Pulsars_project

Predicting a pulsar star

By Steven Jones

12/08/2020

Introduction

Pulsars are a form of Neutron star which is created when a massive star runs out of fuel and collapses in on itself, crushing the protons and electrons into a neutron. Pulsars are rotating Neutron stars and are observed from earth as pulses of radiation lasting from milliseconds to seconds. Pulsars funnel jets of particles, often observed as light, out of their magnetic poles like a lighthouse we only observe the pulses when they face us.

Each pulsar produces a different emissions pattern and a detection is averaged over many rotations. Scientists can use Pulsar stars to search for gravitational waves, study extreme states of matter, search for planets outside of our solar system and measure cosmic distances. However, in practice almost all detections are caused by radio frequency interference (RFI) meaning it is difficult to differentiate legitimate candidates from false RFI.

This is a dataset containing 16,259 false identifications (RFI noise) and 1,639 real pulsars all checked by human annotators. The data was collected by the High Time Resolution Survey.

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

Dr Robert Lyon University of Manchester School of Physics and Astronomy Alan Turing Building Manchester M13 9PL United Kingdom robert.lyon '@' manchester.ac.uk

Project goal

For this project I will be looking at several machine learning algorithms and evaluating their potential for identifying legitimate Pulsars using the HTRU2 dataset. Success will be measured by a combination of factors including accuracy, sensitivity, which is the ability of our model to correctly identify Pulsar stars (true positive) and specificity, the ability to detect a true negative.

These additional measurement are important as we will come to learn that, whilst we can do a very good job of predicting accuracy, a perfect system is not possible and so researchers would have to prioritise not missing Pulsars from data (high sensitivity) with time spent manually verifying observations (high specificity).

Key steps performed

The report initially looks at the data and its distribution to get any insight as to which machine learning methods would work best. This is a classification project and, as such, I have first looked at 2 common classification models, KNN and random forest. Both models will be optimised as well.

Once we have a good baseline, I will apply more models to the problem simultaneously to see if we can improve results by taking the majority decision on results.

Analysis

A Quick look at the data shows it has 9 columns, 17,898 rows and 1,639 of the observations are actual Pulsars.

```
# No. of columns
ncol(dl)
```

```
## [1] 9
```

```
#No. of rows
nrow(dl)
```

```
## [1] 17898
```

```
#No. of pulsars
sum(dl$tClass)
```

```
## [1] 1639
```

We can also check out the Mean, Median, quartiles, min and max measurements of each column:

```
##      MeanIP             SdIP         ExcKurtIP           SkewIP
##  Min.   :  5.812   Min.   :24.77   Min.   :-1.8760   Min.   :-1.7919
##  1st Qu.:100.930   1st Qu.:42.38   1st Qu.: 0.0271   1st Qu.:-0.1886
##  Median :115.078   Median :46.95   Median : 0.2232   Median : 0.1987
##  Mean   :111.080   Mean   :46.55   Mean   : 0.4779   Mean   : 1.7703
##  3rd Qu.:127.086   3rd Qu.:51.02   3rd Qu.: 0.4733   3rd Qu.: 0.9278
##  Max.   :192.617   Max.   :98.78   Max.   : 8.0695   Max.   :68.1016
##    MeanCurve           SdCurve        ExcKurtCurve      SkewCurve
##  Min.   :  0.2132   Min.   :  7.37   Min.   :-3.139   Min.   :  -1.977
##  1st Qu.:  1.9231   1st Qu.: 14.44   1st Qu.: 5.782   1st Qu.:  34.961
##  Median :  2.8018   Median : 18.46   Median : 8.434   Median :  83.065
##  Mean   : 12.6144   Mean   : 26.33   Mean   : 8.304   Mean   : 104.858
##  3rd Qu.:  5.4643   3rd Qu.: 28.43   3rd Qu.:10.703   3rd Qu.: 139.309
##  Max.   :223.3921   Max.   :110.64   Max.   :34.540   Max.   :1191.001
##      tClass
##  Min.   :0.00000
##  1st Qu.:0.00000
##  Median :0.00000
##  Mean   :0.09157
##  3rd Qu.:0.00000
##  Max.   :1.00000
```

And we can do the same for just the false Pulsar signals:

```
##      MeanIP             SdIP         ExcKurtIP           SkewIP
##  Min.   : 17.21   Min.   :28.70   Min.   :-1.87601   Min.   :-1.7919
##  1st Qu.:105.25   1st Qu.:43.38   1st Qu.: 0.00865   1st Qu.:-0.2220
##  Median :117.26   Median :47.49   Median : 0.18666   Median : 0.1241
##  Mean   :116.56   Mean   :47.34   Mean   : 0.21044   Mean   : 0.3808
```

```
##  3rd Qu.:128.29   3rd Qu.:51.32   3rd Qu.: 0.39289   3rd Qu.: 0.6634
##  Max.   :192.62   Max.   :98.78   Max.   : 4.78579   Max.   :24.8724
##    MeanCurve          SdCurve        ExcKurtCurve      SkewCurve
##  Min.   :  0.2132   Min.   :  7.37   Min.   :-3.139   Min.   :  -1.977
##  1st Qu.:  1.8570   1st Qu.: 14.14   1st Qu.: 6.615   1st Qu.:  47.580
##  Median :  2.6355   Median : 17.62   Median : 8.760   Median :  90.675
##  Mean   :  8.8633   Mean   : 23.29   Mean   : 8.863   Mean   : 113.620
##  3rd Qu.:  4.2270   3rd Qu.: 24.52   3rd Qu.:10.935   3rd Qu.: 145.827
##  Max.   :223.3921   Max.   :110.64   Max.   :34.540   Max.   :1191.001
##      tClass
##  Min.   :0
##  1st Qu.:0
##  Median :0
##  Mean   :0
##  3rd Qu.:0
##  Max.   :0
```

And again for the true Pulsar signals:

```
##      MeanIP            SdIP         ExcKurtIP           SkewIP
##  Min.   :  5.812   Min.   :24.77   Min.   :-0.09489   Min.   :-1.139
##  1st Qu.: 31.777   1st Qu.:32.28   1st Qu.: 1.55604   1st Qu.: 3.805
##  Median : 54.297   Median :37.34   Median : 2.96126   Median :11.610
##  Mean   : 56.691   Mean   :38.71   Mean   : 3.13065   Mean   :15.554
##  3rd Qu.: 79.277   3rd Qu.:43.76   3rd Qu.: 4.58693   3rd Qu.:24.882
##  Max.   :139.258   Max.   :83.80   Max.   : 8.06952   Max.   :68.102
##    MeanCurve          SdCurve         ExcKurtCurve       SkewCurve
##  Min.   :  0.4866   Min.   :  7.659   Min.   :-1.8623   Min.   :  -1.8747
##  1st Qu.: 12.7596   1st Qu.: 43.334   1st Qu.: 0.7565   1st Qu.:  -0.2283
##  Median : 33.4950   Median : 59.367   Median : 1.9183   Median :   2.5851
##  Mean   : 49.8260   Mean   : 56.469   Mean   : 2.7571   Mean   :  17.9317
##  3rd Qu.: 78.3136   3rd Qu.: 70.986   3rd Qu.: 3.7100   3rd Qu.:  13.4697
##  Max.   :199.5778   Max.   :109.655   Max.   :30.8839   Max.   :1017.3832
##      tClass
##  Min.   :1
##  1st Qu.:1
##  Median :1
##  Mean   :1
##  3rd Qu.:1
##  Max.   :1
```

