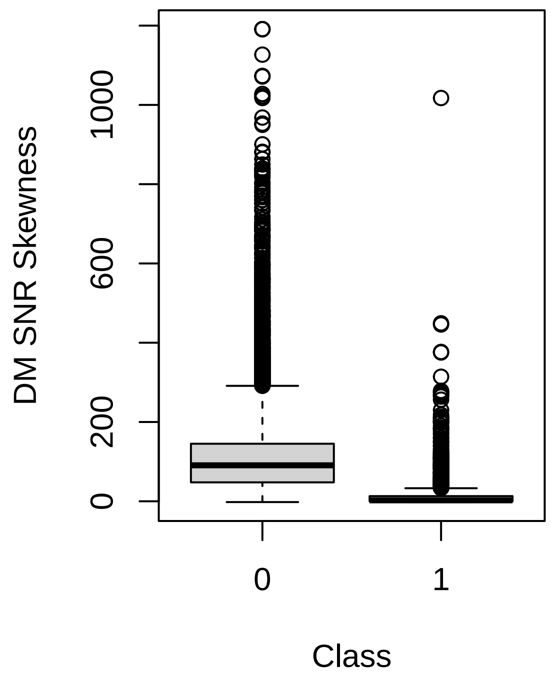
Graphically explore the data set

```
par(mfrow=c(2,2))
plot(train$Class, train$IPMean, xlab="Class", ylab="IP mean")
plot(train$Class, train$IPSTD, xlab="Class", ylab="IP STD")
plot(train$Class, train$IPExcessKurtosis, xlab="Class", ylab="IP Excess Kurtosis")
plot(train$Class, train$IPSkewness, xlab="Class", ylab="IP Skewness")
```



```
par(mfrow=c(1,2))
plot(train$Class, train$DMSNRSTD, xlab="Class", ylab="DM SNR STD")
plot(train$Class, train$DMSNRSkewness, xlab="Class", ylab="DM SNR Skewness")
```





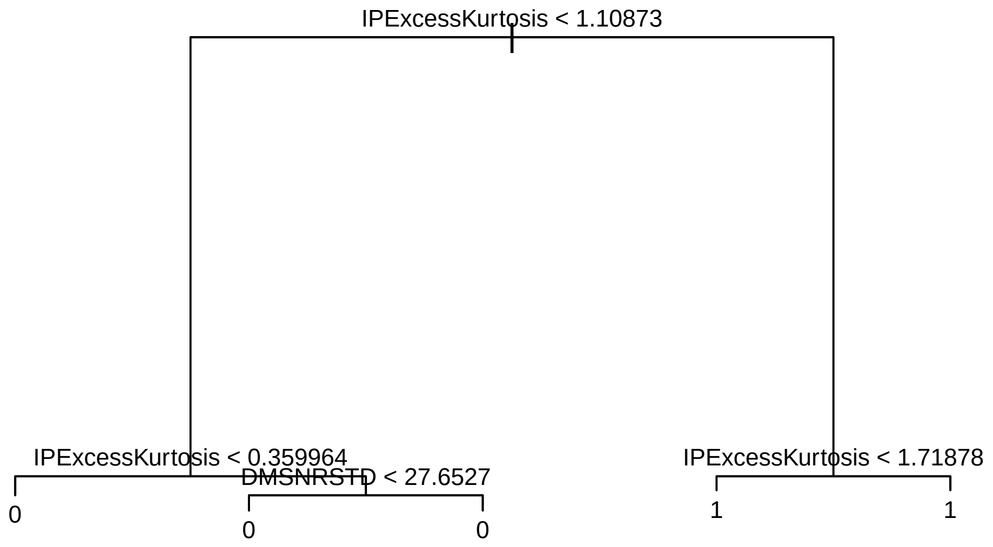
Decision Tree

```
library(tree)
start_time <- Sys.time()
tree1 <- tree(Class~., data=train)
summary(tree1)
```

