R Notebook

Name: Gabriel Bentley

Date: 10/20/22

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

```
library(e1071)
library(MASS)

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(8834)
spec <- c(train=.6, test=.2, validate=.2)
i <- sample(cut(1:nrow(df),
                nrow(df)*cumsum(c(0,spec)), labels=names(spec)))
train <- df[i=="train",]
test <- df[i=="test",]
vald <- df[i=="validate",]
```

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.   :24.77   Min.   :-1.87601   Min.   :-1.7819
##  1st Qu.:100.814   1st Qu.:42.34   1st Qu.: 0.02931   1st Qu.:-0.1835
##  Median :114.926   Median :46.92   Median : 0.22727   Median : 0.2031
##  Mean   :110.989   Mean   :46.49   Mean   : 0.48040   Mean   : 1.7772
##  3rd Qu.:127.146   3rd Qu.:50.95   3rd Qu.: 0.47600   3rd Qu.: 0.9352
##  Max.   :192.617   Max.   :98.78   Max.   : 7.87963   Max.   :65.3860
##    DMSNRmean          DMSNRSTD      DMSNRExcesskurtosis DMSNRSkewness
##  Min.   :  0.2132   Min.   :  7.37   Min.   :-3.139      Min.   :  -1.949
##  1st Qu.:  1.9214   1st Qu.: 14.42   1st Qu.: 5.793      1st Qu.:  35.253
##  Median :  2.7885   Median : 18.41   Median : 8.451      Median :  83.405
##  Mean   : 12.4363   Mean   : 26.22   Mean   : 8.316      Mean   : 104.909
##  3rd Qu.:  5.4586   3rd Qu.: 28.24   3rd Qu.:10.703      3rd Qu.: 139.300
##  Max.   :223.3921   Max.   :109.71   Max.   :34.540      Max.   :1191.001
##  Class
##  0:9764
##  1: 974
##
##
##
##
```

```
str(train)
```

```
## 'data.frame':    10738 obs. of  9 variables:
##  $ IPMean              : num  119 114 110 101 136 ...
##  $ IPSTD               : num  48.8 51.9 49 51.7 51.7 ...
##  $ IPExcessKurtosis    : num  0.0315 -0.0945 0.1376 0.3938 -0.0459 ...
##  $ IPSkewness          : num  -0.1122 -0.288 -0.2567 -0.0112 -0.2718 ...
##  $ DMSNRmean           : num  0.999 2.738 1.508 2.841 9.343 ...
##  $ DMSNRSTD            : num  9.28 17.19 12.07 21.64 38.1 ...
```

```
##  $ DMSNRExcesskurtosis: num  19.21 9.05 13.37 8.3 4.35 ...
##  $ DMSNRSkewness      : num  479.8 96.6 223.4 71.6 18.7 ...
##  $ Class              : Factor w/ 2 levels "0","1": 1 1 1 1 1 2 1 1 1 1 ...
```

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