The first steps are to get the data ready which will allow us to create models. A quick check shows us the data is complete:

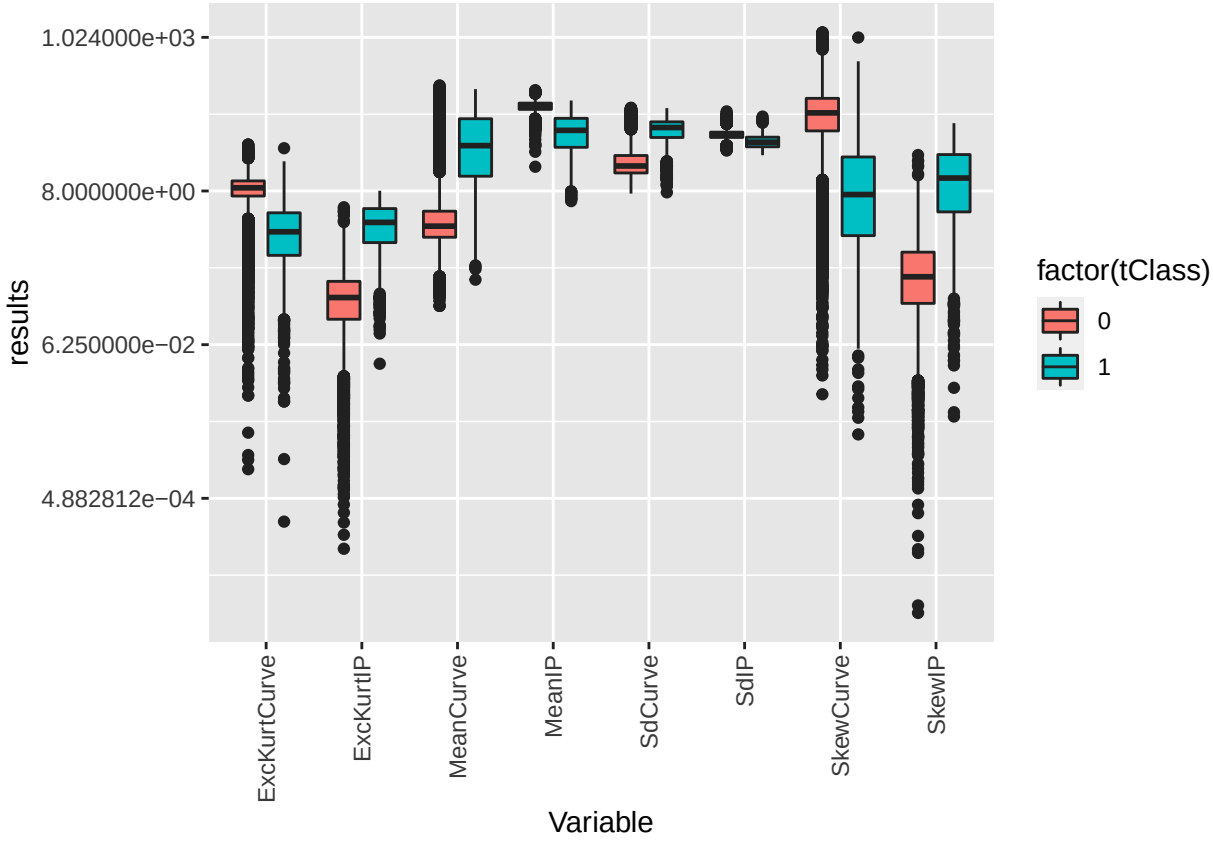
```
# Check for NA's
sum(is.na(dl))
```

```
## [1] 0
```

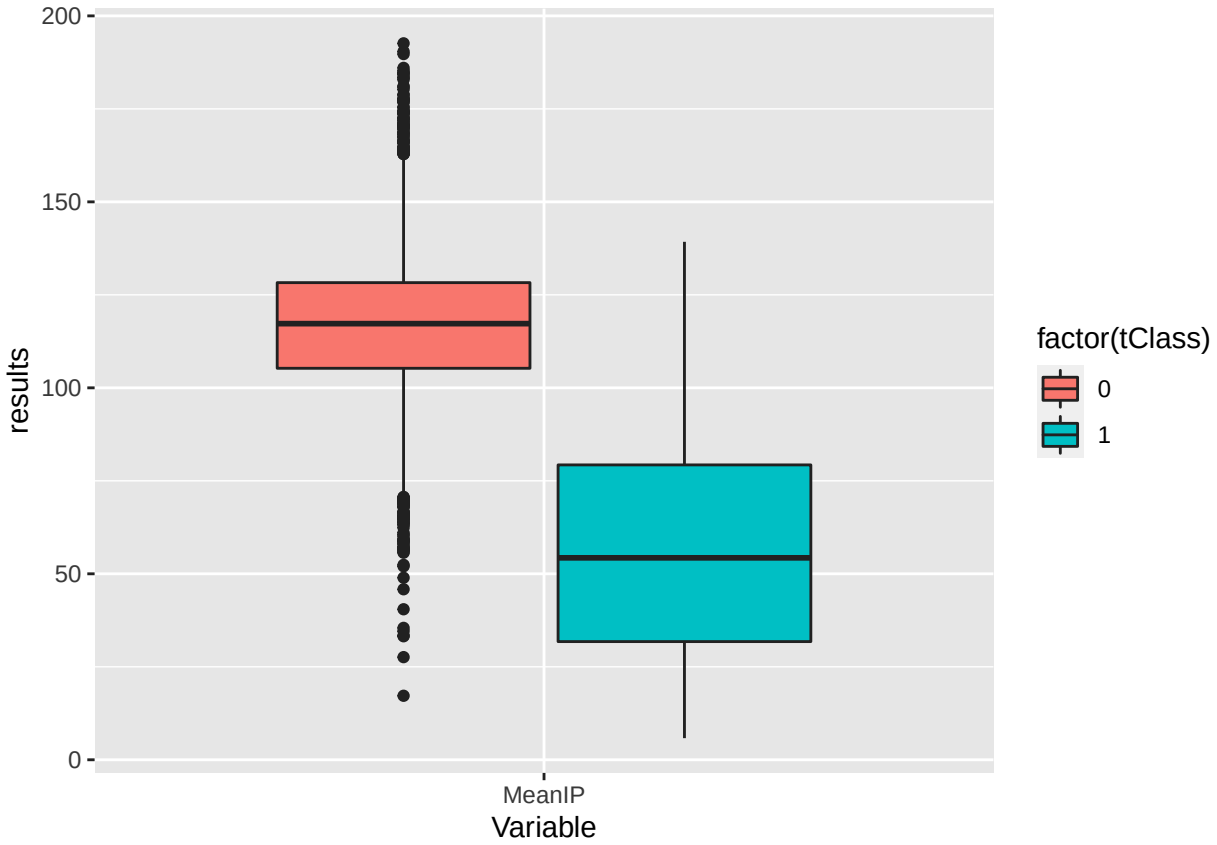
If we plot each observation and split the data by Pulsar or not Pulsar we can see both the opportunity and the challenges, we are going to face. On one hand every measurement has overlapping observations so we can see why it has been difficult to classify results but on the other hand the range between the 1st and 3rd

quartile does not overlap for any of the observation types. This will be a good place to start when coming up with predictions and should prove a useful benchmark.

