Discovering pulsars and transients with intelligent algorithms

Alex Lisboa-Wright
8928493

School of Physics and Astronomy
The University of Manchester

Fourth Year MPhys Report

May 2017

Project undertaken in collaboration with Lewis Smith (ID 8933715)

Project supervised by Dr Michael Keith

This is the final report of a full-year project

Abstract

This project aimed to use intelligent algorithms to improve the detection of pulsars and fast radio bursts (FRBs). For pulsars, supervised-learning algorithms were used. A statistically significant improvement in the rate of discovery of real pulsars in the data was achieved, by adding simulated pulsars to the training data. For FRBs, unsupervised-learning algorithms, which were trained only on noise data, were assessed to be more useful due to the lack of representative FRB datasets. In particular, the Isolation Forest classifier was shown to be both time-efficient and effective in finding FRBs.

1 Introduction

1.1 Pulsars and millisecond pulsars

Pulsars are rotating neutron stars, formed during supernova (SN) explosions, whose magnetic axes are sufficiently aligned with the Earth that radiation from the magnetic polar regions can be detected. They have high rotation frequencies (low periods) due to conservation of angular momentum holding during the supernova event as the star radially sheds mass. Millisecond pulsars (MSPs) are a class of pulsars with unusually high frequency - high enough that they are thought to have gained additional angular momentum from a source other than the SN in which they are born, such as a companion star [13].

1.2 Fast radio bursts

Fast radio bursts (FRBs) are transient (single-event) objects visible in the radio spectrum. They are characterised by short durations (less than 5 milliseconds) high signal-to-noise ratios (SNRs) and high dispersion measures (DMs) (see Section 3) after standard observational data processing, leading to their description as astrophysical sources of extragalactic origin. They are rare (18 confirmed as of the date of submission of this report, including a single repeating source [1]) and they can share similarities with terrestrial signals, including detections known as "perytons", which were later found to be caused by radiation from a microwave oven [2]. These features combine to make FRBs extremely enigmatic and their discoveries difficult.

1.3 Aims and motivation for the project

The underlying aim of this year-long project was to explore ways of improving detection of pulsars, with a particular focus on MSPs, and fast radio bursts using intelligent algorithms. This field of study is motivated by the need for larger populations of confirmed pulsars and FRBs in order to better understand their underlying physics (and in the case of FRBs, their physical sources). As with other neutron stars, pulsars are useful as testing objects for general relativity, as markers of stellar populations due to their origin and for the physics of their degenerate neutron interiors and the structure associated with these. Due to the fractionally very small rate of change of the period between pulse signal for pulsars, any significant variations in the arrival times of the pulses, factoring in the motion of the Earth, are due to massive objects near the pulsar. This last application is especially relevant to MSPs. Fast radio bursts are of high interest in radio astronomy due to their unknown nature of their sources and the indication that the sources are located outside the Milky Way.

The Square Kilometre Array (SKA) is a series ofs array of dipole antennae and 15m-diameter radio dishes, covering a total area of circa 1km^2, which is currently under construction. Its purpose is to provide a much higher-resolution and higher-sensitivity radio picture of the sky, with myriad scientific objectives ranging all the way up to cosmological observations. It is due to come online within the next decade. The rate of data produced will be larger than that of the internet. Hence, storing all the SKA data will be impossible. Intelligent algorithms are being developed to

classify data in real-time as it is collected by radio telescopes as data streams, with the SKA being the foremost application of this.

In this project, improvements to pulsar detection were attempted using observational datasets and features employed by previous research, simulating pulsar data points and combining these with the observational data to produce better training data for the intelligent algorithms. Improvements to FRBs involved finding a reliable set of features, given the lack of research, carried out using feature selection stability testing, and testing these features on real noise and simulated FRB data.

1.4 The High Time Resolution Universe Survey

HTRU is a survey conducted, initially using the Parkes Radio Telescope in Australia for the southern hemisphere, and later, the Effelsberg Radio Telescope in Germany for the northern hemisphere additionally from 2010 onwards [3][4], with the purpose of scanning the entire plane of the sky for pulsar and transient signals at the highest resolution used so far in the field. It uses multi-beam receivers to increase the resolution and observes at a radio frequency of 1.4 GHz.

1.5 The GHRSS survey

The GMRT High Resolution Southern Sky survey is a survey conducted with the Giant Metrewave Radio Telescope (GMRT) at a frequency of 322 MHz, for detecting pulsar and transient signals [8]. Data recorded by this survey was used in this project as the noise data in the FRB datasets, to which simulated FRBs were added.

2 Experimental background - astrophysical measurements

This section summarises the main observational astrophysical quantities used in the experimental data, which are useful in distinguishing pulsars and FRBs from other sources of radio emission, such as radio frequency interference (RFI) or general astrophysical background emission. These quantities are the period of the emission (central to the pulse profile below), the signal-to-noise-ratio (SNR) and the dispersion measure (DM) measure. For FRBs, the relevant quantities are the DM, the SNR and the boxcar width.

2.1 Pulse profile

The pulse profile is a histogram of the average amplitude of the received radio emission as a function of the phase of an average pulse period. This profile is shown in the bottom left panel of Figure 1. By comparing the panels of the two figures, it is clear that a signal from a pulsar will have a more obvious curve (in red) and a clear peak or peaks (since the range of the phase extends beyond a single pulse, so the early part of the pulse repeats). Pilia et al. [5], using the LOw-Frequency ARray (LOFAR), demonstrate that there is a wide variety of pulse profile shapes, including multiple-peaked and heavily asymmetric profiles.