```
##       IPMean    IPSTD IPExcessKurtosis  IPSkewness  DMSNRmean  DMSNRSTD
## 6  119.48438 48.76506      0.031460220 -0.11216757  0.9991639  9.279612
## 15 114.36719 51.94572     -0.094498904 -0.28798409  2.7382943 17.191891
## 16 109.64062 49.01765      0.137635830 -0.25669978  1.5083612 12.072901
## 17 100.85156 51.74352      0.393836792 -0.01124074  2.8411371 21.635778
## 18 136.09375 51.69100     -0.045908926 -0.27181639  9.3428094 38.096400
## 19  99.36719 41.57220      1.547196967  4.15410604 27.5551839 61.719016
## 20 100.89062 51.89039      0.627486528 -0.02649780  3.8837793 23.045267
## 21 105.44531 41.13997      0.142653801  0.32041968  3.5518395 20.755017
## 23 117.36719 53.90861      0.257953441 -0.40504908  6.0183946 24.766123
## 25 112.71875 50.30127      0.279390953 -0.12901071  8.2817726 37.810012
## 27 119.43750 52.87482     -0.002549267 -0.46036029  2.3653846 16.498032
## 28 123.21094 51.07801      0.179376819 -0.17728516  2.1070234 16.921773
## 29 102.61719 49.69235      0.230438984  0.19332537  1.4891304 16.004411
## 30 110.10938 41.31817      0.094860398  0.68311261  1.0100334 13.026275
## 31  99.91406 43.91950      0.475728501  0.78148620  0.6195652  9.440976
## 32 128.34375 52.17211     -0.049280401 -0.20825699  2.1739130 12.993947
## 34 121.13281 47.63261      0.177360308  0.02491811  2.1513378 20.552437
## 35 102.32812 48.98040      0.315729409 -0.20218332  1.8988294 13.839040
## 38 107.87500 37.33066      0.496004760  1.48181586  1.1739130 12.016913
## 39 118.84375 45.93192     -0.109242666  0.13768355  