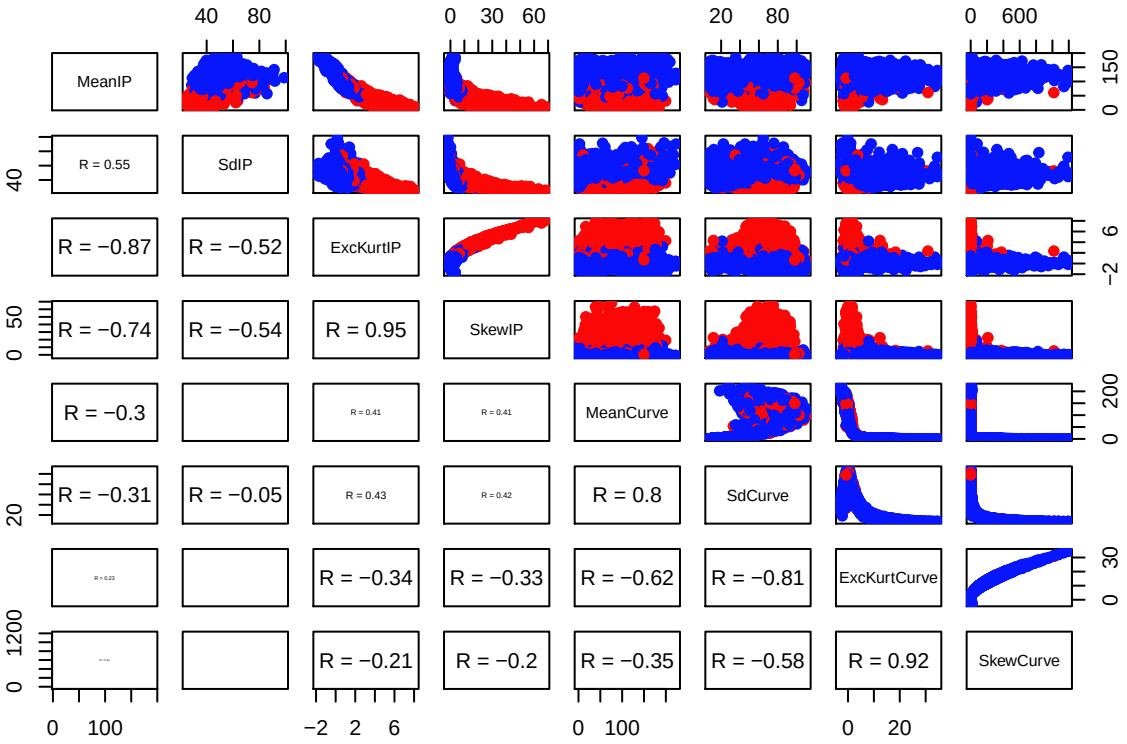
We also see a significant number of outliers; we can observe that the data is skewed for most of the measurements and that it is loosely grouped.



If we focus in on one example we can see more clearly how the observations overlap.