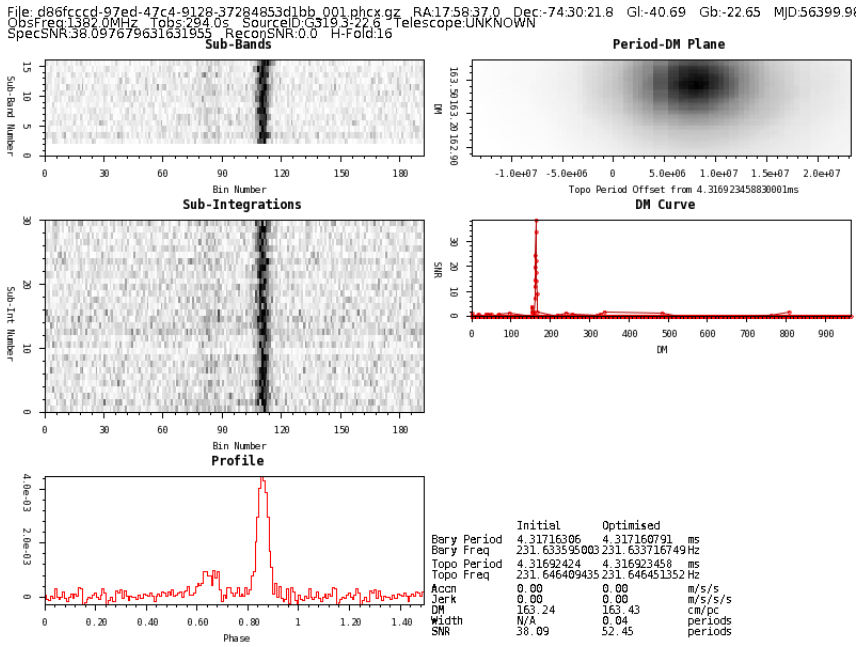

Figure 1: psrsoft image output for a simulated pulsar data file

2.2 Signal-to-noise ratio

The SNR is simply the ratio of the power received from the source itself (the signal) versus that from other (background) objects, such as the cosmic microwave background (CMB), the atmosphere, the interstellar medium (ISM) or other origins (depending on the observing frequency). This could potentially include the telescope instrumentation. Readings with a higher SNR have a smaller relative error via signal averaging statistics, in which the SNR is proportional to the signal mean divided by the noise standard deviation. Therefore, higher SNRs are more likely to represent real source emission. The SNR can, however, be rendered less effective as a classification feature by RFI [6] and must therefore be used in conjunction with other features when evaluating a source.

2.3 Dispersion measure

The dispersion measure (DM) is defined as the integral of the electron number density, n_e, over the line of sight to a given astronomical source [9]:

$$\mathrm{DM} = \int_0^L n_e dl \tag{1}$$

where L is the line-of-sight distance to the source. The unit of the DM is usually given as $\mathrm{cm}^{-3}\mathrm{pc}$. The DM is thus an electron column density, along the line-of-sight, between the observer and source. The DM is important because an ionized medium (such as the ISM), if situated along the line-of-sight, can cause the light emitted by the source to become dispersed by refraction or absorption and re-radiation, depending on the wavelength (and therefore the frequency) of the incident radiation. This causes a wavelength-dependent delay in the time at which the signal is detected. For two different frequencies, f_1 and f_2, this time delay is given in SI units by:

$$\Delta t = \frac{e^2}{2\pi mc}(f_1^{-2} - f_2^{-2}) \times \mathrm{DM} \tag{2}$$

It is this frequency-dependent delay which is detected by the telescope's instrumentation, from which the DM is calculated. The DM is therefore an indicator of the amount of gas between an observer and a given source of emission. An example of the inverse square-law relation in Equation (2) can be seen in Figure 2.

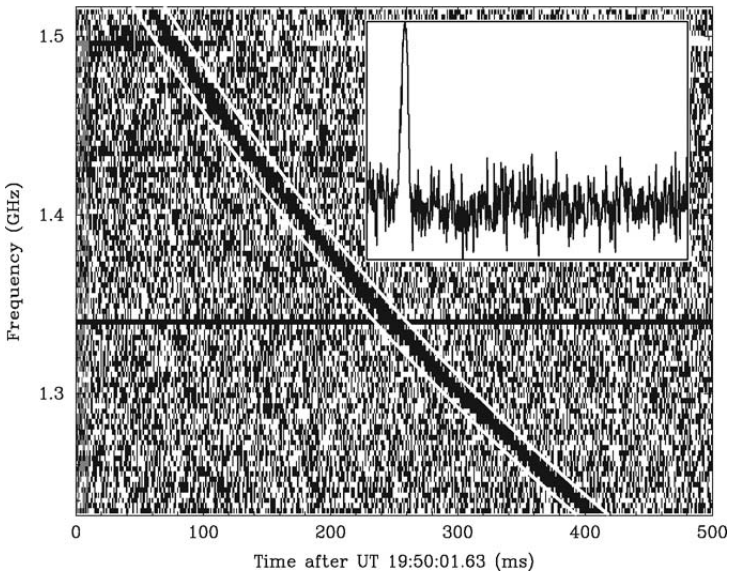

Figure 2: Frequency-time plot for the Lorimer burst, the first confirmed FRB. Source: [10]

2.4 Boxcar width

The boxcar width is essentially the time duration of a pulse of radio emission. A boxcar (top-hat) function is used as a smoothing function for the signal by convolving the pulse with the boxcar. Various widths of the boxcar function are tested, and the width taken forward is that which produces the highest SNR for the signal being studied. This width is treated as the duration of the signal.

This measure is important when searching for signal classes which are at least partially characterised by their duration - including FRBs, which last for less than 5 milliseconds. Boxcars are used in preference to Gaussian fits due to being much less complex mathematically while still providing reasonable accuracy for pulse duration estimation. This is highly advantageous in radio astronomy, where there are large quantities of received signals (from both astronomical and other sources) needing to be processed in even short observations.

3 Data and pipelines

3.1 Pulsar data

This project used the PulsarFeatureLab processing pipeline created by Lyon et al. [11] to process all data, including simulated data, to extract the desired features. It

is designed to be compatible with various data file formats and has multiple output file type options. In addition, its feature extractor program can output any of a selection of feature groups, although it can be modified to take other feature sets as arguments.