2.3327759 14.716029
##    DMSNRExcesskurtosis DMSNRSkewness Class
## 6            19.206230     479.75657     0
## 15            9.050612      96.61190     0
## 16           13.367926     223.43842     0
## 17            8.302242      71.58437     0
## 18            4.345438      18.67365     0
## 19            2.208808       3.66268     1
## 20            6.953168      52.27944     0
## 21            7.739552      68.51977     0
## 23            4.807783      25.52262     0
## 25            4.691827      21.27621     0
## 27            9.008352      94.75566     0
## 28           10.080333     112.55859     0
## 29           12.646535     171.83290     0
## 30           14.666511     231.20414     0
## 31           20.106639     475.68022     0
## 32            9.965757     141.51008     0
## 34            9.920468      99.74708     0
## 35           11.619939     172.13037     0
## 38           14.534290     252.69474     0
## 39            9.634175     118.66968     0
```

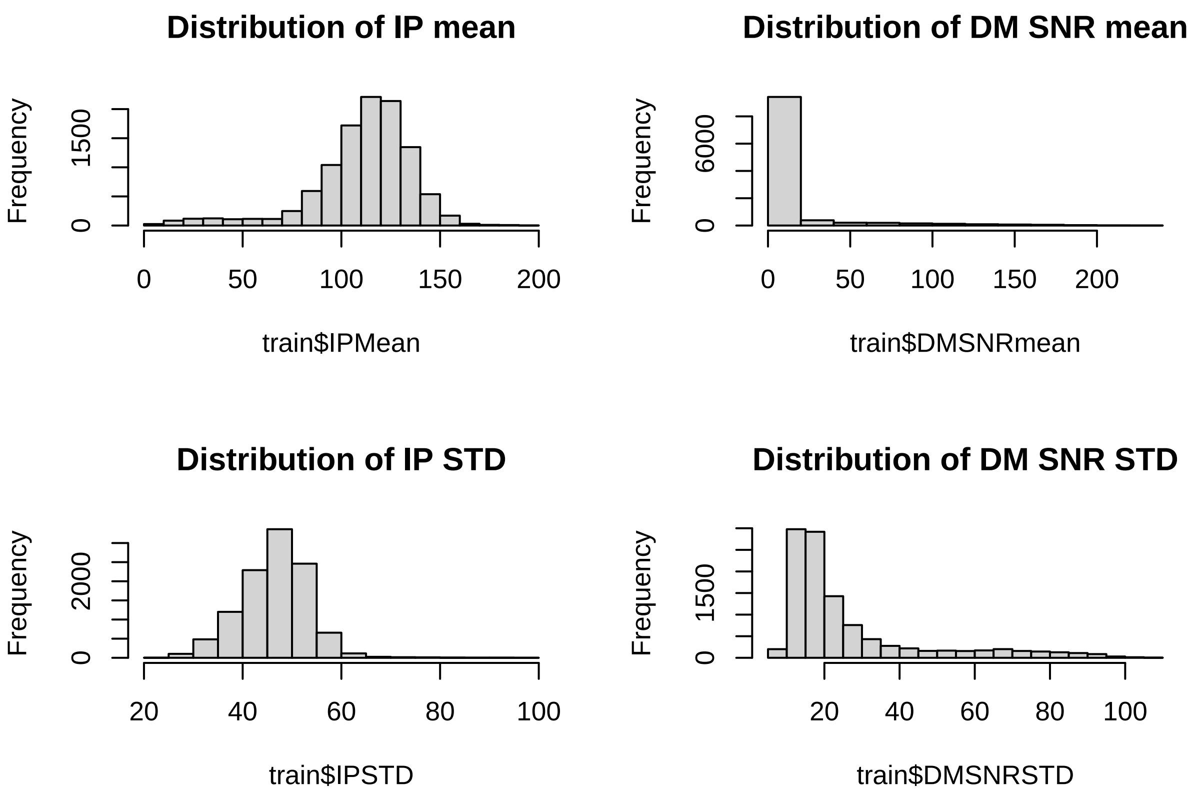
```
colSums(is.na(train))
```

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

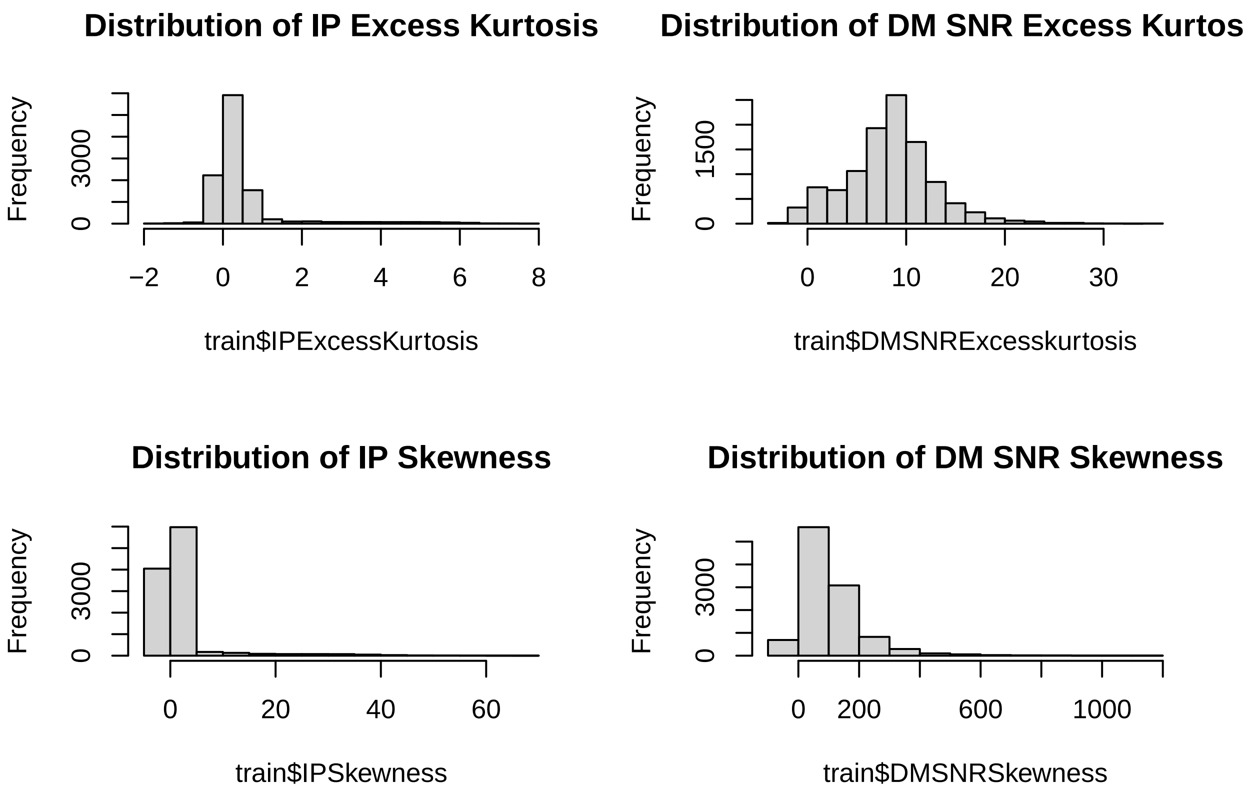
```
##                          0
```

Graphically explore the data set

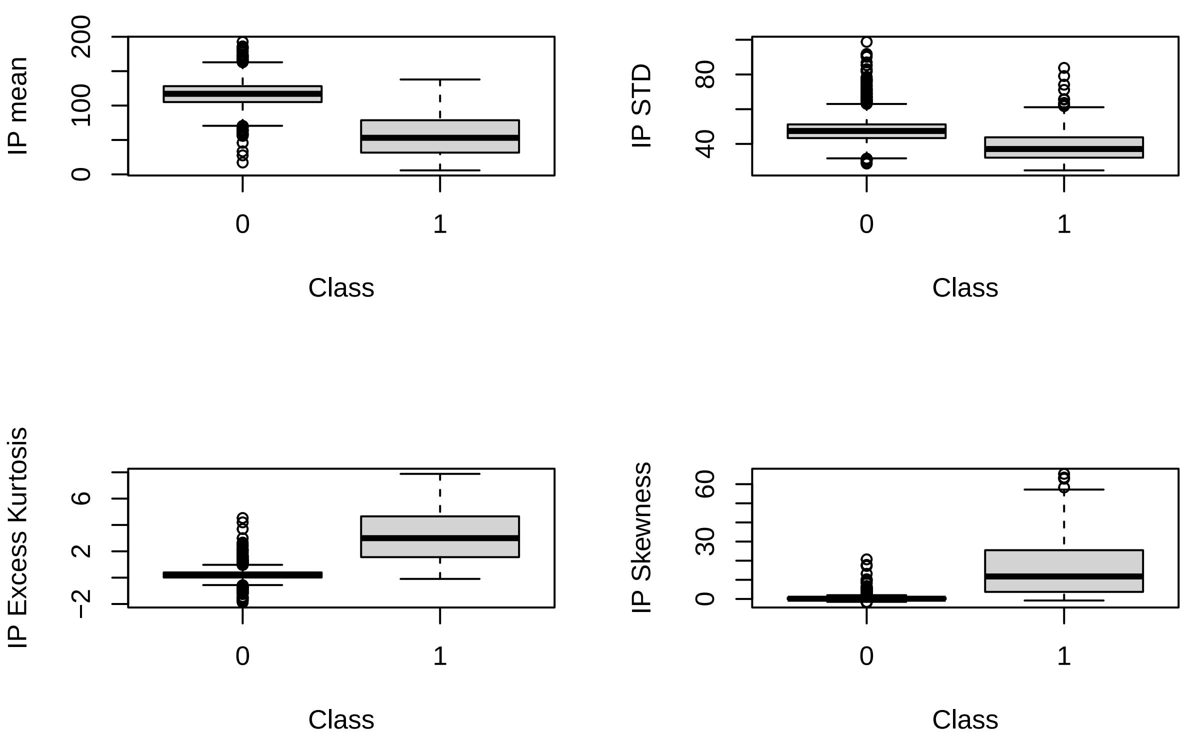
```
par(mfrow=c(2,2))
hist(train$IPMean, main="Distribution of IP mean")
hist(train$DMSNRmean, main="Distribution of DM SNR mean")
hist(train$IPSTD, main="Distribution of IP STD")
hist(train$DMSNRSTD, main="Distribution of DM SNR STD")
```



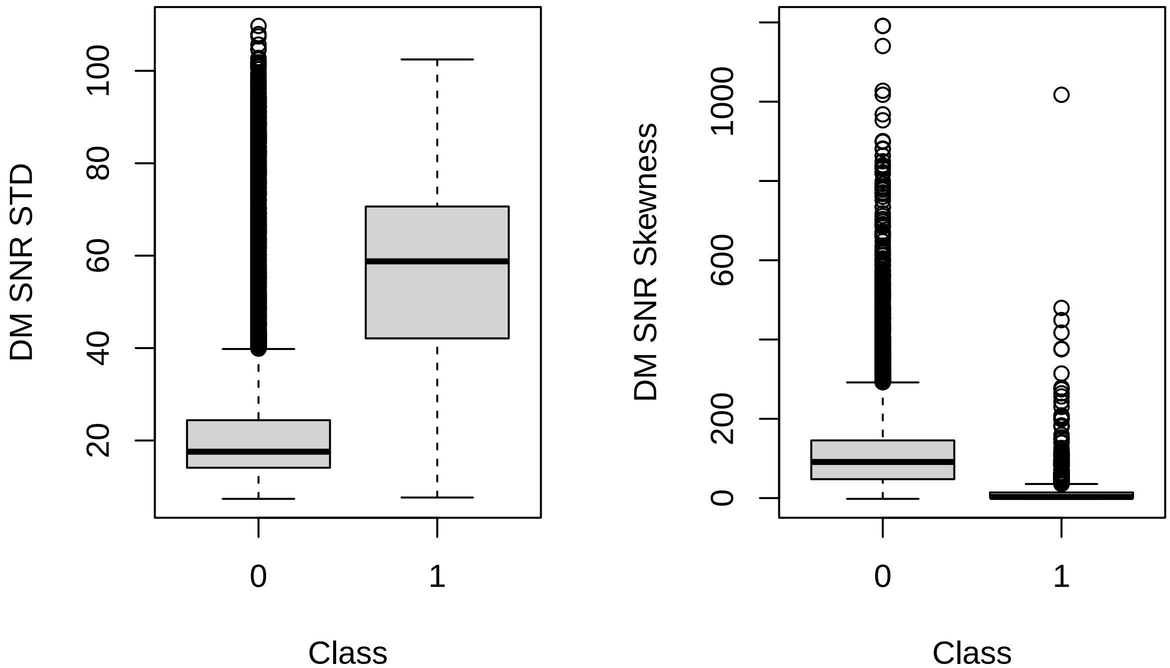