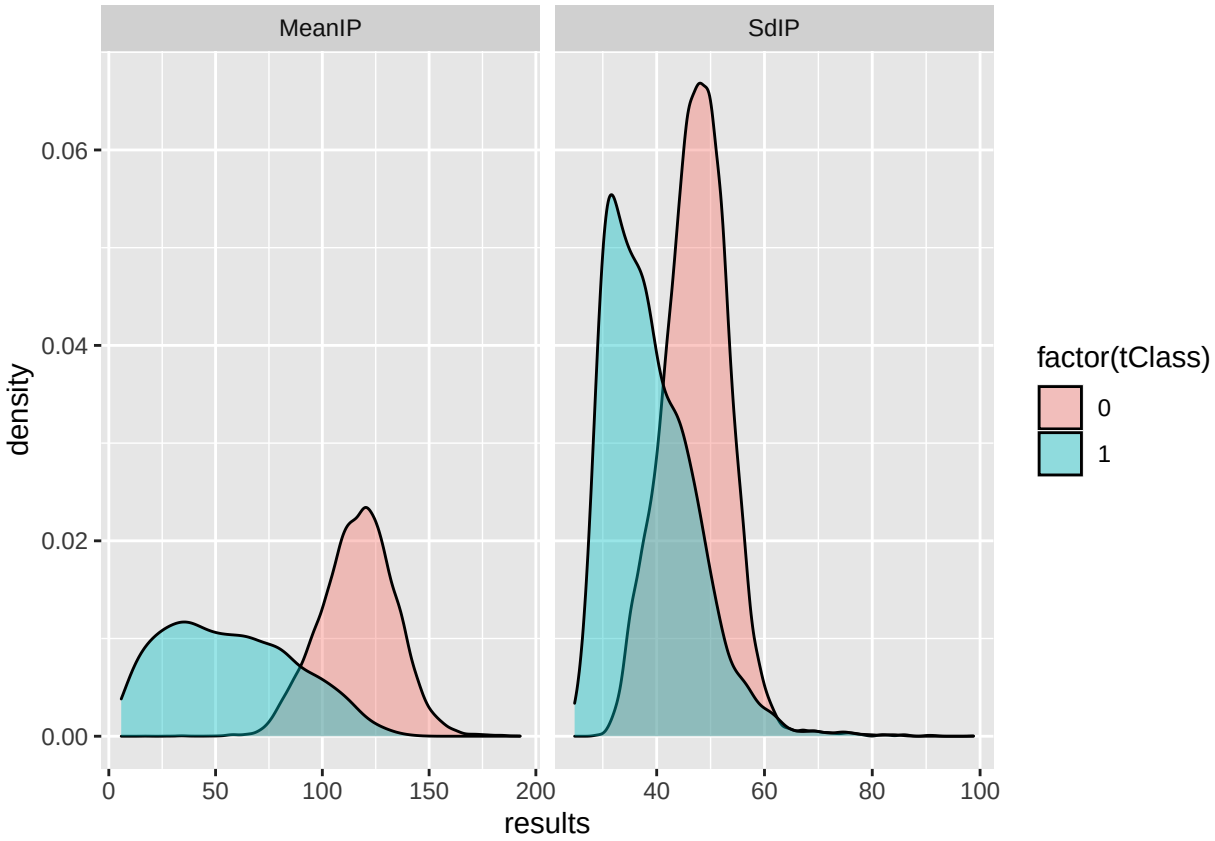


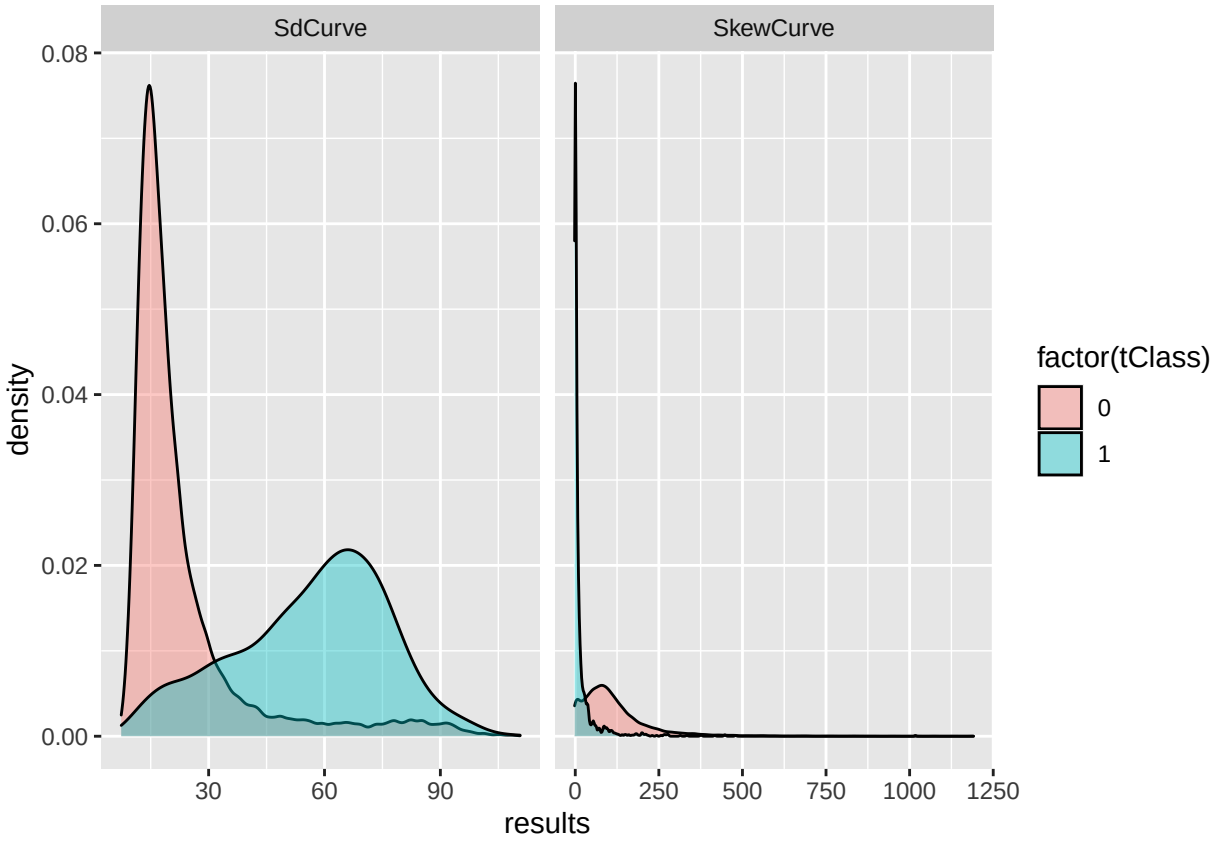
Next, I wanted to take a look at how each observation could be plotted against each other. We can see from the charts again that each observation includes some sort of overlap but that we should be able to achieve a relatively high accuracy for categorising Pulsars because some of the variables include observations which are mostly completely separate (Skew IP for example).