3.1.1 Observational data

The observational pulsar datasets used for this project were:

- HTRU1: Dataset from HTRU processed by Morello et al. [12]. Contained 74 classified MSPs, 1122 other classified pulsars and 89996 noise instances.
- HTRU2: Dataset from HTRU processed by Lyon et al. [11] using PulsarFeatureLab. This was the principle test dataset in the pulsar part of the project. Contained 71 classified MSPs, 1568 other classified pulsars and 16259 noise instances.
- LOTAAS: A dataset from the LOFAR Tied-Array All-Sky Survey (LOTAAS). This did not contain any classified MSPs and was therefore not useful for classification in this project.

In this project, MSPs and other pulsars (all treated during classification as being "class 1") were distinguished by defining a threshold pulse period, below which classified pulsars were treated as MSPs and above which they were treated as non-MSPs. The definition described by Lorimer [13] of $P_{MSP} \lesssim 30$ ms was used as a guideline. The MSPs in the datasets, in order to include a few pulsars with periods slightly above 30ms, were defined as $P_{MSP} \lesssim 31$ ms. This maximised the number of MSPs, which were used as a testing set for the classifiers. Even so, there were very few examples of MSPs in the datasets, even relative to the number of other pulsars. For this project, the output pulsar data files were chosen to be in .arff (Attribute-Relation File Format) format in order to be compatible with the Waikato Environment Knowledge Analysis (WEKA) machine learning tool [14]. WEKA was used to visualise and count the different data types. The purpose of this was to find relationships between features to optimize the simulated pulsar data created during the project.

Candidates are classified by filtering out the most promising objects, then processing the data into graphical form (see Figure 1), which can be analysed by eye as having characteristics of a pulsar [12]. Pulsars can be determined by a relatively narrow and dark vertical region in the sub-band and sub-integration panels, a relatively compact single dark region in the period-DM plane, a profile histogram that forms something that is approximately a continuous curve and a single DM-SNR peak, since the pulsar is a single source. Therefore, the object in Figure 1 is a good example of a pulsar, while the object in Figure 16 (see Appendix) is clearly not. These images were generated using the psrsoft shell script [15].

It was discovered that a significant fraction of the labelled millisecond pulsars (12 objects) in HTRU2 were not distinct pulsars. Instead, they proved to be one of the following:

- harmonics of detected pulsars (i.e. signals with integer multiples of the actual pulse frequency of the detected object), distinguished in psrsoft by having multiple lines in the sub-integration and sub-band plots and several peaks in their pulse profiles, all while having the same optimised SNR and DM outputs as the original detection.
- RFI (i.e terrestrial inteference). This is characterised in psrsoft by a DM value that is very small (usually <10 pc cm^{-3}), as well as by a linear distribution of counts along a line of constant DM in the period-DM plane, indicating a signal with a continuous distribution of different periods and which is thus aperiodic.
- a noise detection which displayed some, but not all, criteria used to define a pulsar by eye.

A few real objects also had unusual DM-SNR curves, in that their curves had far more points than other real objects, which changes the shape of the curves to appear less smooth (since the curve simply connects the plotted points, as shown in Figure 1) and therefore the relevant statistics. It was found that the observations in question were performed in the same observation season, so it was attributed to a change in the data-processing pipeline used only during that time period.

3.1.2 Simulated MSP data

The simulated pulsars were created with a random distribution of SNRs greater than 5 and uniform distibution of periods in the range of 5-20 milliseconds. This period range was chosen in order to coincide with the expected region of future MSP discoveries, especially MSPs which are easier to distinguish from non-MSP pulsars. The simulated pulse profiles were injected into real noise data files to make the curves more realistic than injecting them directly into the pipeline alone, as the curves would have benn far too smooth and therefore too easily separated from real data. To generate the final simulated data the injection was performed on a multiple-core GPU, due to the large processing power required. This produced a series of directories, each containing one simulated MSP data file and several noise files. The process used injection software provide by DSPSR [16] and Tempo2 [17].

Extraction of the simulated pulsar file was achieved by applying a score to each file. The score was a sum of the squares of the fractional deviations, of both the pulse period and the DM value, from the original injected data values. Therefore, the pulsar file would ideally have a score of zero. However, the fitting during the process allowed the parameters to be adjusted, hence the need for the score. The program compared the scores within each directory and took the lowest score forward, subject a strict tolerance level of deviation from the original injected values. The tolerance for the frequency/period was selected to be the maximum size of the Doppler effect due to the orbital motion of the Earth, while the DM tolerance was set to be 0.7. The feature data was extracted and converted into ARFF format using PulsarFeatureLab.

3.2 FRB data

For training classifiers, given the small number of confirmed FRBs, the data was mostly observed noise, denoted as class-0 objects, together with simulated FRBs.

