Data Science: Capstone Project - Predicting Pulsar Stars

Man Chun Hui

Abstract

In this **IDV project**, we used different machine learning algorithms to improve the prediction accuracy of **Pulsars**[1]. Exploration of the data[2] showed it to have prevalence, in favour of non pulsar stars, and that it would be hard to develop manual rules to accurately predict some of the **Pulsars**. The highest **accuracy** of **98.16%** was obtained using the **Decision Trees** algorithm while the highest **F1 Score** of **93%** was also obtained using the **Naive Bayes** algorithm.

1 Introduction

The data set, HTRU2[3], used in this **project** describes a sample of pulsar candidates collected during the High Time Resolution Universe Survey.

> "Pulsars are a rare type of Neutron star that produce radio emission detectable here on Earth. As pulsars rotate, their emission beam sweeps across the sky, and when this crosses our line of sight, produces a detectable pattern of broadband radio emission. As pulsars rotate rapidly, this pattern repeats periodically. Thus pulsar search involves looking for periodic radio signals with large radio telescopes.
>
> Each pulsar produces a slightly different emission pattern, which varies slightly with each rotation. Thus a potential signal detection known as a 'candidate', is averaged over many rotations of the pulsar, as determined by the length of an observation. In the absence of additional info, each candidate could potentially describe a real pulsar. However in practice almost all detections arecaused by radio frequency interference (RFI) and noise, making legitimate signals hard to find."
>
> By Dr Robert Lyon[4]

Light exploration of the HTRU2 data set, refer to Table 1, shows that there are 16,259 examples of non pulsar starts caused by RFI/noise, and 1,639 real pulsar examples[5] Additionally it is clear that legitimate pulsar examples are in the minority, and spurious/non pulsar examples are in the majority.

Table 1: HTRU2 dataset exploration

No. Of Rows	No. Of Columns	No. of Non Pulsars	No. of Pulsars
17898	9	16259	1639

Going forward in this **IDV project** we will be using Machine learning algorithms to automatically label pulsar candidates that facilitates rapid analysis as accurately as possible.

[1]Pulsars are rotating neutron stars observed to have pulses of radiation at very regular intervals that typically range from milliseconds to seconds - https://imagine.gsfc.nasa.gov/science/objects/neutron_stars1.html

[2]https://www.kaggle.com/pavanraj159/predicting-a-pulsar-star/download

[3]https://www.kaggle.com/pavanraj159/predicting-a-pulsar-star

[4]https://www.kaggle.com/pavanraj159/predicting-a-pulsar-star

[5]These examples have all been checked by human annotators.

2 Methods / Analysis

2.1 HTRU2 data

To make analysis of the dataset more manageable the original column names have been shortened, details in *Table 2* below.

Table 2: Renaming Columns

Original_Column_Names	Shortened_Column_Names
Mean.of.the.integrated.profile	mean_ip
Standard.deviation.of.the.integrated.profile	sd_ip
Excess.kurtosis.of.the.integrated.profile	exk_ip
Skewness.of.the.integrated.profile	skew_ip
Mean.of.the.DM.SNR.curve	mean_ds_c
Standard.deviation.of.the.DM.SNR.curve	sd_ds_c
Excess.kurtosis.of.the.DM.SNR.curve	exk_ds_c
Skewness.of.the.DM.SNR.curve	skew_ds_c
target_class	target

The HTRU2 data is already in tidy[6] format with the predictors in the first eight columns and the *target_class* label being the final entry. The *target_class* labels are 0 (negative) and 1 (positive) with positive meaning it is a pulsar star.

Table 3: HTRU2 dataset (First 10 Rows)

mean_ip	sd_ip	exk_ip	skew_ip	mean_ds_c	sd_ds_c	exk_ds_c	skew_ds_c	target
140.56250	55.68378	-0.2345714	-0.6996484	3.1998328	19.110426	7.975532	74.24222	0
102.50781	58.88243	0.4653182	-0.5150879	1.6772575	14.860146	10.576487	127.39358	0
103.01562	39.34165	0.3233284	1.0511644	3.1212375	21.744669	7.735822	63.17191	0
136.75000	57.17845	-0.0684146	-0.6362384	3.6429766	20.959280	6.896499	53.59366	0
88.72656	40.67223	0.6008661	1.1234917	1.1789298	11.468720	14.269573	252.56731	0
93.57031	46.69811	0.5319048	0.4167211	1.6362876	14.545074	10.621748	131.39400	0
119.48438	48.76506	0.0314602	-0.1121676	0.9991639	9.279612	19.206230	479.75657	0
130.38281	39.84406	-0.1583228	0.3895404	1.2207358	14.378941	13.539456	198.23646	0
107.25000	52.62708	0.4526880	0.1703474	2.3319398	14.486853	9.001004	107.97251	0
107.25781	39.49649	0.4658820	1.1628771	4.0794314	24.980418	7.397080	57.78474	0