```
hist(train$IPExcessKurtosis, main="Distribution of IP Excess Kurtosis")
hist(train$DMSNRExcesskurtosis, main="Distribution of DM SNR Excess Kurtosis")
hist(train$IPSkewness, main="Distribution of IP Skewness")
hist(train$DMSNRSkewness, main="Distribution of DM SNR Skewness")
```




```r
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")
```



```r
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")
```



Regular Logistic Regression

```r
library(caret)
```

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

```r
library(mccr)

glm1 <- glm(Class~., data=train, family = "binomial")
summary(glm1)
```

```
##
## Call:
## glm(formula = Class ~ ., family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.3682  -0.1679  -0.1025  -0.0584   3.6112
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -8.511237   1.259485  -6.758 1.40e-11 ***
## IPMean               0.030294   0.007626   3.972 7.12e-05 ***
## IPSTD               -0.034884   0.013851  -2.519   0.0118 *
## IPExcessKurtosis     6.546618   0.393291  16.646  < 2e-16 ***
## IPSkewness          -0.602479   0.057578 -10.464  < 2e-16 ***
## DMSNRmean           -0.026916   0.004198  -6.412 1.44e-10 ***
```

```
## DMSNRSTD              0.044763   0.009591   4.667 3.06e-06 ***
## DMSNRExcesskurtosis -0.010765   0.105702  -0.102   0.9189
## DMSNRSkewness       -0.003021   0.003608  -0.837   0.4024
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 6532.3  on 10737  degrees of freedom
## Residual deviance: 1587.7  on 10729  degrees of freedom
## AIC: 1605.7
##
## Number of Fisher Scoring iterations: 8
```