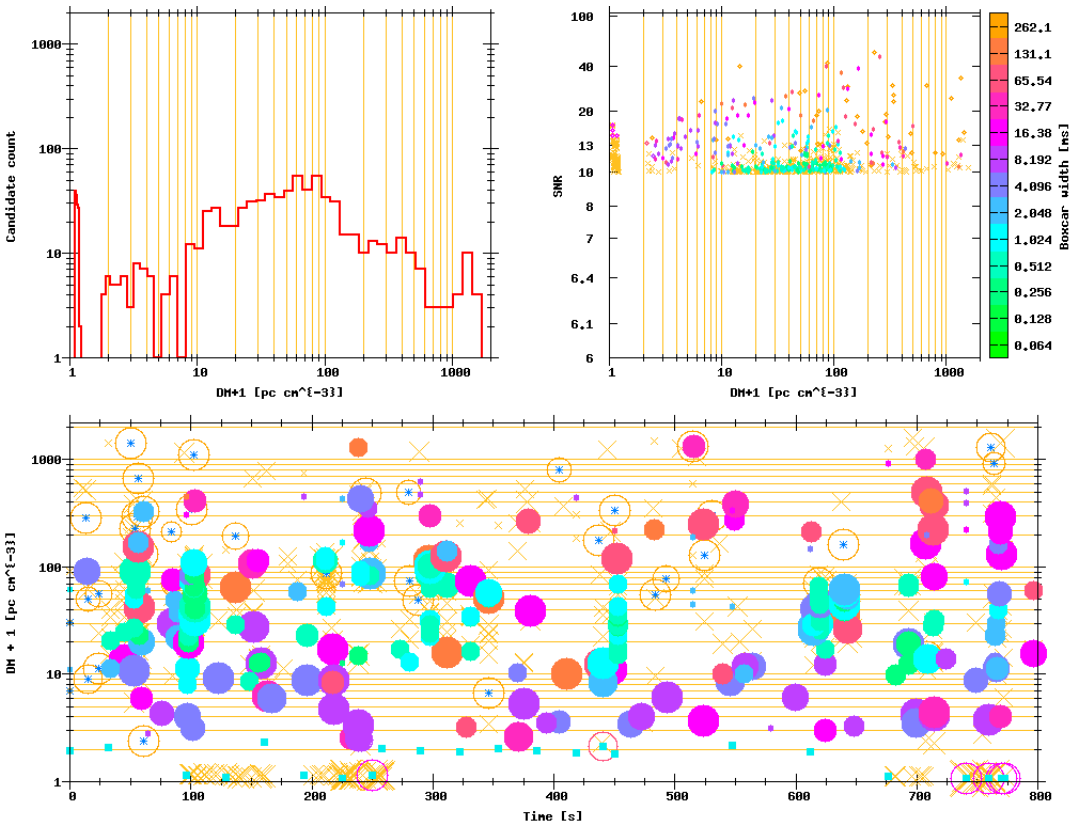

Figure 3: Example of the Heimdall output

The simulated FRBs were produced, injected into the data and then identified using the Heimdall pulse-detection code (http://sourceforge.net/projects/heimdall-astro) and injected into the noise data as class 1 objects. This gives data as shown in Figure 3. Each circle in the DM-time plot (bottom panel) represents a data point. The size of each of the circles in the figure is determines by the SNR of that particular data point and the colour indicates the optimal boxcar width of the data. As singular objects with small, FRBs might be expected to stand out as isolated points in the SNR-DM plot (top-right panel) and in the DM-time plot. Given the brief and transient nature of FRBs, they are unlikely to make an impression on the candidate count-DM plot (top-left panel).

3.2.1 Feature selection

Having obtained the data, it was then necessary to select the features to be used for classification. In particular, it was necessary to make a stable feature selection, i.e. select a set of features that is useful for classification of different training datasets. This was done by compiling a list of potential features, then applying these features to a group of similar ("perturbed") datasets to test their stability. These datasets were created from the dataset containing all relevant data by using k-fold cross validation. The value of k was set to be 5 in order to maintain small perturbations between the datasets produced. The joint mutual information (JMI) of the features (see Section 4.1) was calculated for each dataset, and the features were ranked according to their JMI values, as before with the pulsar data. Then, the n features with the highest JMI scores for each dataset are selected as the best feature set for

that dataset, with $n < k$. For k datasets and f features, the objective is to produce the following matrix, A:

$$A = \begin{bmatrix} \mathbf{s_1} \\ \mathbf{s_2} \\ . \\ : \\ \mathbf{s_k} \end{bmatrix} = \begin{bmatrix} x_{1,1} & x_{1,2} & ... & x_{1,f} \\ x_{2,1} & x_{2,2} & ... & x_{2,f} \\ . & . & & . \\ : & : & & : \\ x_{k,1} & x_{k,2} & ... & x_{k,f} \end{bmatrix}$$

where for all i and j, $x_{i,j} \in [0, 1]$, i.e. the elements of A are binary. If $x_{i,j} = 1$, the jth feature in the ith dataset has a JMI score which is in the top n values for the ith dataset, and is "selected" [18]. Otherwise, the element is zero and is not selected. The objective of creating this matrix is to check the similarity of all the individual rows (which are the results from each dataset) - a stable feature selection will produce a matrix of identical rows. Otherwise, small perturbations in data will directly affect the suitability of features, rendering the selection unstable, which is potentially catastrophic for observational searches for FRBs. This is shown in the example below, with $n = 2$, $f = 3$ and $k = 3$. Note the difference between rows in A_{stable}, which are identical, and $A_{unstable}$, where each dataset returns a different optimal set of features:

$$A_{stable} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \end{bmatrix}, A_{unstable} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

There are a variety of causes for instability in feature selection - it can be due to one or more of the following:

- the datasets may vary significantly, such as one dataset having all the FRBs or many more high-SNR noise data points. This may cause some features to be significantly more or less effective. The value of k must be chosen carefully, therefore, to avoid this scenario while also being large enough to test the feature selection stability for other weaknesses.

- a number (very close to, or greater than, the value of $f-n$) of the features given to the JMI calculator may produce very small JMI scores which are sensitive to random fluctuations over repeated runs. To solve this problem requires a large number of features which give JMI scores high enough to avoid fluctuations becoming significant. It should be noted that the available data must enable such features to be used - this is not guaranteed to be the case.

While stability can be visualized, particularly in simple cases like the above example, it must be quantified in order to be useful. This requires numerical similarity measures or functions, ϕ, such that the overall stability measure, $\hat{\Phi}_A$, can be calculated via:

$$\hat{\Phi}_A = \frac{1}{k(k-1)} \sum_{i=1}^{k} \sum_{j=1, j \neq i}^{k} \phi(\mathbf{s_i}, \mathbf{s_j}) \tag{3}$$

Nogueira and Brown (2016)[18] list the properties of an ideal ϕ function in the general case and compare them with those of functions currently in use. Following their

recommendations with regard to these properties, the Pearson correlation coefficient was used to define ϕ as follows:

$$\phi_{Pearson}(\mathbf{s_i}, \mathbf{s_j}) = \frac{\sum_{m=1}^{f}(x_{i,m} - \bar{x}_i)(x_{j,m} - \bar{x}_j)}{\sqrt{\sum_{m=1}^{f}(x_{i,m} - \bar{x}_i)^2}\sqrt{\sum_{m=1}^{f}(x_{j,m} - \bar{x}_j)^2}} \tag{4}$$

where $\bar{x}_i = n/f$ It should be noted that $\phi_{Pearson}$ has a lower boundary of -1, not 0. However, it does returns 0 for a completely unstable (random) FS process, as required.