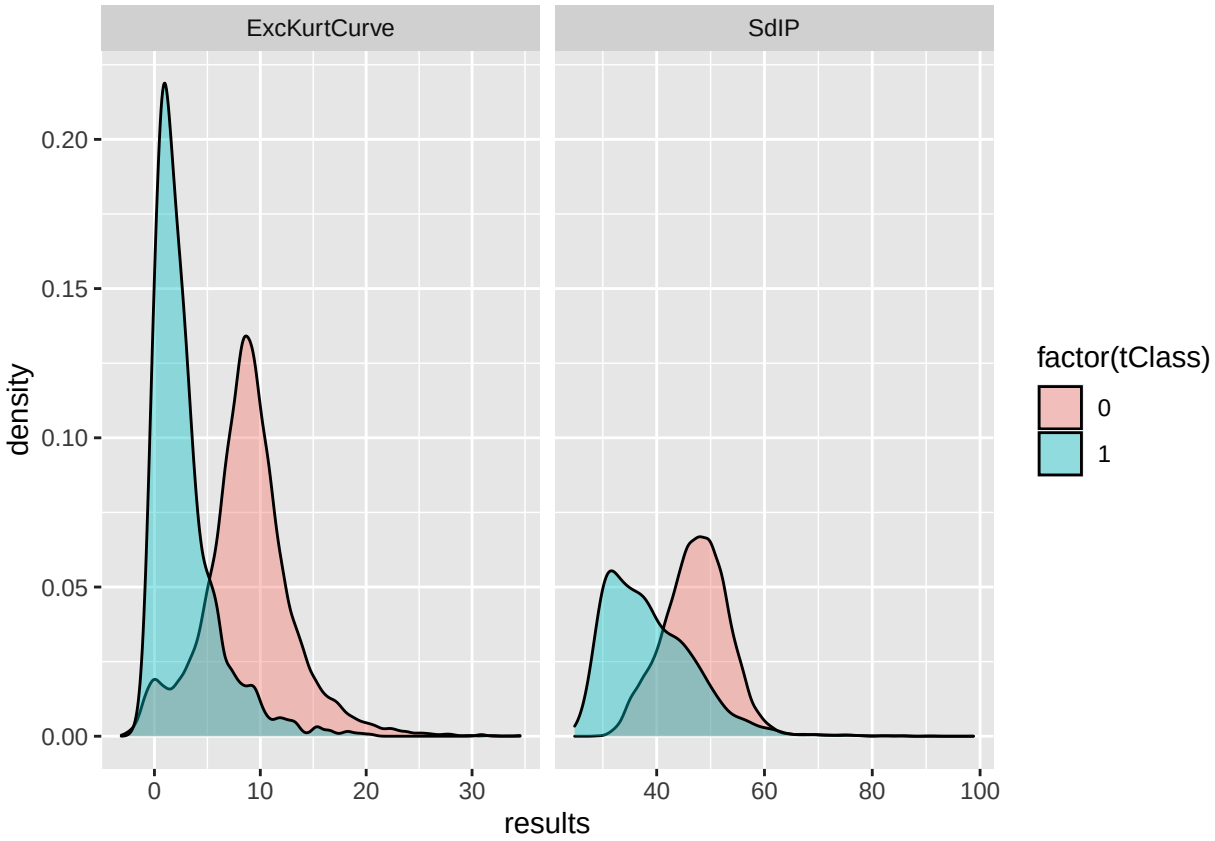


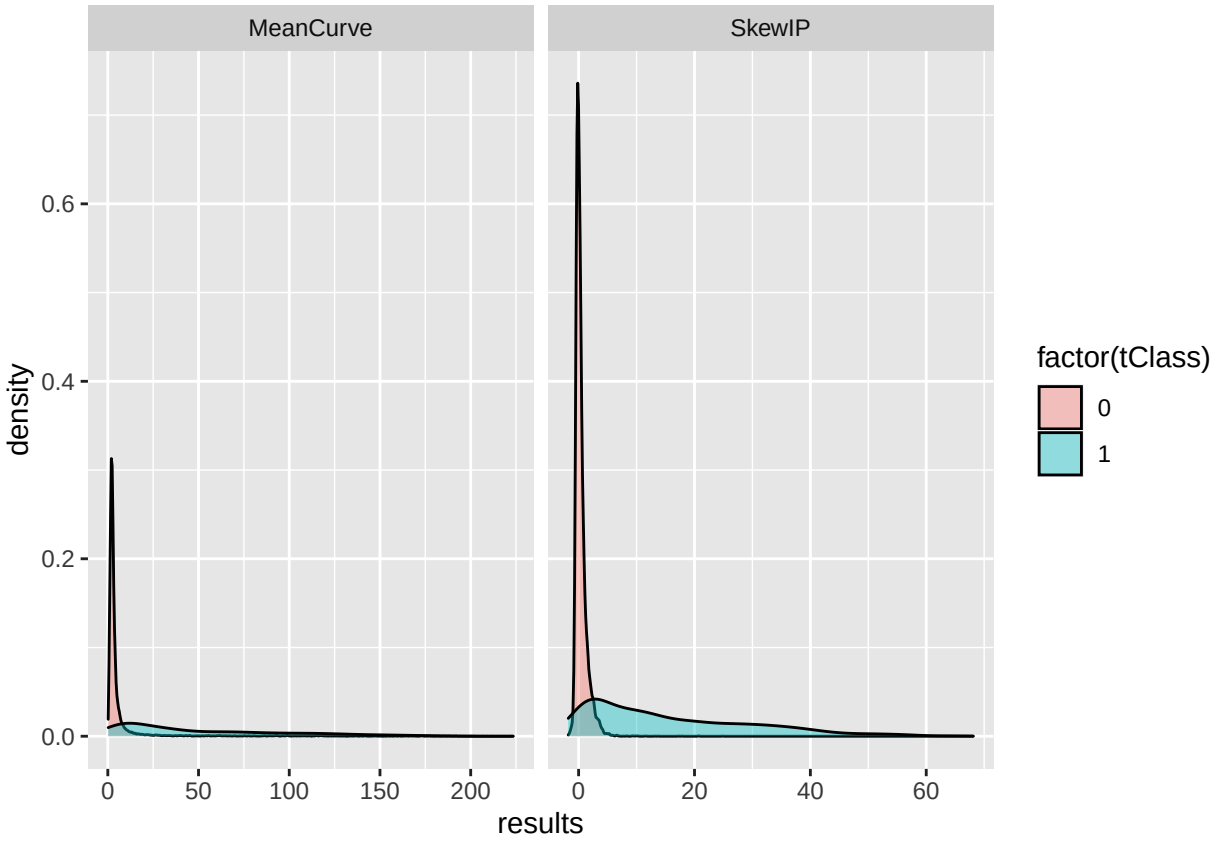
Distributions

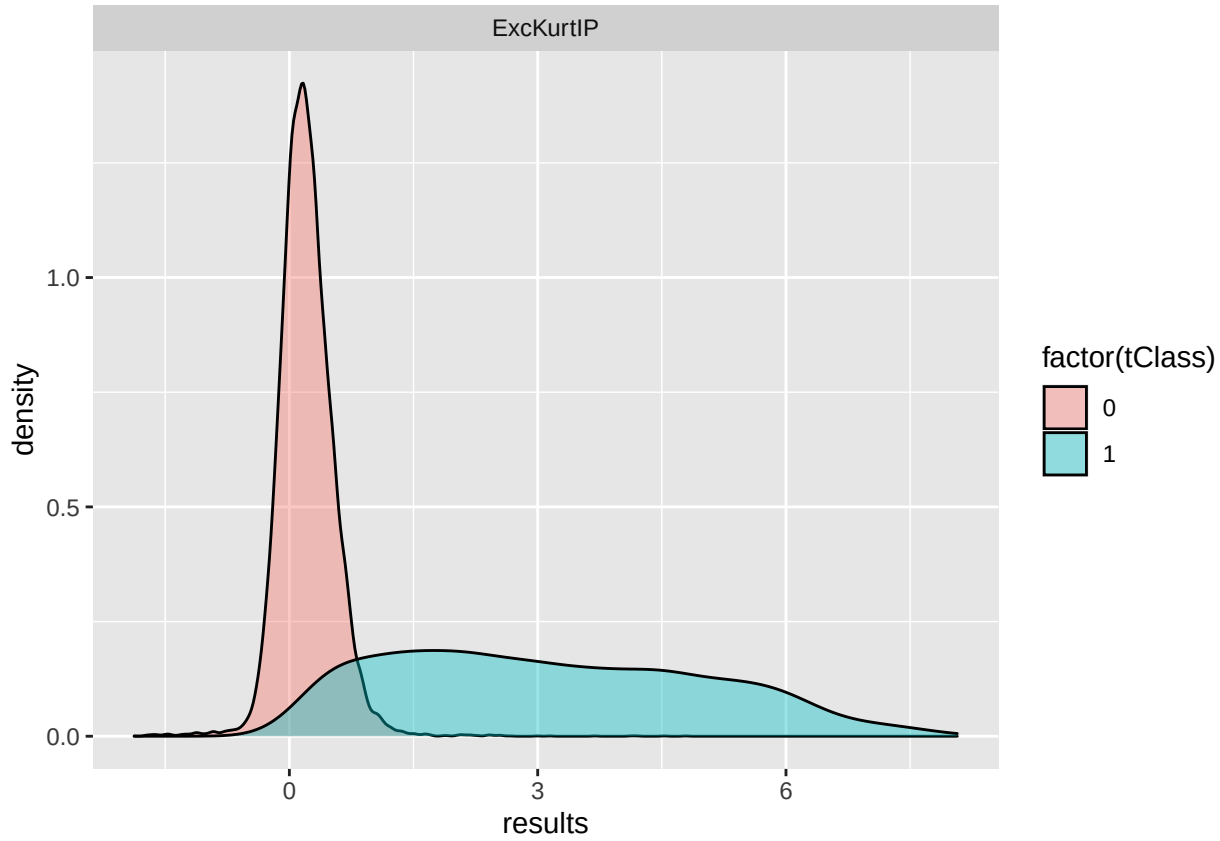
A look at the distributions also highlights that there are large areas where a significant part of the Pulsar distribution is unique. This indicates that we should be able to get a good degree of accuracy simply from predicting Pulsars should one of the measurements be above or below this cutt-off.


MeanIP
SdIP
density
0.06
0.04
0.02
0.00
0
50
100
150
200
40
60
80
100
results
factor(tClass)
0
1


SdCurve
SkewCurve
0.08
0.06
0.04
0.02
0.00
density
30
60
90
0
250
500
750
1000
1250
results
factor(tClass)
0
1


ExcKurtCurve
SdIP
density
0.20
0.15
0.10
0.05
0.00
0
10
20
30
40
60
80
100
results
factor(tClass)
0
1


MeanCurve
SkewIP
density
0.6
0.4
0.2
0.0
0
50
100
150
200
0
20
40
60
results
factor(tClass)
0
1



We can see that some of the distributions are normally distributed whilst the Pulsar observations are not (ExcKurtIP for example).

Modelling approach

From the data exploration I learned that there are large areas where there is no crossover of the data between the Pulsars and non-pulsars and so for the first approach, and to get a baseline, I have simply predicted Pulsar if the observation falls within the quartiles with no crossover.

From here I have used two common classification models, KNN and Random forest and will optimise both of these before creating an ensemble to see if it's possible to improve the results further.

Results

Classification based on 1st, 2nd and 3rd quartiles

We noted earlier in our exploratory data analysis that although each observation had overlapping results this only represented a small number of observations. As a result, for our baseline I'm going to classify each observation for each category into:

+1 if the observation falls within 1st to 3rd quartile of all Pulsar observations 0 if the observation sits in the overlapping region

Once we have these, I will take a total for each row and use the majority vote to decide if we predict Pulsar or not.

