A suitable classification model for Pulsar detection

By
Maya Dere (3675042) & Hongli Li (3697150)

RMIT University

COSC 2670 Practical Data Science- Assignment 2

S3697150@student.rmit.edu.au
maya.dere@gmail.com

13 May 2018

Table of contents

1. Abstract

The aim of this report was to find a suitable classification model to select real pulsars from a huge number of candidates with increased accuracy and efficiency. Three classification models (K Nearest Neighbours, Decision Tree and Random Forest) were adopted, trained by 60% of the 17898 candidate samples. Overall, the results indicate that random forest model gives the highest accuracy score as well as relatively low misclassification error rate. The report concludes that Random Forest is the best performing model among the three. It is recommended that not to use this model to predict new set of candidates as the highly imbalanced distribution of the Pulsars and non-pulsars in nature.

2. Introduction

Pulsars are highly sought after by researchers because they are an important source of clues to the investigation and discovery of the universe and human's everyday life. Its application extends to the fields of space-time probing, global positioning system and gravitation wave. (Lyon, 2016)[1] A Pulsar is a type of star who emits radiation signals frequently like pulses. The reality is that among the huge number of signals captured, there is only about 10% can be detected as real pulsars, whereas the rest of the non-pulsar signals could be produced by ground interference or magnetic waves associated with human activities. (XU Yu-yun, 2017)[2] This makes the pulsar detection so hard. In the past decades of years, researchers have used a range of techniques to distinguish real pulsars and non-pulsars, the accuracy of the results and productivity is not ideal but is getting better. Since the launch of SKA telescope in 2015, super mass volume of pulsar candidates are being captured and therefore an on-time analysis for this big volume of data is needed.[1] Given this, the practical data science techniques are the best time-efficient and accurate-wise tool to tackle this

problem, the classification model speeds up the detection of new pulsars and the accuracy of the detection is increased significantly. This report hypotheses that there is a best classification model that can greatly increase the accuracy and efficiency of the process of distinguishing pulsars and non-pulsars. Three classification models will be used, namely the K Nearest Neighbours, Decision Tree and Random Forest. Feature selection and data exploration will also be covered in detail to display insights from the pulsar-candidate dataset.

3. Methodology

3.1 Data Retrieval

The dataset 'HTRU2' is provided by UCI ML Repository (Lyon D. R., 2017) [3] where 17898 pulsar candidates were collected with only 1639 (approx. 9.1%) are real pulsars. The highly imbalanced data class is due to the nature of the pulsar star distribution (see graph-9). This intrinsic imbalance impacts the performance of data modelling which will give poor accuracy on the minority class. (Chawla, 2004)[4]. Hence this report addressed this issue in the analysis by using K-Folds Cross validation to reduce this impact.

Each candidate is described by 8 numeric features, and a single class variable (where 0 = non-pulsar, 1= pulsar). The first four are statistics (Mean, SD, Excess Kurtosis, and Skewness) of the integrated pulse profile (ip). The remaining four are the same statistics of the Dispersion Measurement Curve (DM-SNR curve).[3] In the following sections regarding data exploration the information of the attributes will be more clearly explained.

3.2 Data Exploration

In this section, firstly the distribution of each single variable is visualized. The eight numeric features are plotted in density graphs and the discrete class variable is plotted in a bar chart. Secondly, the relationship between each feature against the class variable is illustrated in scatter plots, whenever the scatters are not 'helpful' to distinguish pulsars and non-pulsars, boxplot is used instead. Thirdly, the scatter plot of the two 'most helpful' features and of two statistics of the ip (integrated pulse profile) are illustrated. The visualisations are done in iPython, methods provided by Dr. Yongli Ren & Ahmed Mourad (2018).[5] These visualisations are in the Results section and the insights are interpreted in the Discussion section in the sub-heading of data exploration.

3.3 Data Modelling and evaluation

Split data into Train/Test set

Firstly formulate the classification problem by splitting the dataset into 40% of test set and 60% of training set, feature data is assigned to X and target variable is assigned to y, now the whole dataset has been split into four portions namely X_trian, X_test, y_train and y_test. This split will be applied to all three classification models below.[5]

K-Folds Cross validation

K-Folds Cross validation [5] is used to randomise data, in order to reduce the impact of the intrinsic imbalance of the pulsar data. The K-Folds algorithm divide the dataset into k parts, in this analysis the data was divided into five folds. So each time when a parameter is added or its value is changed, it can be ensured that random samples were selected without any bias. This method will apply to all three classification models below.