4 Summary of machine learning

Machine learning, as the name implies, is a process in which a computer is able to learn and adapt without detailed or specific additional programming being necessary. The mathematical objects which undergo machine learning are know as "intelligent algorithms" for the same reason. Machine learning in data analysis can be broadly divided into supervised and unsupervised learning:

- Supervised learning: the algorithm (or "classifier") is given a set of features ("inputs") to use for classifying data and a set of output classes ("targets") in which to place the data after the process, which are the only responses the algorithm can use.
- Unsupervised learning: as before, the classifier is given the data and input features, but nothing else - the data points are not given a class at all, and the classifier looks for trends in the data. The data points which follow this trend, within a certain threshold known as the anomaly score, are assigned to one class, while data points with higher anomaly scores are classified as "anomalous", i.e. not of the first class, which does not necessarily group them in a single class (although for the relevant datasets in this project, there are only two classes in the data).

Machine learning is applied to classify data by optimizing, or "fitting", a mathematical model to the training dataset, in order to best separate the different classes by an N-dimensional hyperplane (where N is the number of features used for classification) described by the model. This model is then used to classify the testing set, produced the final classification results. The nature of the model is important, and therefore the process of fitting it, while also avoiding "overfitting" or "underfitting" it, is of paramount importance. An underfitted model (left-hand panel in Figure 4) is one which is too simple and could be more complicated while remaining applicable to different datasets (for example in cross-validation).

An overfitted model (right-hand panel), by contrast, is specific to only one dataset - it adapts to separate the red and blue points almost perfectly in that dataset, but its accuracy is too sensitive to changes in the data, which will occur if it is applied to another dataset, thereby making the model virtually useless. An ideal model (centre

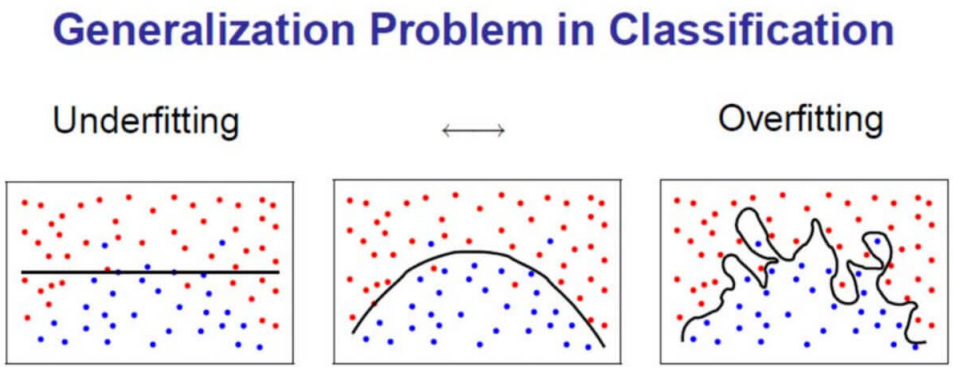

Figure 4: Underfitting (left) and overfitting (right) of a classification model (solid black line) for the same dataset. Source: https://tomrobertshaw.net/2015/12/introduction-to-machine-learning-with-naive-bayes/

panel) has some level of complexity to adequately separate the classes (better than an underfitted model), but is equally accurate for different datasets.
The training process for large datasets often takes the form of k-fold cross-validation, or simply cross-validation. In this, the training dataset is divided into k random subsets. One set is used to test the others against using the classification algorithm. The process is repeated k times, and the average classification results are calculated and used to determine the measures of accuracy (see below) are calculated [19]. The number of folds in cross-validation is important, as each subset needs to have enough variety within it to contribute meaningfully to the process - if it is too pure relative to the other subsets, the subset reacts differently towards the testing data subset in an adverse manner, which lowers the overall accuracy of the process. This is more likely to occur with a high k-value. If the k-value is too low, the cross-validation is not repeated enough times for the final average to be representative of the original dataset. Both cases cause the performance of the classification model produced by the cross-validation to deteriorate.

4.1 Information theory

To start classification, it is necessary to select features through which to express the original data. Determining the best set of features requires the use of information theory. For a given discrete random variable (in this case a data feature X), its information entropy, $H(X)$, is defined as a measure of the amount of information needed to describe the variable and the average information required to describe the variable [20]. It is defined as:

$$H(X) = -\sum_{x \in X} P(x) \log_2 P(x) \tag{5}$$

where $P(x)$ is the probability that $X = x$ (a.k.a. the probability mass function). From this equation, one can deduce that $0 \leqslant H(X) \leqslant 1$. The condition entropy, of a feature X given another feature Y, is similarly defined as:

$$H(X|Y) = -\sum_{y \in Y} P(y) \sum_{x \in X} P(x|y) \log_2 P(x|y) \tag{6}$$

where $P(y)$ is the probability that $Y = y$ and $P(x|y)$ is the probability of $X = x$ given $Y = y$. The joint entropy of two variables X and Y is:

$$H(X,Y) = -\sum_{x \in X} \sum_{y \in Y} P(x,y) \log_2 P(x,y) \tag{7}$$

where $P(x,y)$ is the joint probability of x and y occuring [20]. In this project's datasets, the variable Y takes the form of the class label [11], which is the variable of most interest in this dataset, as it is the output variable in this project. The mutual information (MI), $I(X;Y)$, between X and Y is the amount of information that can be determined through one variable about another and can be defined in terms of their entropies by:

$$I(X;Y) = H(X) - H(X|Y) \tag{8}$$

If two variables X and Y are statistically independent, their MI is zero. From this one can finally define the joint mutual information (JMI) of other features to a given feature X:

$$JMI(X) = \sum_{X' \in F} I(X, X'; Y) \tag{9}$$

where F is the set of features other than X [11] and $I(X, X'; Y)$ is defined as [20]:

$$I(X, X'; Y) = I(X; Y|X') + I(X'; Y) \tag{10}$$

4.2 Classification measurements

The results of binary classification come in the form of the true positive (TP), false positive (FP), true negative (TN) and false negative (FN) totals, which denote the number of each class (positive or negative) that were classified correctly or incorrectly, respectively, by the classifier. These can be summarised as the so-called confusion matrix [21], C:

$$C = \begin{bmatrix} TP & FP \\ FN & TN \end{bmatrix}$$

These numbers are important, as they define ta number of measures of accuracy [11] used in this project. The recall (or true positive rate), R, is the fraction of all true positives that were classified correctly:

$$R = \frac{TP}{TP + FN} \tag{11}$$

The precision, P, is the fraction of the data points classified as positives which are actually positives:

$$P = \frac{TP}{TP + FP} \tag{12}$$

The specificity, S, is the negative analogue to the recall, as it is the fraction of real negatives that are correctly classified:

$$S = \frac{TN}{TN + FP} \tag{13}$$

The false positive rate, FPR, is equal to $1 - S$, as it is the fraction of negatives incorrectly classified as positives:

$$FPR = \frac{FP}{FP + TN} \tag{14}$$

From these definitions, it can be deduced that R, FPR and S are all independent of class imbalance - the relative difference between the numbers of positive and negative data in a given dataset.

These are used, in turn, to define more acrruacy measures. The geometric mean, or G-mean, G, is a measure which is sensitive to fractional misclassification, rather that the absolute value of the elements of the confusion matrix themselves, [21] and is defined as:

$$G = \sqrt{R \times S} \tag{15}$$

Therefore, for datasets with a large class imbalance, such as the datasets used in this project, the G-mean is particularly sensitive to misclassification in the smaller class, in this case the pulsars. The F-score, F, is a measure that accounts for both precision and recall, and therefore is sensitive to overall fractional errors in the model for classifying positives:

$$F = 2 \times \frac{P \times R}{P + R} \tag{16}$$

Finally, the overall accuracy, A, is the fraction of all of the data points which are correctly classified:

$$A = \frac{TP + TN}{TP + FP + TN + FN} \tag{17}$$

For datasets from radio astronomy surveys, which contain many more negatives (noise) than positives (astronomical objects, such as pulsars and FRBs), the accuracy is not particularly sensitive towards the classifier's performance on the positive class. This is problematic since, in this project, the performance on the smaller class is more important regarding the final objective (discovering astronomical sources). Therefore, the recall of the data was of greater interest than the accuracy for this project.

Receiver operator characteristic (ROC) curves (see Figure 10) are a useful and widely-applied method for visualising the performance of a binary classifier algorithm. They are 2D plots of recall (y-axis) against false positive rate (x-axis), and thus are independent of class imbalance. The curves are generated by varying the threshold above which data points are classified as positives/anomalous objects. For supervised classifiers, the threshold is determined by the probability score. For unsupervised classifiers, it is based on the anomaly score of data. Due to the variation in the threshold, there will be a point for all algorithms where the threshold for labelling points as non-negative is maximally strict, in which case all points are classified as negative ($R = TPR = 0$). There will also be a corresponding point at which the threshold is maximally generous and classifies everything as non-negative ($R = TPR = 1$). Therefore, all curves must start at (0,0) and end at (1,1).

ROC curves allow for a comparison of a classifier results with that of a purely random process, which follows the line $R = FPR$ in an ROC curve - each point has an equal and independent chance of being classified correctly or otherwise. A perfect classification, in which all data are correctly labelled, independent of the threshold value (i.e. $R = 1$, $FPR = 0$), would be a step function, with the step at $FPR = 0$ going from $R = 0$ to $R = 1$. The relative performance of algorithms can be seen visually by examining how close the curves are to the ideal curve, or, more analytically, by calculating the area under the curve (AUC). The ideal classifier has an AUC of 1 (the entire area covered by the axes) and the purely random classifier has an AUC of 0.5. The better a classifier's performance, the greater its AUC score.

Precision-recall curves (see Figure 11) are also useful. This is because the project focuses primarily on the detection of non-noise class objects. The precision is therefore of interest. Its sensitivity to class imbalance is used here, to help determine how high the user should make the threshold probability/anomaly score for a given classifier to classify a data point as non-negative. As with the ROC curve, the curve is generated by varying this threshold. The curve allows the user to determine how high the recall of a classifier (how generous the threshold score) can be without this leading to an excessively low precision value. The curve for a given classifier will start at $(R,P) = (0,1)$ and end at (1,0) as the threshold changes from being maximally strict to maximally generous. A perfect classifier would generate a curve in the form of a step function with the step at $R = 1$.

4.3 Classifier algorithms

4.3.1 Supervised learning algorithms

The algorithms used in this project covered a wide range of properties and behaviours, since the aim of the project was to improve, and therefore prioritise, overall classifier performance in a general scenario, rather than focus on particular properties of certain algorithms. The ScikitLearn algorithm modules used [22] were:

- CART_tree: Decision tree (DT) module. A decision tree separates data into different nodes depending on feature values (a process known as "recursive partitioning"), then splits the data within those nodes into more nodes and so on until the nodes contain data which best resembles the desired outputs. Small trees are easy to interpret visually as a series of logical steps (see Figure 5 in the Appendix) and trees in general are fast to construct compared to other types of logarithms, even for large datasets, and require little preparation of data. However, they are prone to overfitting and are often noisy [23].