First, we set up the data by changing the Pulsars into factors with two levels. Lets check this has worked:

```
## [1] "factor"
```

```
## [1] "factor"
```

```
## [1] "0" "1"
```

```
## [1] "0" "1"
```

Now we can implement our classification based on 1st and 3rd quartiles.

```
# Classififcation by quartiles
Pulsar_quartiles <- dl %>%
  filter(tClass == 1) %>%
  select(all_of(measure))

# Get quantiles for each measurement
Pul_qunts <- apply(Pulsar_quartiles, 2,quantile)


# Dividing into groups depending on which quartiles we're selecting. Group A 3rd quartile of Pulsars is
Grpa <- c("MeanIP","SkewCurve","ExcKurtCurve","SdIP")
Grpb <- c("SdCurve","MeanCurve","SkewIP","ExcKurtIP")

# 3rd quartils for Group A Pulsar distribution
Pul_max_a <- Pul_qunts[4,Grpa]

# 1st quartile for Group B Pulsar distribution
Pul_min_b <- Pul_qunts[2,Grpb]

# Predict Pulsar if observation between 1st - 3rd quartile for Group A observations and 2nd - 4th Group
predict_Pulsar <- train_set %>%
  mutate(a = ifelse(MeanIP<Pul_max_a,1,0),
         b = ifelse(SkewCurve<Pul_max_a,1,0),
         c = ifelse(ExcKurtCurve<Pul_max_a,1,0),
         d = ifelse(SdCurve>Pul_min_b,1,0),
         e = ifelse(MeanCurve>Pul_min_b,1,0),
         f = ifelse(SkewIP>Pul_min_b,1,0),
         g = ifelse(ExcKurtIP>Pul_min_b,1,0)) %>%
  select(tClass,a,b,c,d,e,f,g) %>%
  mutate(pred = ifelse(a+b+c+d+e+f+g>3.5,1,0))

# Calculate accuracy
mean(predict_Pulsar$pred == predict_Pulsar$tClass)
```

```
## [1] 0.8709337
```

For some of the observations we classified them as pulsars if they were within 1st - 3rd quartile and for the others it was the 2nd - 4th quartile which we chose.

We achieved an accuracy of 87%.

Let us look at the results in more detail:

```r
# Calculate confusion matrix
confusionMatrix(as.factor(predict_Pulsar$pred), as.factor(train_set$tClass), positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 12898   326
##          1  1753  1131
##
##                Accuracy : 0.8709
##                  95% CI : (0.8657, 0.8761)
##     No Information Rate : 0.9095
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.4557
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.77625
##             Specificity : 0.88035
##          Pos Pred Value : 0.39216
##          Neg Pred Value : 0.97535
##              Prevalence : 0.09045
##          Detection Rate : 0.07021
##    Detection Prevalence : 0.17904
##       Balanced Accuracy : 0.82830
##
##        'Positive' Class : 1
##
```

By further diving into the results we can see we've successfully identified 77% of the Pulsars using this method (Sensitivity) and correctly identified 88% of the Non Pulsars correctly (Specificity).

KNN

Now we need to check and see how much better our well-known machine learning models can do.

We can train a K-nearest neighbour model with the following code:

```r
train_knn <- train(tClass ~ ., method = "knn", data = train_set)
knn_pred <- predict(train_knn, test_set)
mean(knn_pred == test_set$tClass)
```

```
## [1] 0.972067
```

We get an accuracy of over 97% which is pretty good.

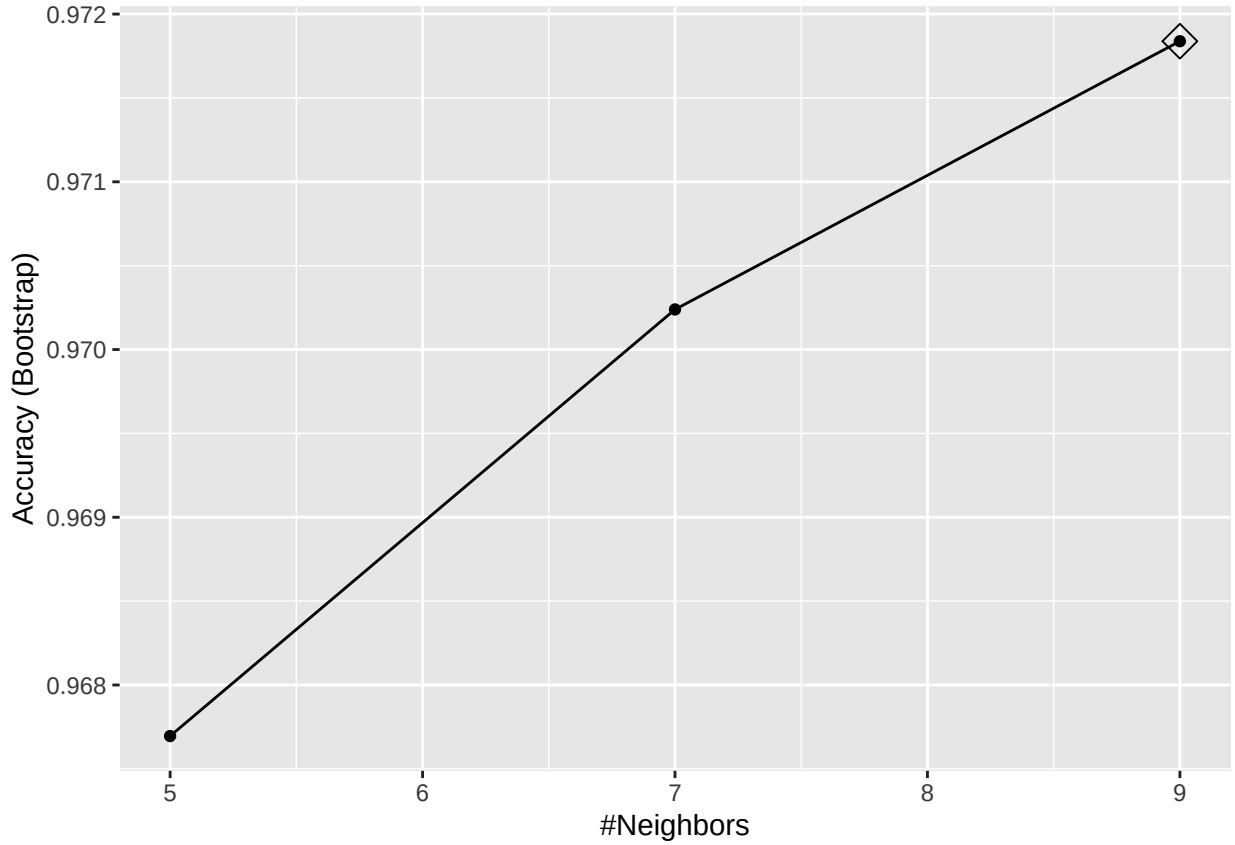
Next, let's find out more about KNN and see how we could tune the model to improve the results further:

```
modelLookup("knn")
```