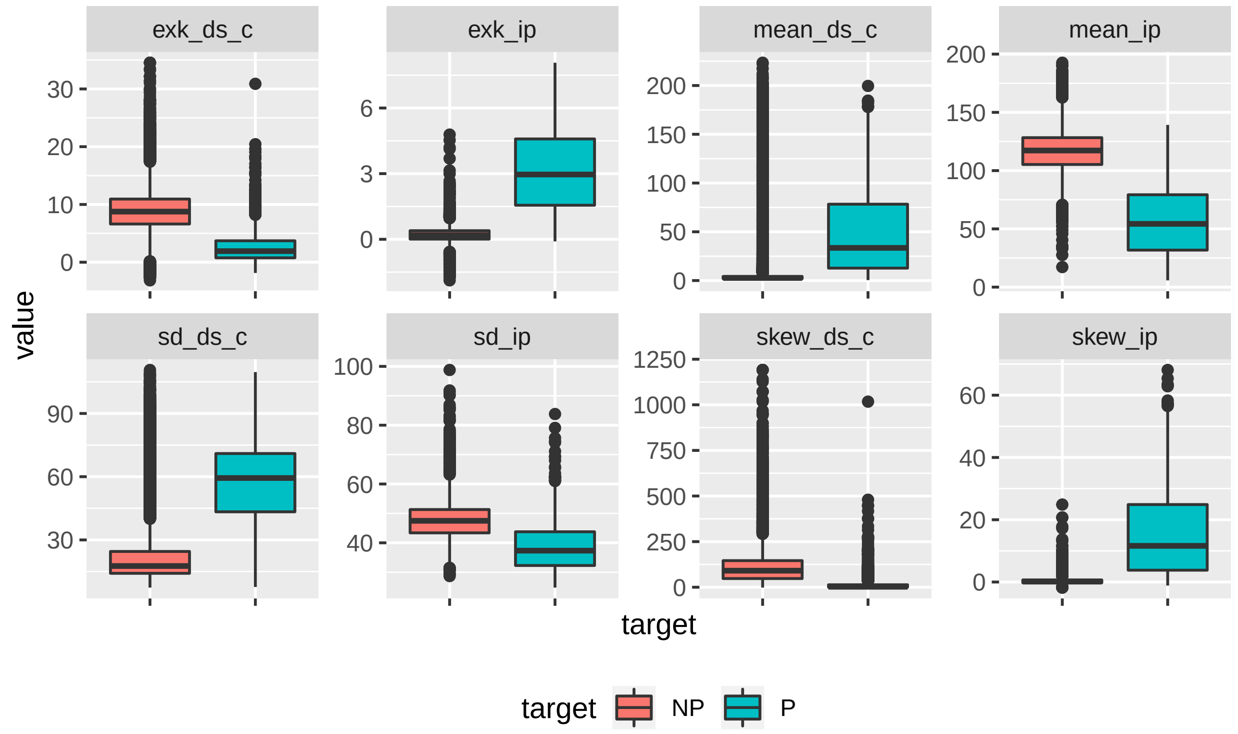
2.2 HTRU2 data exploration

The *target* label has now been converted to **NP** (negative) and **P** (positive) with positive meaning it is a pulsar star.

```r
#Pulsars "P" & Not Pulsars "NP" Analysis
MyData$target <- ifelse(MyData$target == 1, "P", "NP")
MyData %>% gather(predictors, value, -target) %>%
  ggplot(aes(target, value, fill = target)) +
  geom_boxplot() +
```

[6]https://cran.r-project.org/web/packages/tidyr/vignettes/tidy-data.html

```
  facet_wrap(~predictors, scales = "free", ncol = 4) +
  theme(axis.text.x = element_blank(), legend.position="bottom")
```



Review of the graphs above shows that while some pulsars have very distinguishable characteristics, which will allow easy identification, others do not. Furthermore its evident that a combination of predictors will give the highest prediction accuracy, it would be difficult to develop such rules manually and where machine learning excels at.