(3.3a) K Nearest Neighbour (KNN) Modelling

Import the KNN classifier, fit the X_train and y_train data to the KNN model and predict the classification for X_test data, the obtained predicted classes then are compared to the y_test (the existing classification status), construct the confusion matrix and classification report based on the predicted classes against the y_test classes. (Ren, 2018)[6]

From the confusion matrix, the classification error rate is calculated by dividing sum of non-diagonal entries (sum of misclassifications) by total sum of entries (sum of all samples) in the matrix.

From the classification report, the precision of classifying pulsars (1), non-pulsars (0) and their average can be obtained. Similarly, the respective recall values and F1-Scores can be obtained.

Feature selection – Hill Climb Technique

In this research, the Hill-Climb algorithm [5] has been used to select features. The aim is to pick up the features that are most strongly related to the class variable and eliminate any unimportant features. The algorithm has been executed eight times and each time the regarded important features will be selected in the output. The results are recorded and the frequency of each feature being selected is counted. See

figure_F below in the Results section, the level of importance of each feature are visualised. The final choice of variable used for proceeding analysis will be discussed in the Discussion section regarding Feature Selection.

Tuning Parameters

Start with choosing the optimal k value that gives the highest precision score or the lowest classification error rate, in this case the lowest misclassification error rate is plotted against all k values, this is visualised as figure_K in the Results section.

Secondly, based on the optimal k value, test each possible values for each parameter of the KNN model and select the value that gives the best accuracy. [5]

The 5-Folds cross validation has been applied to reduce the impact of the imbalanced data. To improve the accuracy of the model, tune the parameter 'weight', in this case when the weight is 'uniform', the accuracy is higher than when weight equals 'distance'. So any following parameters will be tested based on the Uniform weight. In turn, tune the parameter p, algorithm and leaf size and compare the precision score. Table_K1 shows that given the best k value, select the best value for parameter 'weight', and then given the best k value and the weight value, select the best value for parameter p.

In terms of the evaluation, the confusion matrix, classification error rate, precision, Recall and F1-score are summarised in table_K2, comparison is made between the model with default (no parameters) and the model with tuned parameters.

(3.3b) Decision Tree (DT) Modelling

Similarly as the procedures in KNN modelling, firstly train the model with default parameters (no parameters), then run the hill-climb feature selection algorithm. Secondly, test the model's accuracy by changing values of each parameter of the decision tree model using the selected features. The visualised decision tree is in figure_D in the Results section.[6]

Improving the model's accuracy by tunning values of the parameters 'criterion', 'max_depth' and 'min_samples_split'.[6] Again, the classification error rate, precision, Recall and F1-score are summarised in table_3.3b, comparison is made between the model with default (no parameters) and the model with tuned parameters.

(3.3c) Random Forest (RF) Modelling

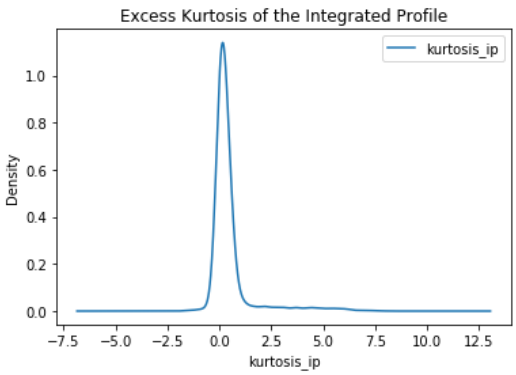
Similarly as the procedures for the other two models, firstly train the model with default parameters (no parameters), then run the hill-climb feature selection algorithm. Secondly, test the model's accuracy by changing values of each parameter of the random forest model using the selected features.