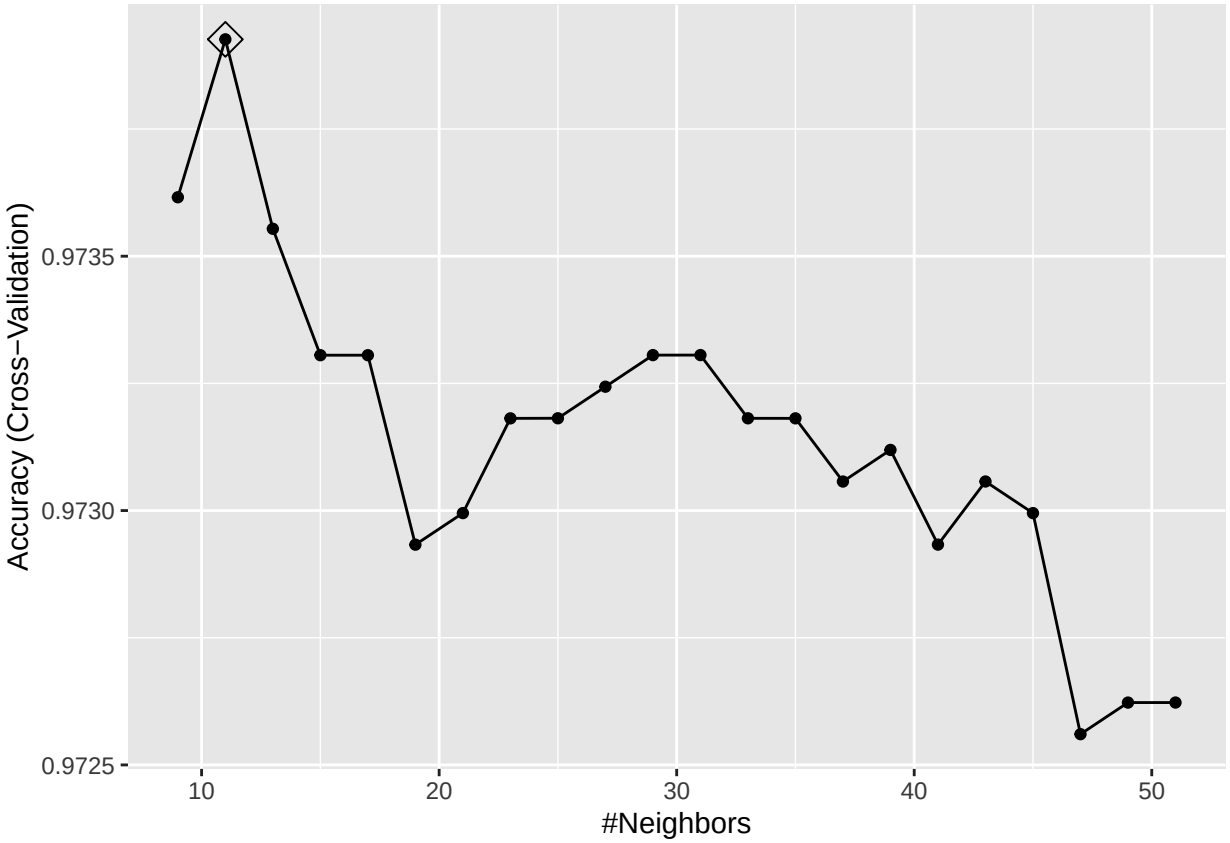
```
##   model parameter      label forReg forClass probModel
## 1   knn         k #Neighbors   TRUE     TRUE      TRUE
```

We can see that the number of nearest neighbours is the only tuning parameter. By default, KNN evaluates between 5 and 9 nearest neighbours. Let us plot these to see which was the best performing of our tuning parameter.

```
# PLot default Neighbours values
ggplot(train_knn, highlight = TRUE)
```



From the plot we can see the default values of KNN with 9 producing the best accuracy. Because 9 is our maximum of the default values we need to run the model again with numbers higher than 9 to check if 9 is the best accuracy. This time we will also use cross validation to improve our confidence levels.



We can see that 11 KNN is the best performing model increasing our accuracy significantly to over 0.9735. Let's run the model again with 11 nearest neighbours and evaluate the results:

```
# We run the model with 11 nearest neighbours
train_knn_cv <- train(tClass ~ ., method = "knn",
                      data = train_set,
                      tuneGrid = data.frame(k = 11),
                      trControl = control)

# Run predictions on test set
train_knn_cv_pred <- predict(train_knn_cv,test_set)
```

```
#Calculate accuracy, sensitivity, specificty etc
confusionMatrix(train_knn_cv_pred,test_set$tClass,positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1596   42
##          1   12  140
##
##                Accuracy : 0.9698
##                  95% CI : (0.9608, 0.9773)
##     No Information Rate : 0.8983
##     P-Value [Acc > NIR] : < 2.2e-16
```

```
##
##                   Kappa : 0.8218
##
##  Mcnemar's Test P-Value : 7.933e-05
##
##             Sensitivity : 0.76923
##             Specificity : 0.99254
##          Pos Pred Value : 0.92105
##          Neg Pred Value : 0.97436
##              Prevalence : 0.10168
##          Detection Rate : 0.07821
##    Detection Prevalence : 0.08492
##       Balanced Accuracy : 0.88088
##
##        'Positive' Class : 1
##
```

What is interesting is that although the accuracy has increased significantly on our baseline model, the sensitivity has decreased by around 1% indicating that it's worse performing at correctly predicting Pulsars.

Now that I have optimised the KNN algorithm I will try some other models and compare them. The next algorithm is 'Random forest' whereby our algorithm will create decision trees from many different samples of data and average the outcome out over however many 'forests' we choose to run.

Random Forest

We can train our model using the following code:

```
# Train Random Forest
Train_RF <- train(tClass ~ .,
                  method = "Rborist",
                  nTree = 5,
                  data = train_set)
```

By using the random forest we've managed to increase our accuracy further to 0.976.

Lets look at the results in more detail:

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1599   33
##          1    9  149
##
##                Accuracy : 0.9765
##                  95% CI : (0.9684, 0.983)
##     No Information Rate : 0.8983
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8636
##
##  Mcnemar's Test P-Value : 0.0003867
```