2.3 Create Test and Train set

Before progressing with experimenting with various different machine learning models we first normalise the data and then create seperate training and test sets.

```
# Scaling and Preparing Data
startype <- as.factor(MyData$target)
pstarpred <- as.matrix(select(MyData, -c(length(MyData))))
x_centered <- sweep(pstarpred, 2, colMeans(pstarpred))
x_scaled <- sweep(x_centered, 2, colSds(pstarpred), FUN = "/")

# Create Test and Train set
set.seed(1, sample.kind = "Rounding")
test_index <- createDataPartition(startype, times = 1, p = 0.2, list = FALSE)
test_x <- x_scaled [test_index,]
test_y <- startype[test_index]
train_x <- x_scaled [-test_index,]
train_y <- startype[-test_index]
```

2.4 Machine Learning - Introduction

We explored 5[7] different classification algorithms (Logistic Regression (glm), Naïve Bayes (naive_bayes), K-Nearest Neighbours (knn), Decision Tree (rpart), and Random Forest (rf) to assess which ones would provide the prediction best accuracy.

For each of the algorithm assessments we will use the same *trainControl* method of 5 x k-fold **Cross Validation** repeated 3 times, finally each assessment will show the code used, the best tuning parameter if available and the accuracy of the subject algorithm.

```
#training method
tr <- trainControl(method = "repeatedcv",
                   number = 5,
                   repeats = 3)
```

2.4.1 Machine Learning - Logistic regression algorithm

```
# Logistic regression algorithm
train_glm <- train(train_x,
                   train_y,
                   method = "glm",
                   tuneLength=10,
                   trControl = tr)
glm_preds <- predict(train_glm, test_x)
glm_cm <- confusionMatrix(glm_preds, test_y)
glm_cm$overall["Accuracy"]
```

```
##  Accuracy
## 0.9807263
```

2.4.2 Machine Learning - Naive bayes algorithm

```
# Naive bayes algorithm
set.seed(1, sample.kind = "Rounding")
train_nb <- train(train_x,
                  train_y,
                  method = "naive_bayes",
                  tuneLength=10,
                  trControl = tr)
nb_preds <- predict(train_nb, test_x)
train_nb$bestTune
```

```
##   laplace usekernel adjust
## 2       0      TRUE      1
```

```
nb_cm <- confusionMatrix(nb_preds, test_y)
nb_cm$overall["Accuracy"]
```

```
##  Accuracy
## 0.9689944
```

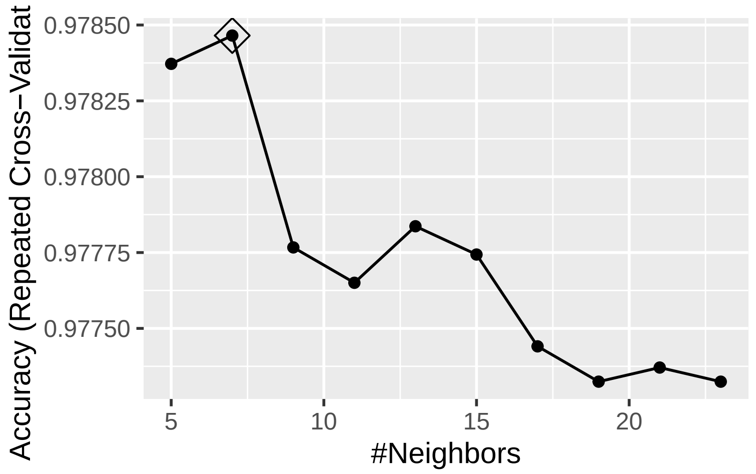
[7]https://analyticsindiamag.com/7-types-classification-algorithms/

2.4.3 Machine Learning - K-Nearest Neighbours algorithm

```r
# knn algorithm
set.seed(1, sample.kind = "Rounding")
train_knn <- train(train_x,
                   train_y,
                   method = "knn",
                   tuneLength=10,
                   trControl = tr)
knn_preds <- predict(train_knn, test_x)
train_knn$bestTune
```

```
##   k
## 2 7
```

```r
ggplot(train_knn, highlight = TRUE)
```



```r
knn_cm <- confusionMatrix(knn_preds, test_y)
knn_cm$overall["Accuracy"]
```