```
probsLR <- predict(glm1, newdata=test, type="response")
predLR <- ifelse(probsLR>0.5, 1, 0)


table(predLR, test$Class)
```

```
##
## predLR    0    1
##      0 3242   58
##      1   23  256
```

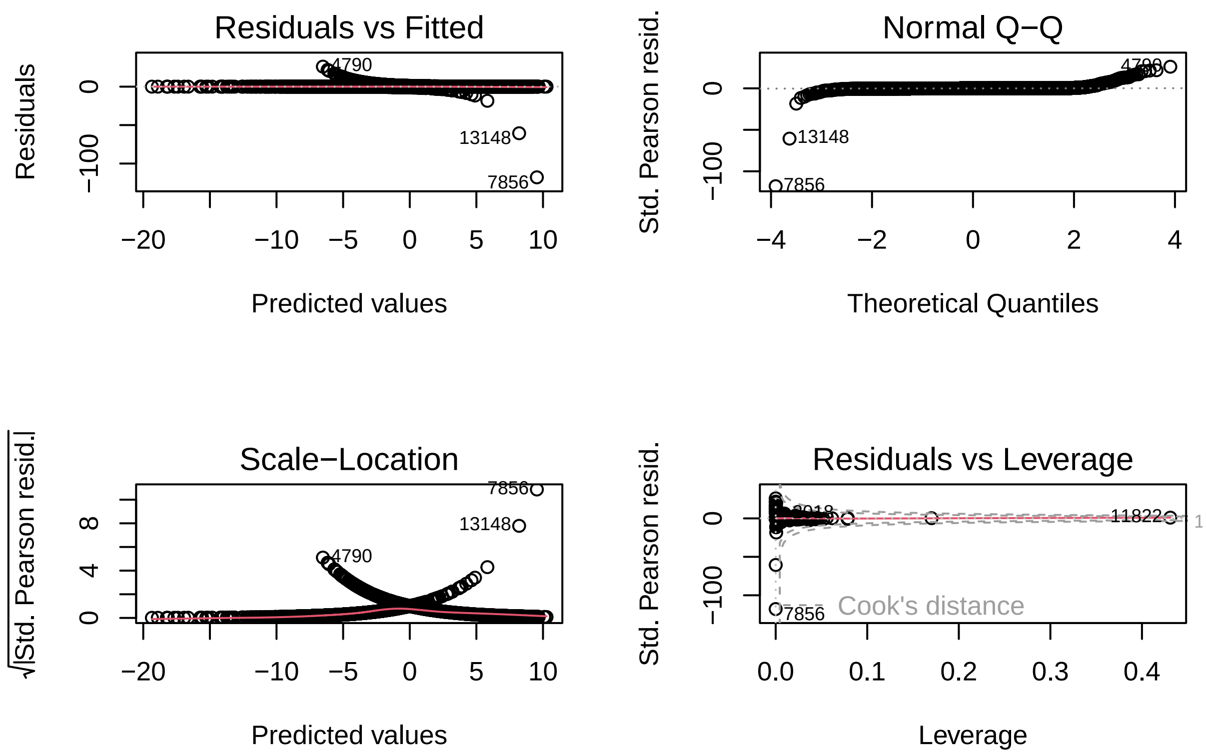
```
acc <- mean(predLR == test$Class)
mcc <- mccr(factor(predLR), test$Class)

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

```
## [1] "accuracy= 0.977367979882649"
```

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

```
## [1] "mcc= 0.852881854735329"
```

```
par(mfrow=c(2,2))
plot(glm1)
```



SVM Classification with linear kernels

```
svm1 <- svm(Class~., data=train, kernel="linear", cost=5, scale=TRUE)
summary(svm1)
```

```
##
## Call:
## svm(formula = Class ~ ., data = train, kernel = "linear", cost = 5,
##     scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  5
##
## Number of Support Vectors:  581
##
##  ( 291 290 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