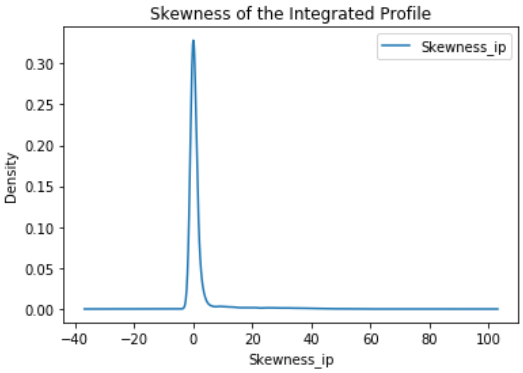
Improving the model's accuracy by tunning values of the parameters 'n_estimator' and 'criterion'. Again, the classification error rate, precision, Recall and F1-score are summarised in table_3.3c, comparison is made between the model with default (no parameters) and the model with tuned parameters. (Patel, 2017)**[7]**

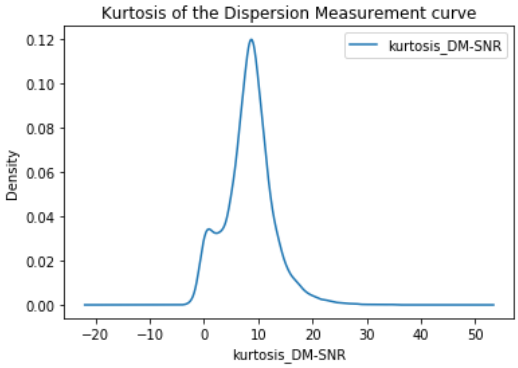
4. Results

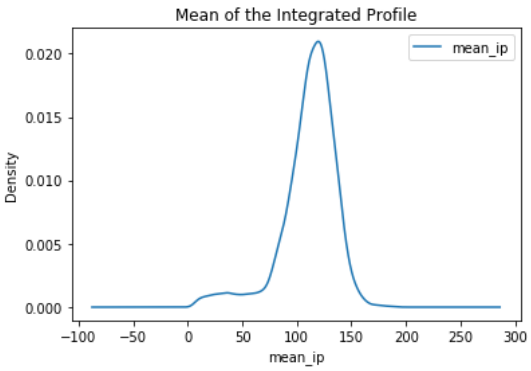
Data exploration

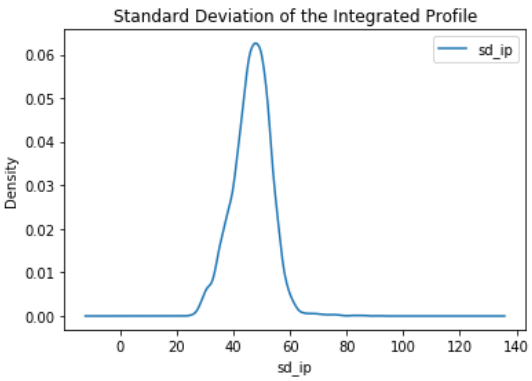
Explore the distribution for each feature:

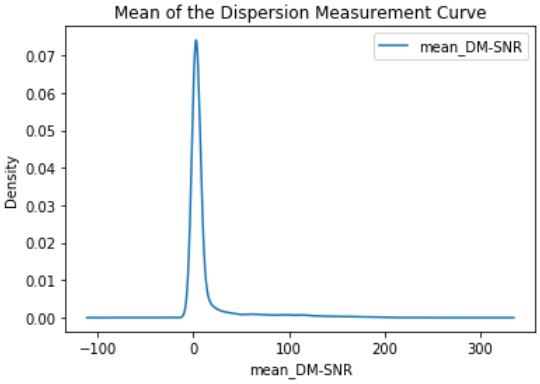


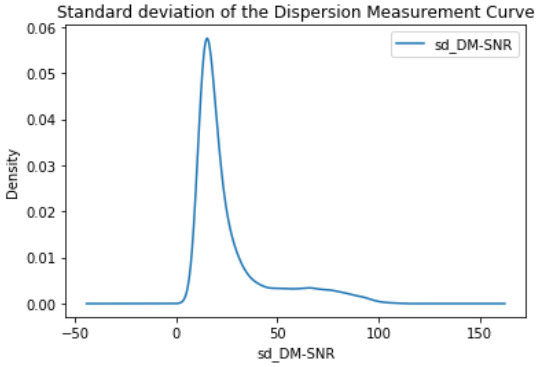


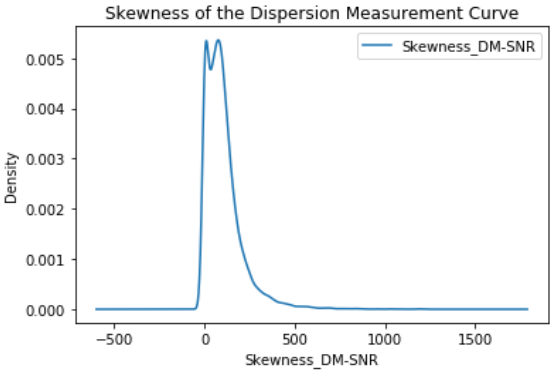




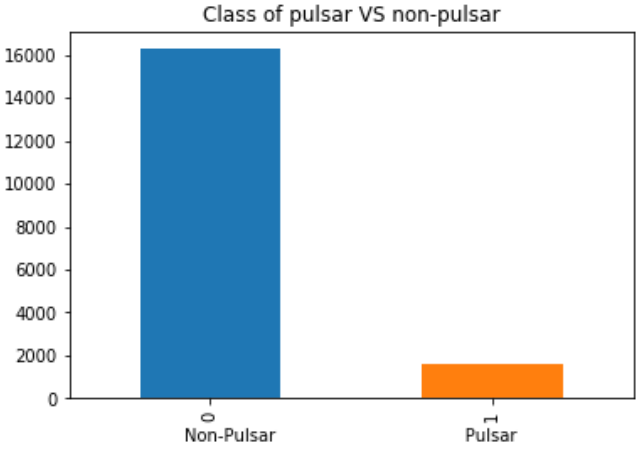






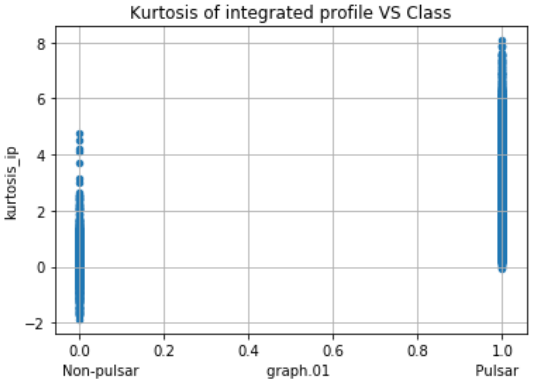


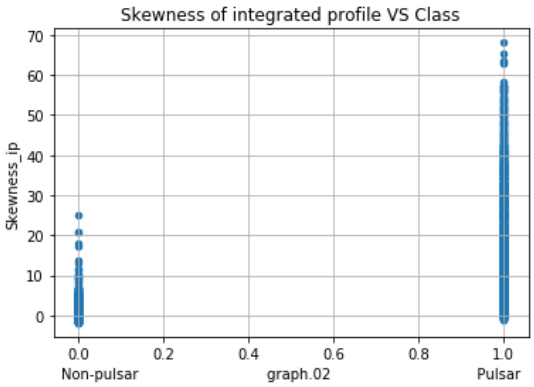
The distribution of the ‘Class’ variable (the target)