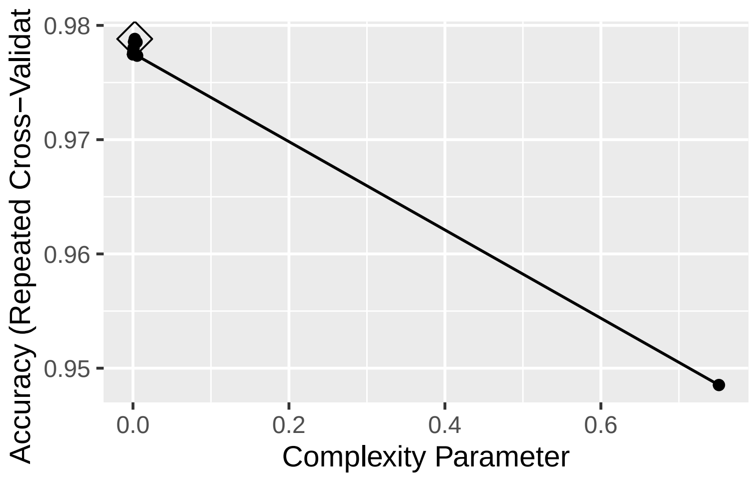
```
##  Accuracy
## 0.9807263
```

2.4.4 Machine Learning - Descision Trees algorithm

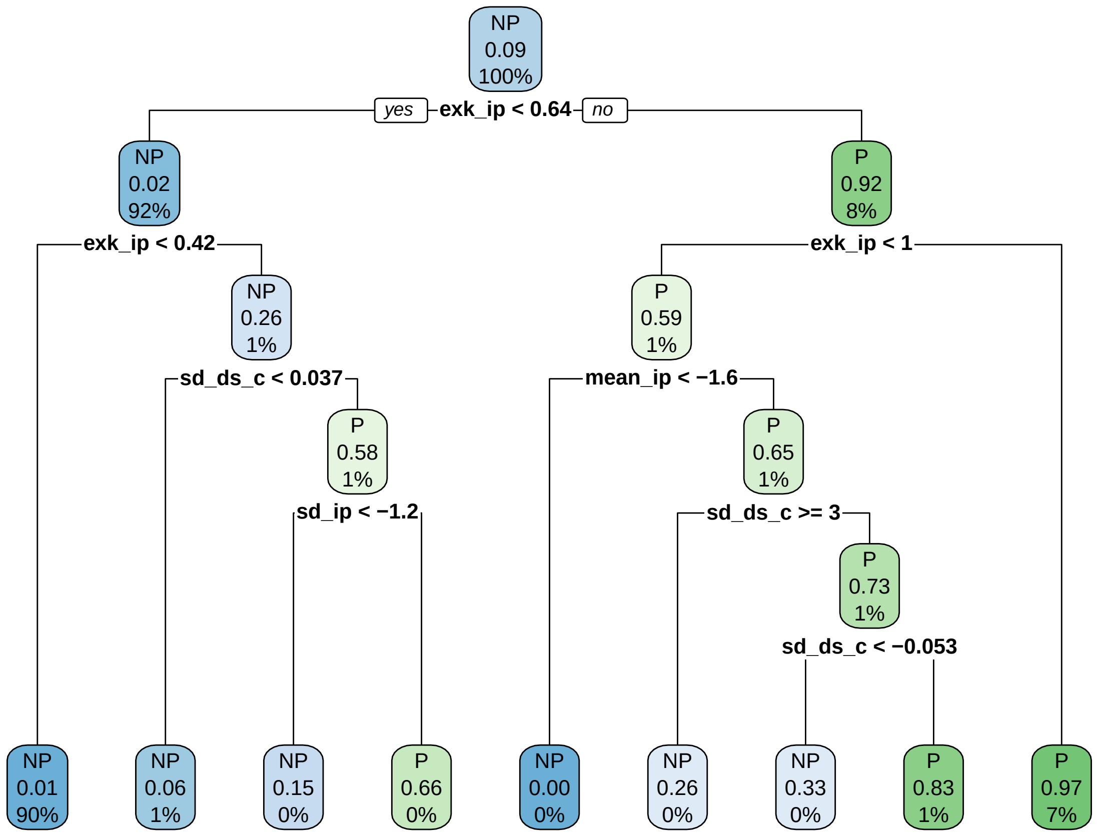
```r
# Rpart algorithm
set.seed(1, sample.kind = "Rounding")
train_rpart <- train(train_x,
                     train_y,
                     method = "rpart",
                     tuneLength=10,
                     trControl = tr)
rpart_preds <- predict(train_rpart, test_x)
train_rpart$bestTune
```

```
##           cp
## 7 0.00228833
```

```r
ggplot(train_rpart, highlight = TRUE)
```



```r
rpart.plot(train_rpart$finalModel)
```