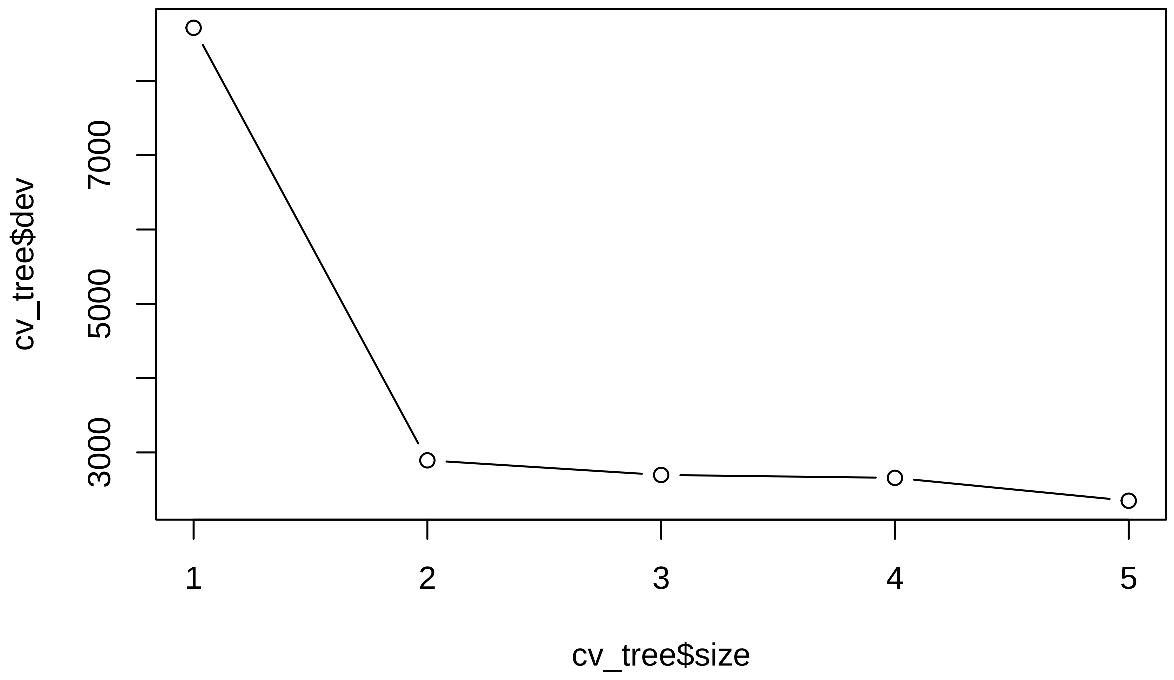
```
##
## Classification tree:
## tree(formula = Class ~ ., data = train)
## Variables actually used in tree construction:
## [1] "IPExcessKurtosis" "DMSNRSTD"
## Number of terminal nodes:  5
## Residual mean deviance:  0.1571 = 2248 / 14310
## Misclassification error rate: 0.02235 = 320 / 14317
```

```
end_time <- Sys.time()

plot(tree1)
text(tree1, cex=0.75, pretty=0)
```


IPExcessKurtosis < 1.10873
IPExcessKurtosis < 0.359964
DMSNRSTD < 27.6527
IPExcessKurtosis < 1.71878
0
0
0
1
1

```r
cv_tree <- cv.tree(tree1)
plot(cv_tree$size, cv_tree$dev, type='b')
```



```r
predDT <- predict(tree1, newdata=test, type="class")

table(predDT, test$Class)
```

```
##
## predDT    0    1
##      0 3205   56
##      1   35  284
```

```r
acc <- mean(predDT==test$Class)
mcc <- mccr(factor(predDT), test$Class)

print(paste("accuracy=", acc))
```

```
## [1] "accuracy= 0.974581005586592"
```

```
print(paste("mcc=", mcc))
```

```
## [1] "mcc= 0.848452402207317"
```

```
print(paste("time taken=", (end_time - start_time), "seconds"))
```

```
## [1] "time taken= 0.151061058044434 seconds"
```

Random Forest

```
library(randomForest)
```

```
## randomForest 4.7-1.1

## Type rfNews() to see new features/changes/bug fixes.
```

```
set.seed(5050)

start_time <- Sys.time()
rf <- randomForest(Class~., data=train, importance=TRUE)
rf
```

```
##
## Call:
##  randomForest(formula = Class ~ ., data = train, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 2.01%
## Confusion matrix:
##       0    1 class.error
## 0 12935   83 0.006375787
## 1   205 1094 0.157813703
```

```
end_time <- Sys.time()

pred <-  predict(rf, newdata=test)
acc <- mean(pred==test$Class)
mcc <- mccr(factor(pred), test$Class)
table(pred, test$Class)
```

```
##
## pred    0    1
##    0 3220   52
##    1   20  288
```

```
print(paste("accuracy=", acc))
```

```
## [1] "accuracy= 0.979888268156425"
```

```
print(paste("mcc=", mcc))
```

```
## [1] "mcc= 0.879158281747481"
```

```
print(paste("time taken=", (end_time - start_time), "seconds"))
```

```
## [1] "time taken= 22.1703000068665 seconds"
```