- Multi-layer Perceptron (MLP): An artificial neural net (ANN), which takes the set of features as an input layer of neurons, with each neuron representing a feature, and generates the output neuron layer through one or more hidden layers in between. Hidden layers each have an unknown number of neurons. Each neuron in a given layer connects to all neurons in the next layer with an individual weighting [24]. The optimization is performed by reducing the errors when data points are passed along the connections. ANNs are less prone

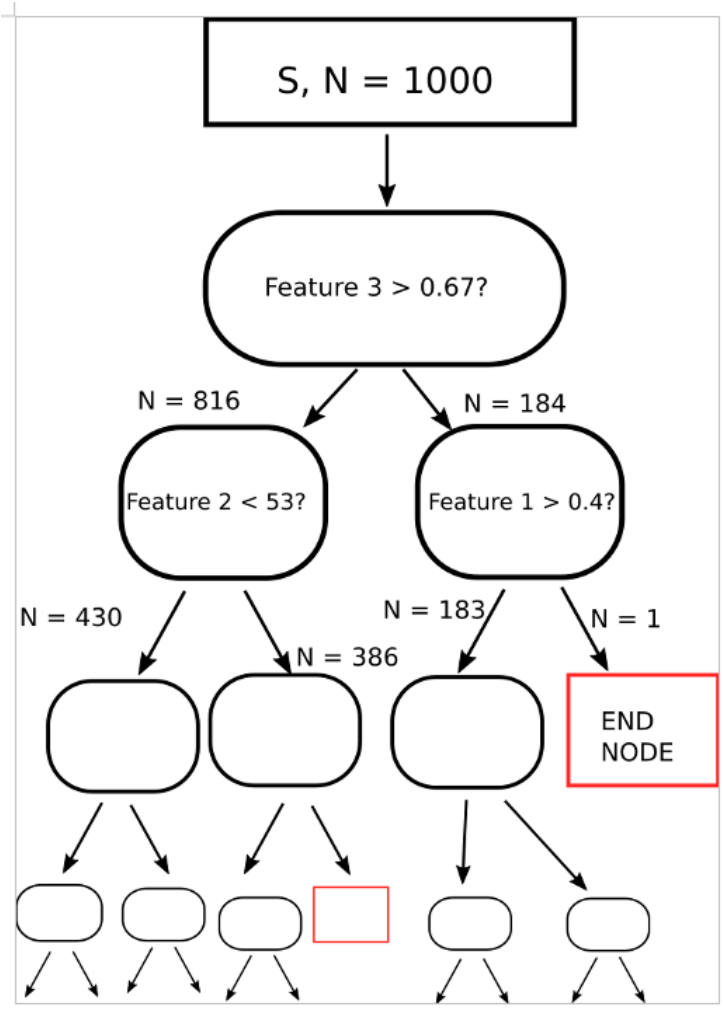

Figure 5: Diagram of a decision tree, in this case an isolation tree. The end nodes are denoted by red square boxes.

to overfitting than decision trees and can model more arbitrary functions to produce greater accuracy, but are computationally more expensive.

- NaiveBayes (NB): Uses a simple approach in which the data features are assumed to be conditionally independent, given the class, and then applies Bayes' theorem to calculate the probability of each class given the features. [25]. A variation on this classifier is the Gaussian NB classifier, in which the data for each class is assumed to be distributed according to a Gaussian probability distribution for each feature. Gaussian NB [22] was used for plotting FRB data.

- Support Vector Machine (SVM): Carries out a mapping in the N-dimensional feature space as described earlier, but is designed for datasets in which it is impossible to separate classes using a linear boundary to represent the model("non-separable" data). It defines a non-linear boundary between classes by using a transformed version of the feature space, in which a linear boundary is created. The algorithm defines a band around the hyperplane, with a margin M defined as the distance from the hyperplane to the limit of the band

on either side [23]. The perfect margin should have all points of one class on one side of it and all those of the other class on the other side. The points not in the correct regions have displacement vectors ξ_i^* (see Figure 6). The algorithm optimizes the model by maximising M subject to a maximum value of the sum of all ξ_i^*.

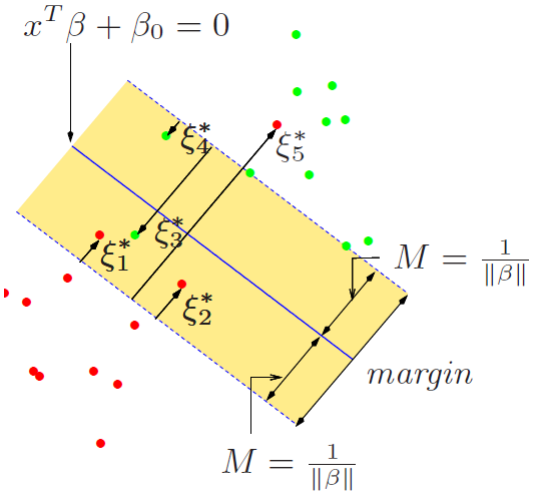

Figure 6: SVM margin band (dashed line), with ξ_i^* being the displacement vectors (in black) of each point from the correct region of the space. The model is the central blue line. Source: [23]

- Random Forest: An ensemble algorithm (which uses several individual algorithms) that compares a group of different decision trees to optimise the training set, through a prediction that averages the predictions of all trees, which are weighted individually, in the neighbourhood of a data point [23]. Random forests are less susceptible to overfitting than individual trees.

- AdaBoost: An ensemble algorithm which "boosts", or improves, the performance of a group of classifiers, some of which are little better than random guesses, known as "weak classifiers", by comparing the accuracy of each classifier to calculate a weight. The final predictive model is then a sum of the models of the individual classifiers multiplied by their individual weights. More accurate classifiers have a greater weight [23].

Additionally, the Gaussian-Hellenger Very Fast Decision Tree (GH-VFDT), as detailed by Lyon et al. [11], was used to confirm the results of Table 11 in the same paper, but was then discarded.