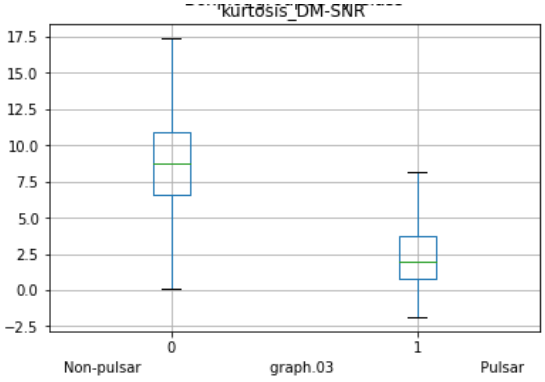


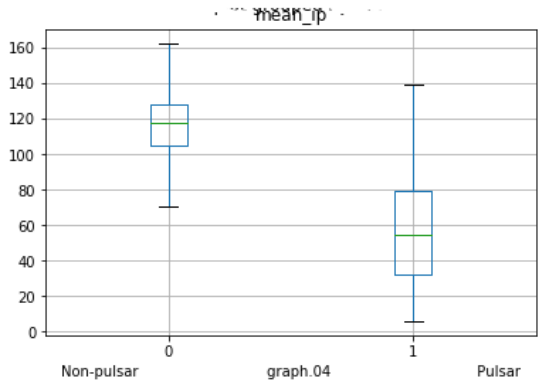
graph-9

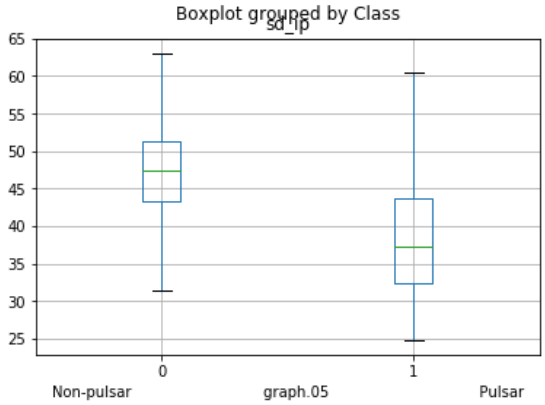
Exploration of the relationship between each feature and the 'Class' variable:

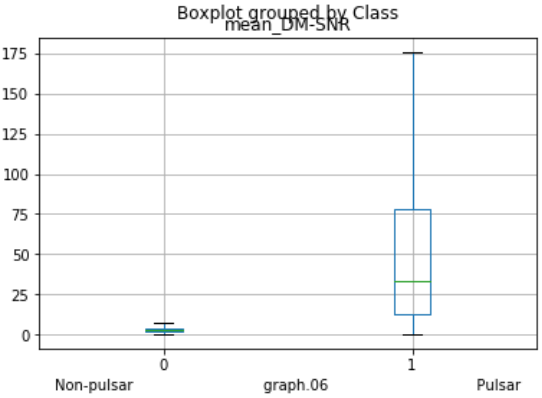


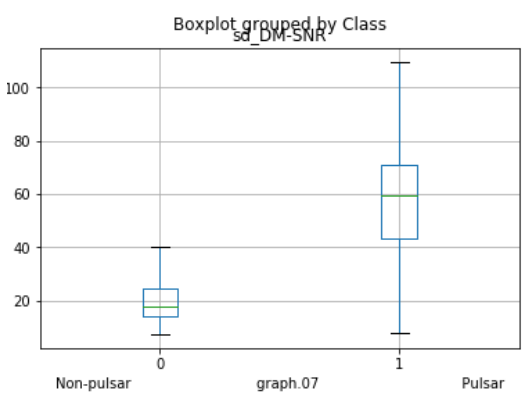


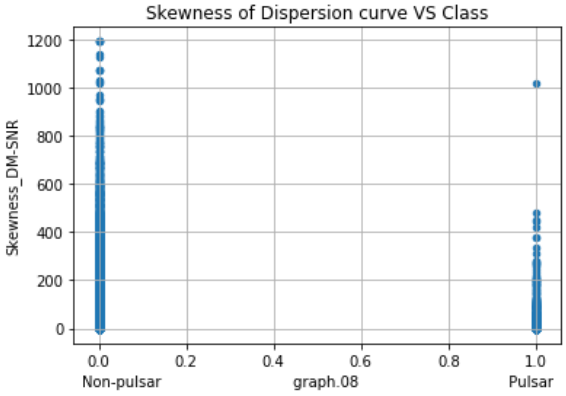




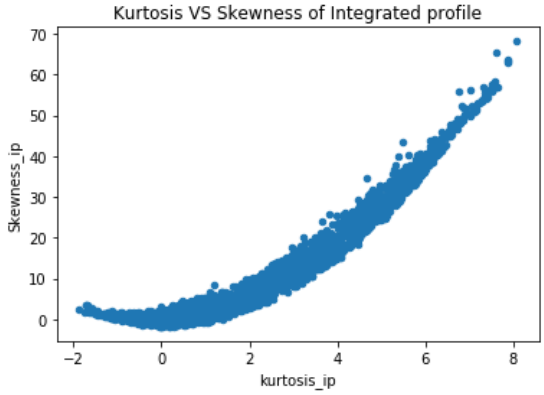




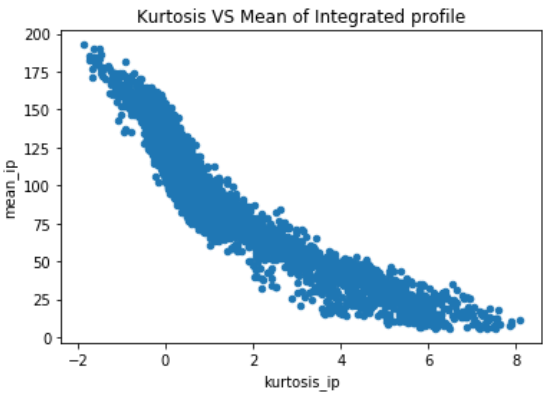




Explore the relationship between the most important and the less important features:

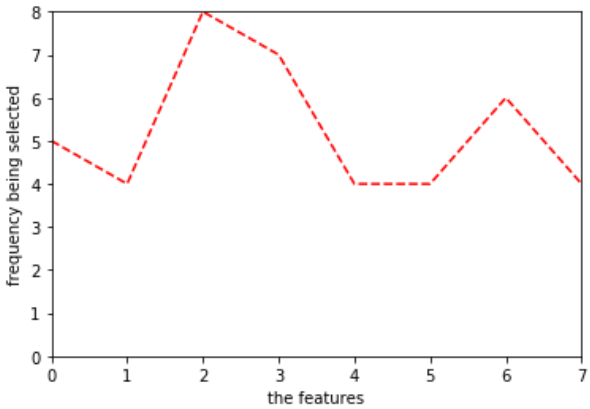


graph.09



graph.10

Feature selection



figure_F. Every single feature has been selected for at least four times out of eight times of running the algorithm.

Modelling and Evaluation – results for (3.3a) KNN model

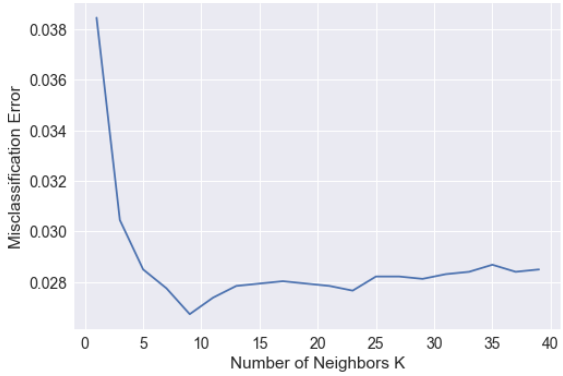


Figure-K. The optimal k value is 9 and is yielding the lowest error rate.

Table_K1

K = 1500	weights='distance'	weights='uniform'
	[6547 19]	[6552 14]
Confusion Matrix	[196 398]	[225 369]
Classification Error Rate	0.033379888	0.033379888
Precision (1=Pulsar)	0.95	0.96
Recall (1=Pulsar)	0.67	0.62
F1-Score (1=Pulsar)	0.79	0.76
weights='uniform'	p = 1	p = 2
	[6556 10]	[6552 14]
Confusion Matrix	[243 351]	[225 369]
Classification Error Rate	0.035335196	0.035335196
Precision (1=Pulsar)	0.97	0.96
Recall (1=Pulsar)	0.59	0.62
F1-Score (1=Pulsar)	0.74	0.76

Table_K1 shows that given the best k value, select the best value for parameter 'weight', and then given the best k and the weight value, select the best value for the next parameter p, and so on.