```
pred <- predict(svm1, newdata=test)
table(pred, test$Class)
```

```
##
```

```
## pred    0    1
##    0 3243   62
##    1   22  252
```

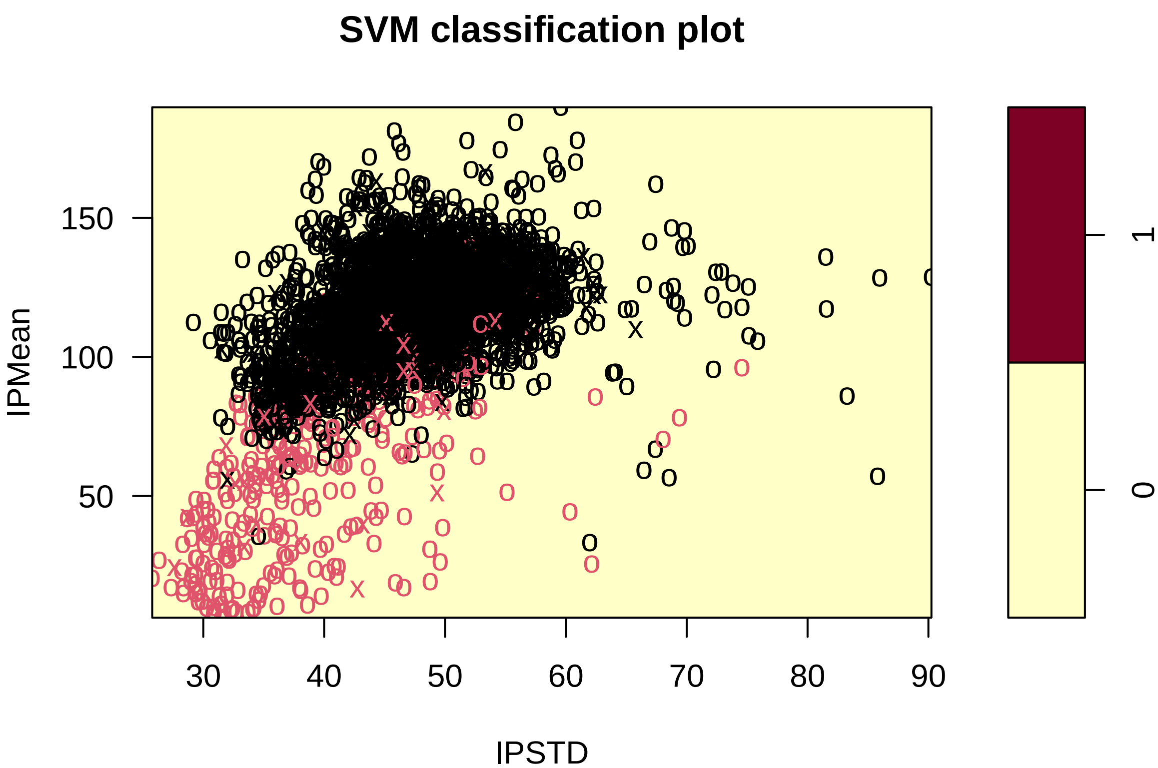
```
acc <- mean(pred == test$Class)
mcc <- mccr(factor(pred), test$Class)

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

```
## [1] "accuracy= 0.976529756915339"
```

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

```
## [1] "mcc= 0.846748817120572"
```

```
plot(svm1, test, IPMean~IPSTD)
```



##

Tune SVM for linear

```
tune_svm1 <- tune(svm, Class~., data=vald, kernel="linear",
                  ranges=list(cost=c(0.001, 0.01, 0.1, 1, 5, 10, 100)))
summary(tune_svm1)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##     5
##
## - best performance: 0.01787709
```

```
##
## - Detailed performance results:
##    cost      error  dispersion
## 1 1e-03 0.03938547 0.010152819
## 2 1e-02 0.02737430 0.008095741
## 3 1e-01 0.02067039 0.006740023
## 4 1e+00 0.01843575 0.004406763
## 5 5e+00 0.01787709 0.004784072
## 6 1e+01 0.01815642 0.004793124
## 7 1e+02 0.01815642 0.004215729
```

SVM Classification with polynomial kernels

```r
svm1 <- svm(Class~., data=train, kernel="polynomial", cost=100, scale=TRUE)
summary(svm1)
```

```
##
## Call:
## svm(formula = Class ~ ., data = train, kernel = "polynomial", cost = 100,
##     scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  100
##      degree:  3
##      coef.0:  0
##
## Number of Support Vectors:  535
##
##  ( 274 261 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

```r
pred <- predict(svm1, newdata=test)