The above plot is a good visual example of where machine learning tools can help develop complex rules in very short amounts of time.

```r
rpart_cm <- confusionMatrix(rpart_preds, test_y)
rpart_cm$overall["Accuracy"]
```

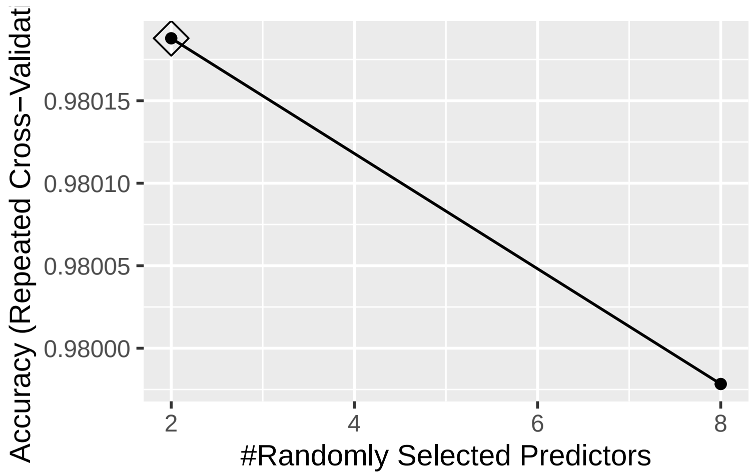
```
##  Accuracy 
## 0.9815642
```

2.4.5 Machine Learning - Random forest algorithm

```
# Random forest algorithm
set.seed(1, sample.kind = "Rounding")
train_rf <- train(train_x,
                  train_y,
                  method = "rf",
                  tuneLength=2,
                  trControl = tr,
                  ntree= 200,
                  importance = TRUE)
rf_preds <- predict(train_rf, test_x)
train_rf$bestTune
```

```
##   mtry
## 1    2
```

```
ggplot(train_rf, highlight = TRUE)
```



```
rf_cm <- confusionMatrix(rf_preds, test_y)
rf_cm$overall["Accuracy"]
```

```
##  Accuracy
## 0.9804469
```

3 Results

The algorithm that achieved the highest **accuracy** of **98.16%** was obtained using the **Decision Trees** algorithm. However it was clear that due to our uneven dataset, where legitimate pulsar examples are in the minority (9%) and spurious/non pulsar examples are in the majority (91%), all the algorithms produced results with very high sensitivity and not so high specificity.

As the goal of this project is to find the algorithm that was the **best** predictor for identification of **pulsar stars**, an algorithm with the highest specificity (**Naive Bayes** achieved 88.7%) would be best suited.

Finally the table below shows why the **F1 Score** is more useful than accuracy in cases where datasets are uneven.

Table 4: HTRU2 dataset - Machine Learning - Scores

Model	Accuracy	F1_Score	Sensitivity	Specificity
Logistic Regression	0.9807263	0.9100541	0.9950800	0.8384146
Naive Bayes	0.9689944	0.9300453	0.9772448	0.8871951
K-Nearest Neighbours	0.9807263	0.9166679	0.9938499	0.8506098
Descision Trees	0.9815642	0.9235699	0.9935424	0.8628049
Random forest	0.9804469	0.9181726	0.9932349	0.8536585

4 Conclusion

In this **IDV project**, we used different machine learning algorithms to improve the prediction accuracy of **Pulsars**[8]. Exploration of the data[9] showed it to have prevalence, in favour of non pulsar stars, and that it would be hard to develop manual rules to accurately predict some of the **Pulsars**. The highest **accuracy** of **98.16%** was obtained using the **Decision Trees** algorithm while the highest **F1 Score** of **93%** was also obtained using the **Naive Bayes** algorithm.

Finally a potential limitation in this study is that we only looked at a small number of the available classification algorithms, therefore further improvement in future work to improve the prediction accuracy would be to look at incorporating other classification algorithms into the study as well as additional tuning of the machine learning parameters for algorithms with tuning parameters.

[8]Pulsars are rotating neutron stars observed to have pulses of radiation at very regular intervals that typically range from milliseconds to seconds - https://imagine.gsfc.nasa.gov/science/objects/neutron_stars1.html

[9]https://www.kaggle.com/pavanraj159/predicting-a-pulsar-star/download