Table_K2

		Default	Tuned Parameter
Confusion Matrix		[6469 56] [137 498]	**[6471 54]** **[133 502]**
Classification Error Rate		0.027	**0.026**
Precision	0 = non pulsar	0.98	**0.98**
	1 = pulsar	0.90	**0.90**
Recall	0 = non pulsar	0.99	**0.99**
	1 = pulsar	0.78	**0.79**
F1-Score	0 = non pulsar	0.99	**0.99**
	1 = pulsar	0.84	**0.84**

Modelling and Evaluation – results for (3.3b) DT model

figure_D The visualised decision tree

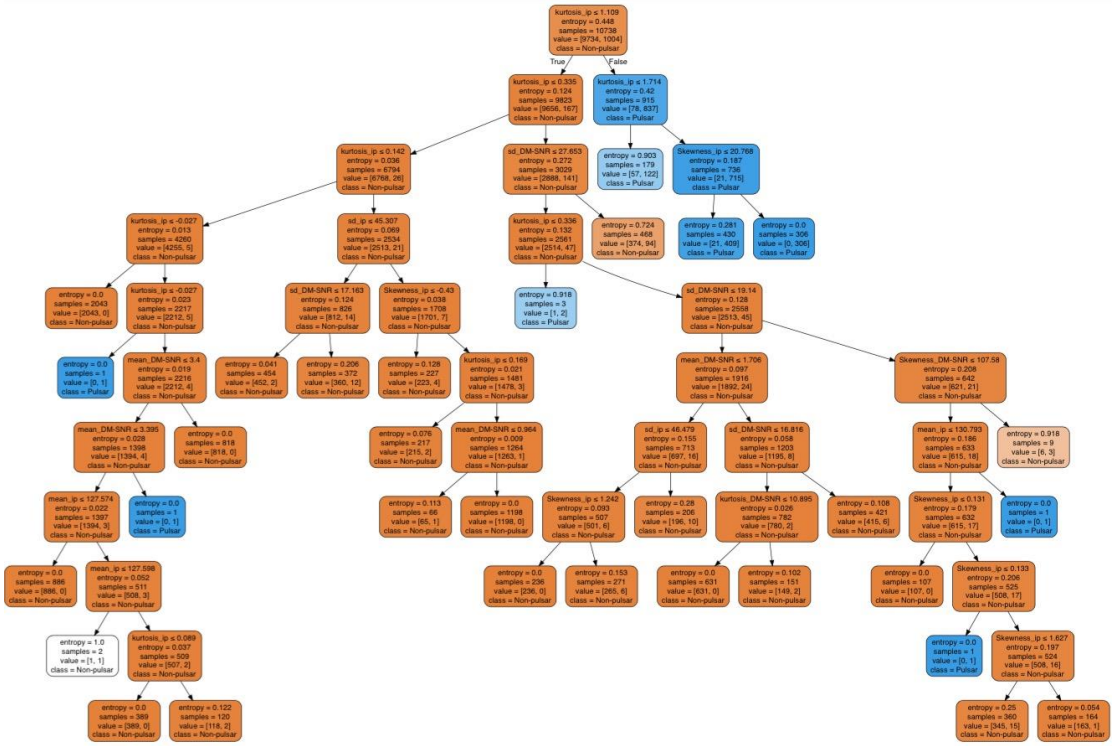


Table_3.3b

		Default	Tuned Parameter
Confusion Matrix		[6390 135] [100 535]	**[6420 105]** **[100 535]**
Classification Error Rate		0.033	**0.029**
Precision	0 = non pulsar	0.98	**0.98**
	1 = pulsar	0.80	**0.84**
Recall	0 = non pulsar	0.98	**0.98**
	1 = pulsar	0.84	**0.84**
F1-Score	0 = non pulsar	0.98	**0.98**
	1 = pulsar	0.82	**0.84**

Modelling and Evaluation – results for (3.3c) RF model

Table_3.3c

		Default	Tuned Parameter
Confusion Matrix		[6484 41] [92 543]	**[6478 47]** **[88 547]**
Classification Error Rate		0.018	**0.018**
Precision	0 = non pulsar	0.99	**0.99**
	1 = pulsar	0.93	**0.92**
Recall	0 = non pulsar	0.99	**0.99**
	1 = pulsar	0.86	**0.86**
F1-Score	0 = non pulsar	0.99	**0.99**
	1 = pulsar	0.89	**0.89**

5. Discussion

Data Exploration

The density graph shows that the integrated profile (ip) has a more normalised distribution than DM-SNR curve. The ip's zero kurtosis means the variation is quite flat, and zero skewness indicates that it is centrally symmetrical. Whereas the DM-SNR curve's distribution has big skewness (long tail) and positive kurtosis values indicating that wide range of peaks are present. Hence the four statistics attributes of ip can better represents the characteristics of pulsars in reality, analysis based on ip features can give higher accuracy in terms of identifying a pulsar.