table(pred, test$Class)
```

```
##
## pred    0    1
##    0 3238   58
##    1   27  256
```

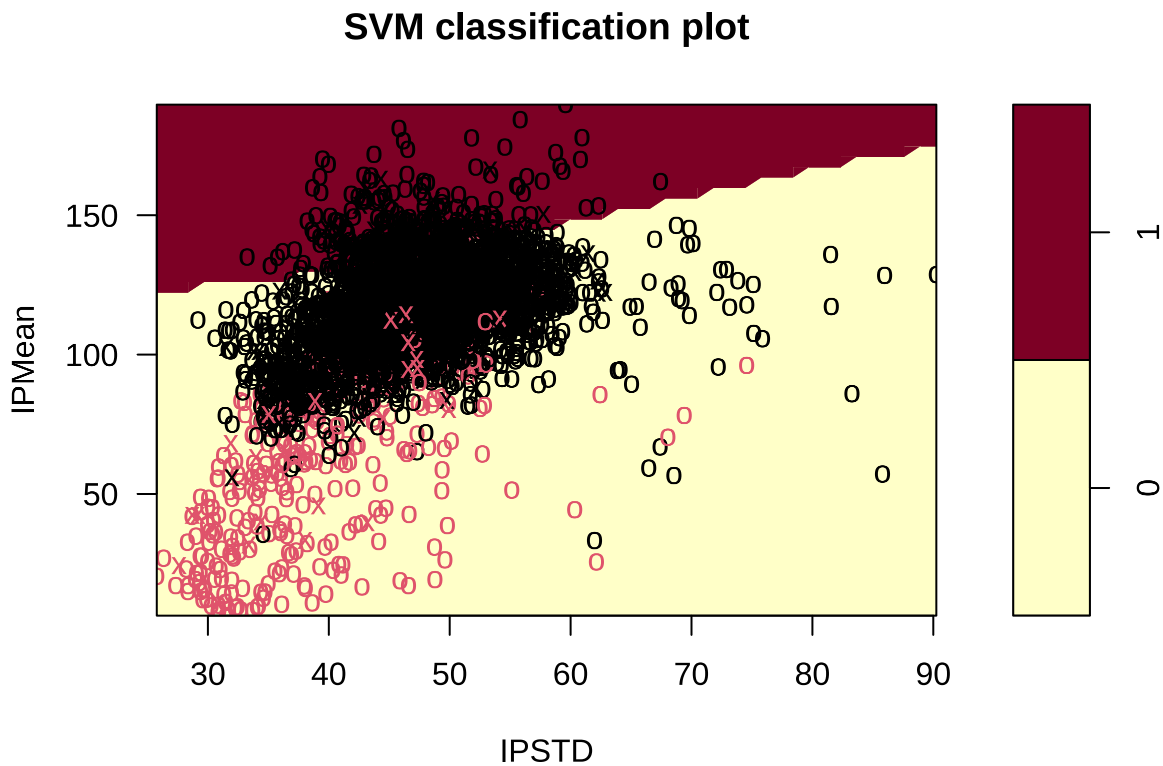
```r
acc <- mean(pred == test$Class)
mcc <- mccr(factor(pred), test$Class)

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

```
## [1] "accuracy= 0.97625034925957"
```

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

```
## [1] "mcc= 0.846062281584738"
```

```
plot(svm1, test, IPMean~IPSTD)
```



##

Tune for SVM Polynomial

```
tune_svm1 <- tune(svm, Class~., data=vald, kernel="polynomial",
                  ranges=list(cost=c(0.001, 0.01, 0.1, 1, 5, 10, 100)))
summary(tune_svm1)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##   100
##
## - best performance: 0.02150838
##
## - Detailed performance results:
##     cost      error  dispersion
## 1 1e-03 0.04664804 0.010538293
## 2 1e-02 0.03603352 0.009070450
## 3 1e-01 0.03072626 0.006714248
## 4 1e+00 0.02430168 0.004570927
## 5 5e+00 0.02318436 0.004935701
## 6 1e+01 0.02262570 0.005175770
```

```
## 7 1e+02 0.02150838 0.008015024
```

SVM Classification with radial kernels

```r
svm1 <- svm(Class~., data=train, kernel="radial", cost=10, gamma = 0.5, scale=TRUE)
summary(svm1)
```

```
##
## Call:
## svm(formula = Class ~ ., data = train, kernel = "radial", cost = 10,
##     gamma = 0.5, scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  10
##
## Number of Support Vectors:  839
##
##  ( 503 336 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

```r
pred <- predict(svm1, newdata=test)