XGBoost

Had to create a data matrix for the non Class values of the data set and a numeric vector for the Class column. Originally when converting the vector to as.numeric it set 0 to 1 and 1 to 2, so I needed to subtract 1 from all values so it would work with the xgboost function.

```
library(xgboost)
xgtrain <- train[,-9]
xgtest <- test[,-9]

trainLable <- as.numeric(train$Class)
testLable <- as.numeric(test$Class)
trainLable <- trainLable - 1
testLable <- testLable - 1

xgtrain <- as.matrix(xgtrain)
xgtest <- as.matrix(xgtest)

class(trainLable)
```

```
## [1] "numeric"
```

```
class(xgtrain)
```

```
## [1] "matrix" "array"
```

```
start_time <- Sys.time()
model <- xgboost(data=xgtrain, label=as.vector(trainLable),
                 nrounds=2, objective='binary:logistic')
```

```
## [1]  train-logloss:0.453122
## [2]  train-logloss:0.320678
```

```
end_time <- Sys.time()

pred <- predict(model, xgtest)
pred <- ifelse(pred>0.5, 1, 0)
table(pred, testLable)
```

```
##     testLable
## pred    0    1
##    0 3217   53
##    1   23  287
```

```
acc <- mean(pred == testLable)
mcc <- mccr(factor(pred), test$Class)

print(paste("accuracy=", acc))
```

```
## [1] "accuracy= 0.97877094972067"
```

```
print(paste("mcc=", mcc))
```

```
## [1] "mcc= 0.872554364869883"
```

```
print(paste("time taken=", (end_time - start_time), "seconds"))
```

```
## [1] "time taken= 0.0789361000061035 seconds"
```

adaboost

```r
library(adabag)
```

```
## Loading required package: rpart

## Loading required package: caret

## Loading required package: ggplot2

##
## Attaching package: 'ggplot2'

## The following object is masked from 'package:randomForest':
##
##     margin

## Loading required package: lattice

## Loading required package: foreach

## Loading required package: doParallel

## Loading required package: iterators

## Loading required package: parallel
```

```r
library(mccr)
start_time <- Sys.time()
adab1 <- boosting(Class~., data=train, boos=TRUE, mfinal=20,
       coeflearn='Breiman')
end_time <- Sys.time()

summary(adab1)
```

```
##            Length Class   Mode
## formula        3  formula call
## trees         20  -none-  list
## weights       20  -none-  numeric
## votes      28634  -none-  numeric
## prob       28634  -none-  numeric
## class      14317  -none-  character
## importance     8  -none-  numeric
## terms          3  terms   call
## call           6  -none-  call
```

```r
pred <- predict(adab1, newdata=test, type="response")
acc <- mean(pred$class==test$Class)
mcc <- mccr(factor(pred$class), test$Class)

print(paste("accuracy=", acc))
```

```
## [1] "accuracy= 0.977932960893855"
```

```r
print(paste("mcc=", mcc))
```

```
## [1] "mcc= 0.868775125656206"
```

```r
print(paste("time taken=", (end_time - start_time), "minutes"))
```

```
## [1] "time taken= 24.1990840435028 minutes"
```

Analysis of the results

Decision tree The decision tree acting as a baseline for predicting if a given star was a Pulsar or a non-pulsar had an accuracy of 97.46% and a mcc of 84.84% making it an accurate model. The time taken for the model to be generated was 0.15 seconds making it a very fast model.

Random Forest The random forest algorithm had an accuracy of 97.99% and a mcc of 87.92% making it more accurate than the base decision tree model for the data. However the time taken for the random forest algorithm was much higher than the time needed to make the decision tree, being 32 seconds.

xgboost The xgboost algorithm had an accuracy of 97.88% and a mcc of 87.26% making it almost exactly as accurate as the random forest algorithm. The big advantage the xgboost algorithm has over the Random Forest algorithm is that it took far less time to generate, with 0.13 seconds it was even faster than the decision tree algorithm.

adaboost The adaboost algorithm had an accuracy of 97.79% and a mcc of 86.88% giving it a similar accuracy to both the random forest and the xgboost algorithms. The time taken for the adaboost algorithm was the longest by far with 80 seconds needed to generate the model over a minute.

Conclusion Of the four different models all of them were accurate at predicting if a star was Pulsar or Non-Pulsar, with the three ensemble models being slightly more accurate. I assume that the data set I use was easy to predict on for several reasons, one there were a lot of data instance, two there were clear differences in the predictor values for a Pulsar and Non-Pulsar star, three the models were doing binary classification with Non-Pulsar having more instances than Pulsar. Since the models are all close in accuracy the deciding factor on which model is the best for the data set comes down to run time and of the three ensemble models xgboost had the lowest run time of them all. It is possible that since the data set is doing binary classification I could have used fast adaboost to increase the execution speed of the adaboost model by around 100 times.