```
##
##             Sensitivity : 0.81868
##             Specificity : 0.99440
##          Pos Pred Value : 0.94304
##          Neg Pred Value : 0.97978
##              Prevalence : 0.10168
##          Detection Rate : 0.08324
##    Detection Prevalence : 0.08827
##       Balanced Accuracy : 0.90654
##
##        'Positive' Class : 1
##
```

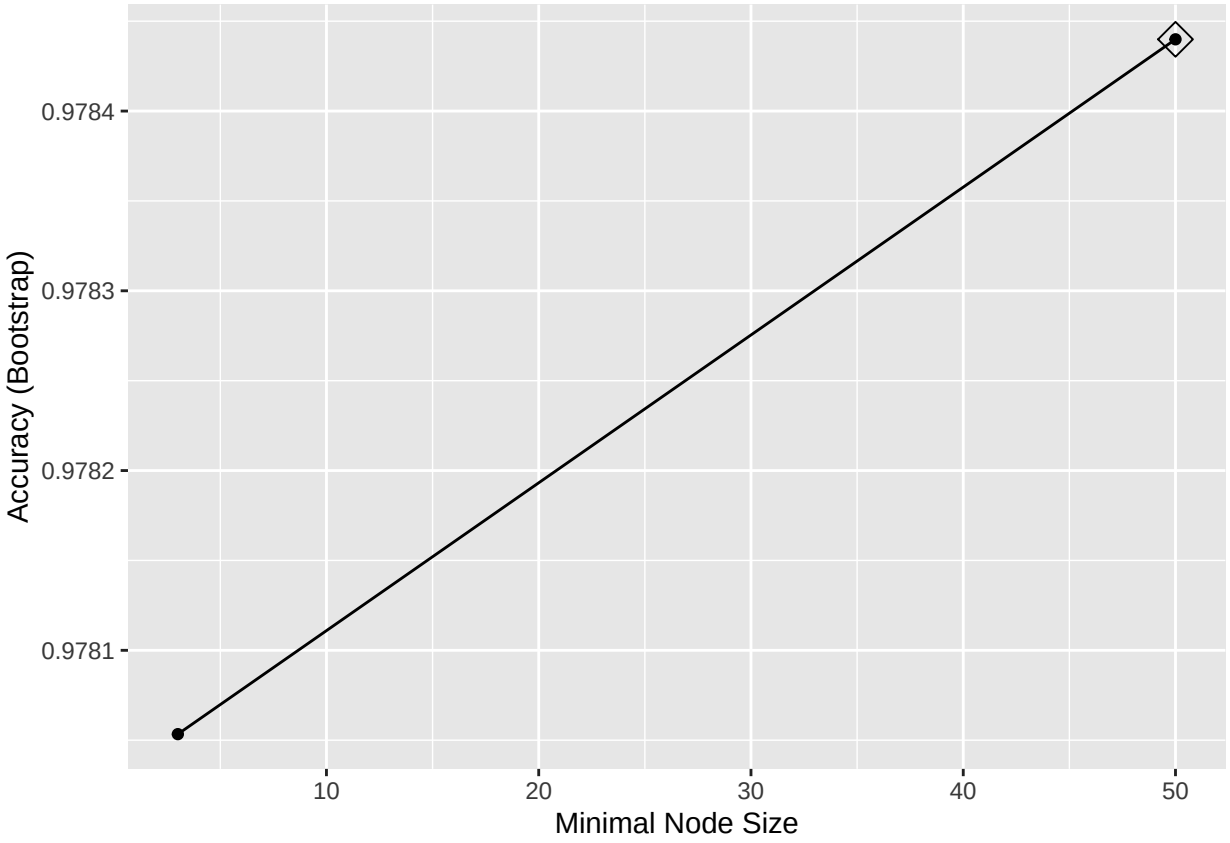
We can see the Random forest improved accuracy is again from high specificity. When it comes to identifying Pulsars, it has missed over 18% but still improves quite significantly on the other two models.

Let's see what tuning parameters are available:

```
modelLookup("Rborist")
```

```
##     model parameter                          label forReg forClass probModel
## 1 Rborist predFixed #Randomly Selected Predictors   TRUE     TRUE      TRUE
## 2 Rborist   minNode             Minimal Node Size   TRUE     TRUE      TRUE
```

Lets tune our model to see if we can improve our accuracy further:



The best accuracy returns at 50 nodes so let's train the Random forest again based on 50 node and look at the results:

```
set.seed(19, sample.kind = "Rounding")

# Train Random Forest
Train_RF_tun_top <- train(tClass ~ .,
                      method = "Rborist",
                      nTree = 5,
                      tuneGrid = data.frame(predFixed = 2, minNode = 50),
                      data = train_set)

# Apply model to test set
RF_Pred_tun <- predict(Train_RF_tun_top,test_set)

# Confusion matrix
confusionMatrix(RF_Pred_tun,as.factor(test_set$tClass),positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1601   34
##          1    7  148
##
##                Accuracy : 0.9771
##                  95% CI : (0.9691, 0.9835)
##     No Information Rate : 0.8983
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8658
##
##  Mcnemar's Test P-Value : 4.896e-05
##
##             Sensitivity : 0.81319
##             Specificity : 0.99565
##          Pos Pred Value : 0.95484
##          Neg Pred Value : 0.97920
##              Prevalence : 0.10168
##          Detection Rate : 0.08268
##    Detection Prevalence : 0.08659
##       Balanced Accuracy : 0.90442
##
##        'Positive' Class : 1
##
```

We have a light increase in accuracy to 97.7% but sensitivity has remained the same.

Finally let's see if an ensemble by majority vote can do any better.

Ensemble

An ensemble is where we will train several different types of machine learning models and predict Pulsar if more than half of the models predict Pulsar. It can be an effective way of minimising the limitations of

certain models and could improve results.

First we define the models we're going to use in the ensemble:

```
# Define models
models <- c("svmLinear", "gamLoess", "qda", "knn","Rborist")
```

After training our model we can report the accuracy for each model and the average across them.

```
## [1] "svmLinear"
## [1] "gamLoess"

## Loading required package: gam

## Loading required package: splines

## Loading required package: foreach

##
## Attaching package: 'foreach'

## The following objects are masked from 'package:purrr':
##
##     accumulate, when

## Loaded gam 1.16.1

## [1] "qda"
## [1] "knn"
## [1] "Rborist"

## [1] 1790    5
```

Accuracy for each model:

```
## [1] 0.9743017 0.9737430 0.9737430 0.9675978 0.9759777
```

Average accuracy across the ensemble:

```
mean(accuracy)
```

```
## [1] 0.9730726
```

We get an average accuracy of 97.3%.

We now want to choose a Pulsar or not by majority decision:

```
# building the ensemble prediction model based on majority decision
maj <- rowMeans(ensemble_pred == "1")
y_hat <- ifelse(maj > 0.5, "1", "0")
```

Let us check the results in more detail:

```r
# Confusion matrix on ensemble model
confusionMatrix(as.factor(y_hat),as.factor(test_set$tClass),positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1599   34
##          1    9  148
##
##                Accuracy : 0.976
##                  95% CI : (0.9678, 0.9826)
##     No Information Rate : 0.8983
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.86
##
##  Mcnemar's Test P-Value : 0.0002522
##
##             Sensitivity : 0.81319
##             Specificity : 0.99440
##          Pos Pred Value : 0.94268
##          Neg Pred Value : 0.97918
##              Prevalence : 0.10168
##          Detection Rate : 0.08268
##    Detection Prevalence : 0.08771
##       Balanced Accuracy : 0.90379
##
##        'Positive' Class : 1
##
```

We get a final accuracy of 97.6% from our ensemble which is an improvement on knn and a very small improvement on random forest. One thing to note is the sensitivity has not increased significantly and has fallen from what we achieved on Random Forest.

Conclusion

In this report we've taken a look at the relationships between measurements of several factors used in identifying Pulsars from false signals from radio frequency interference. We've managed to get our accuracy rate up from 86% up to 97.6% and our sensitivity up from 76% to 81.8%.

The major limitation of our models is they have proved very successful in identifying what is false readings from real Pulsars but less successful in identifying Pulsars correctly. That said we've established a number of different models which each have their strengths and weaknesses and could be used by researchers.

One other limitation I encountered was the time taken to run my ensemble models and in the end I had to reduce the train set down to 10% of its size to get this to run in a reasonable amount of time. With more data I would predict an improved ensemble result.

I would like to have done more work to see if we could come up with a model which had a higher sensitivity. I think for some researchers they would have sacrificed specificity, and spent time manually checking results, if it meant incorrectly identifying Pulsars as false signals.

The models we've established can be used to significantly improve detection of Pulsars which can speed up and help advance research into gravitational waves, study extreme states of matter, search for planets outside of our solar system and measure cosmic distances.