table(pred, test$Class)
```

```
##
## pred    0    1
##    0 3242   54
##    1   23  260
```

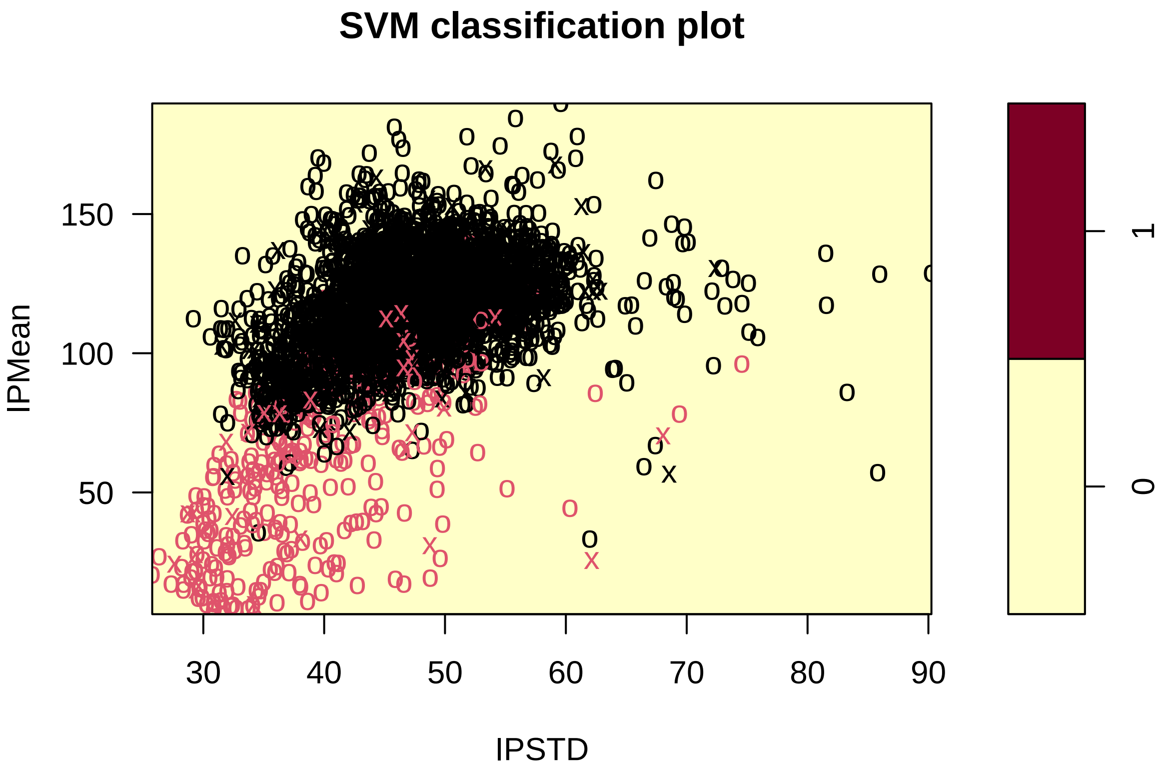
```r
acc <- mean(pred == test$Class)
mcc <- mccr(factor(pred), test$Class)

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

```
## [1] "accuracy= 0.978485610505728"
```

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

```
## [1] "mcc= 0.860701856067451"
```

```r
plot(svm1, test, IPMean~IPSTD)
```



Tune

```
set.seed(6229)
tune.out <- tune(svm, Class~., data=vald, kernel="radial",
                 ranges=list(cost=c(0.1,1,10,100,1000),
                             gamma=c(0.5,1,2,3,4)))
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##    10   0.5
##
## - best performance: 0.01899441
##
## - Detailed performance results:
##     cost gamma      error  dispersion
## 1  1e-01   0.5 0.02681564 0.009329590
## 2  1e+00   0.5 0.01927374 0.007266150
## 3  1e+01   0.5 0.01899441 0.006942775
## 4  1e+02   0.5 0.02234637 0.007212259
## 5  1e+03   0.5 0.03463687 0.008244299
## 6  1e-01   1.0 0.05335196 0.013901151
## 7  1e+00   1.0 0.01983240 0.007728680
## 8  1e+01   1.0 0.02206704 0.007615682
```

```
## 9  1e+02   1.0 0.02737430 0.007987937
## 10 1e+03   1.0 0.03854749 0.010267448
## 11 1e-01   2.0 0.09804469 0.016886034
## 12 1e+00   2.0 0.02430168 0.009680740
## 13 1e+01   2.0 0.02960894 0.008451997
## 14 1e+02   2.0 0.03743017 0.008451997
## 15 1e+03   2.0 0.04078212 0.010384986
## 16 1e-01   3.0 0.09804469 0.016886034
## 17 1e+00   3.0 0.03463687 0.009783185
## 18 1e+01   3.0 0.03798883 0.009694164
## 19 1e+02   3.0 0.04860335 0.008244299
## 20 1e+03   3.0 0.04888268 0.007928020
## 21 1e-01   4.0 0.09804469 0.016886034
## 22 1e+00   4.0 0.04441341 0.009626858
## 23 1e+01   4.0 0.04581006 0.009604318
## 24 1e+02   4.0 0.05474860 0.008553955
## 25 1e+03   4.0 0.05474860 0.008553955
```

Analysis of the results

Logistical regression The accuracy for the logistic regression algorithm was 97.74% with a mcc of 85.29% making the model a very accurate baseline.

SVM Linear Kernel The accuracy for SVM using a linear kernel was 97.65% with a mcc of 84.67% very close to the logistical regression model.

SVM Polynomial Kernel The accuracy for SVM using a polynomial kernel was 97.63% with a mcc of 84.61% being the tiniest bit lower in accuracy than the SVM with a linear kernel.

SVM Radial Kernel The accuracy for SVM using a radial kernel was 97.85% with an mcc of 86.07% having the highest accuracy out of all other models.

Conclusion Of the 3 SVM kernels used the Radial Kernel was the most accurate, being the only model that had a higher accuracy than the base logistical regression model. All of the SVM kernels were tuned to find the best cost and gamma values for the model, but even with changes in those parameters the accuracy did not majorly increase. Since the data set contains only quantitative parameters with a binary classifier being if a given star is a Pulsar or a Non-Pulsar star, I feel that Logistic regression is more appropriate to use, since it's accuracy is extremely similar to all the other models accruacy but it does not require tuning to find the optimal hyper parameters.