Some of the correlation graph between each feature and the class variable shows the characteristics of pulsars and the others show characteristics of non-pulsars. There are a lot of outliers being eliminated in order to give a clear visible difference between the two classes. For example, graph.01 can be generalised that pulsars tend to have positive excess kurtosis values, and whenever a candidate has negative kurtosis it most likely a non-pulsar. Graph.04 summarises that pulsars have a mean around 60 whereas non-pulsars' mean are twice as large as pulsars. So the kurtosis of ip and the mean of ip are 'helpful' features in distinguishing between pulsars and non-pulsars.

Feature selection

For each of the three models, the hill-climb algorithm has been executed eight times. Each time, the frequency of each feature being selected is recorded. Overall, every single feature has been selected in the output for more than four times (four is the median of eight). This concludes that every feature is important in determining the class variable. Hence all eight features will be used in the proceeding analysis.

Performance of Data Modelling

The K-Folds cross validation has split the data into five folds to randomise data, and reduced the impact of imbalanced dataset.

In KNN model, as shown in figure_K, when K equals to 9, the misclassification error drops to a lowest point. This means that including nine neighbours is enough to give a correct classification for a pulsar candidate. Then based on the nine neighbours, the factors that uniform weight and p = 1 make the KNN model reach the highest accuracy, meanwhile the rest of the parameters are proved to be have no impact on the model's accuracy since there is no obvious change in the output's precision.

So the best model is when k=9, weight = uniform and p = 1. This model correctly classified real pulsars for 90% of the time and correctly classified non-pulsars for 98% of the time, as shown in Table_K2 that the precision score for pulsar is 0.90 and for non-pulsar is 0.98.

This makes sense because a highly imbalanced dataset will have the problem that any model could correctly label the majority class with a precision score approaching 100% precise whereas the minority class can only be labelled with a much lower accuracy, according to Chawla. (2004) [4]

Similarly, in the decision tree model and random forest model, the precision score for correctly classified real pulsars are respectively 0.84 and 0.92, compared to the KNN model which gives 0.90. Hence the random forest model with parameters n_estimators=23, criterion='entropy', is the most accurate one to correctly classifies a real pulsar star from the pulsar candidates.

Furthermore, in comparison of the misclassification error rate of the three models, KNN is 0.026, Decision tree is 0.029 and Random Forest is 0.018, the random forest model has the lowest error rate, which again proves to be the optimal model among the three.

6. Conclusion

After applied the three classification models, it turns out to be the random forest model that gives the highest classification accuracy and lowest error rate. For the given HTRU2 dataset and in this context, the random forest model with parameters n_estimators=23 and criterion='entropy' is the best model to classify the pulsar candidates. However, the best model in this context and its suitable parameter values may not be applicable to another set of pulsars candidates, due to the highly imbalanced nature of the pulsar stars. More knowledge and experience is needed to train a model that can reduce the impact of imbalance to the minimum, and this might be a new research question opened up.

7. References:

[1] Lyon, R. J. (2016). *WHY ARE PULSARS HARD TO FIND.* England: The University of Manchester.

[2] XU Yu-yun, L. D.-j.-c. (2017, August). Application of Artificial Intelligence in the Selection of Pulsar Candidate. *PROGRESS IN ASTRONOMY, 35*, 313. doi:10.3969/j.issn.1000-8349.2017.03.03

[3] Lyon, D. R. (2017, 2 14). *HTRU2 Data Set* . Retrieved from UCI Machine Learning Repository: https://archive.ics.uci.edu/ml/datasets/HTRU2

[4] Chawla. (2004). Imbalanced learning problem

[5] Yongli Ren, A. M. (2018). Practical data science - Tute4/ Week 5, Tute5/ Week6, Melbourne, Victoria, Australia.

[6] Ren, Y. (2018). Practical Data Science: Classification lecture slides. Melbourne, Victoria, Australia.

[7] Patel, S. (2017). *Chapter 5: Random Forest Classifier*. Retrieved from https://medium.com/machine-learning-101/chapter-5-random-forest-classifier-56dc7425c3e1