4.3.2 Unsupervised learning algorithms

The following unsupervised learning algorithms were used in this project but were not considered important to the final outcome. Therefore, they will not be discussed in detail:

- One-class SVM: Adaptation of SVM for anomaly detection - it only trains on one class. This is done by transforming the feature space using a kernel

function, then treating the origin of this space as the only member of the second class. Standard SVM procedure is then followed [26]

- Elliptic Envelope: Assumes the negatives are distributed in feature space as a multivariate Gaussian and fits the means and standard deviations to the data. Any outliers from this model are classified as anomalies/non-negatives.

- Principal Component Analysis (PCA): Similar in concept to the elliptic envelope. For a dataset of n features, a PCA classifier finds $q < n$ orthonormal axes such that the variance of the data when projected onto this lower-dimensional feature space, is maximal. These q features are referred to as the principal components. The q features are defined as the eigenvectors of the sample covariance matrix, which measures the data's variance from the sample mean over the original n-dimensional feature space, with the highest eigenvalues [7].

The main unsupervised learning algorithm used in this project, however, was the Isolation Forest (IF) [27]. This is an ensemble algorithm which takes the average result from t individual isolation trees (iTrees), in the same manner as a random forest. An example of which is shown in Figure 5, for each one of n data points being analysed. An iTree behaves in the same manner as a regular decision tree, by separating data points using decision boundaries, based upon the available features, at each node (except for the end nodes). The difference comes from the fact that unsupervised learning output is not restricted to two classes. An iTree instead continues running until a node has been created for each data point or a set limiting number of branching steps is reached, so that overall, all data points are in an "external" or "end" node (a node that no longer branches out). The number of recursive partitions required to separate a given data point, x, into its own node is known as the "path length", $h(x)$. The path length is recorded for each tree, and the results for all t trees are used to generate an isolation score, $s(x, n)$, for the forest for each point. This isolation score is defined as:

$$s(x, n) = 2^{-\frac{E(h(x))}{c(n)}} \tag{18}$$

where $E(h(x))$ is the expectation value of $h(x)$ over t results and:

$$c(n) = 2H(n1)(2(n1)/n) \tag{19}$$

is defined as the mean path length of unsuccessful searches in binary search trees [27] (such as the supervised learning trees mentioned earlier). The isolation score is normalised - if a point is isolated into an external node immediately ($E(h(x)) = 0$), $s(x) = 1$, while $s(x) \to 0$ if $E(h(x)) \to n - 1$, i.e. if x must be separated point-by-point from the other data or the height limit of the tree is reached. The advantage of using this classifier is that it can split the training data into subsamples containing ψ randomly-selected datapoints, and train on these subsamples instead of the entire dataset. This is useful for larger datasets with a class which can be accurately represented with a much smaller subset, such as the noise present in radio astronomy. Thus subsampling, if the subsample is of the correct size, saves time and requires less computing power, while sacrificing very little information for the model being optimised during the training stage.

5 Methodology and results

5.1 Pulsars

The algorithms detailed in Table 11 of Lyon et al. [11] were run on all three datasets to test the reproducibility of the results. The results were reproduced successfully. Then, with the exception of the GH-VFDT, those algorithms (or their ScikitLearn equivalents) were used as classifiers for the pulsar data, along with the AdaBoost and Random Forest ensemble algorithms. Feature selection was achieved using JMI maximisation (see Section 4.1). Two feature sets were compared:

- Lyon features: Detailed by Lyon et al.[11], this is a set of 8 features consisting of 4 statistical measures - the mean, standard deviation, skewness and kurtosis, which are the first, second, third and fourth statistical moments, respectively - applied to both the pulse profile and DM-SNR curve.
- Thornton features: Detailed by Thornton [15], this is the set of 22 features originally implemented for the Stuttgart Neural Network Simulator (SNNS), including the period and dispersion measure as well as how the pulse profile shape matches to various curves, such as Gaussians.

The Lyon feature was judged to be better overall via averaging the ranks of the JMI score for the individual features over the 3 datasets. It was then combined with the period, which was needed to separate the pulsars into MSPs and non-MSPs, and class values, which were designated as outputs, allowing for comparison of the cross-validation and test results with the real values.

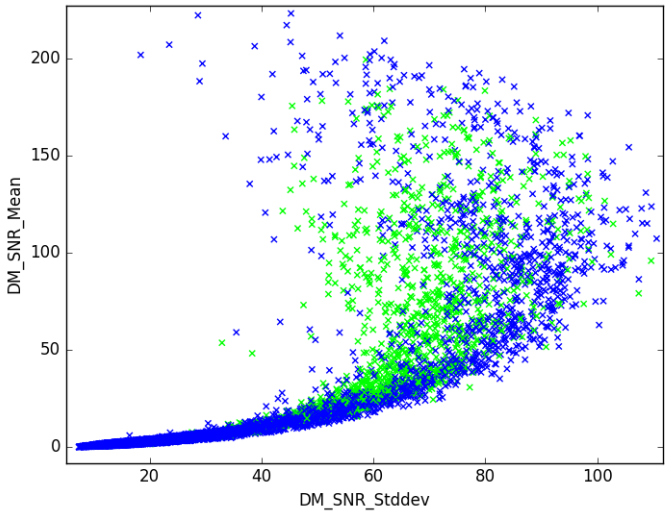

Figure 7: Raw HTRU2 data: pulsars (green), including MSPs, compared to noise (blue)

To check the suitability of the simulated pulsar data and look for relations between features, the data was manipulated to visualise the data in WEKA via plots such as Figures 7 and 8. When the simulated data was declared viable for classification, a comment was added to classified MSPs, allowing them to be distinguished from

other real pulsars as well as the simulated MSPs. Using this comment, the real MSPs were removed from the HTRU2 data and placed into a separate ARFF file, which would be used as the testing dataset during classification, with the remaining data being used as the training set.

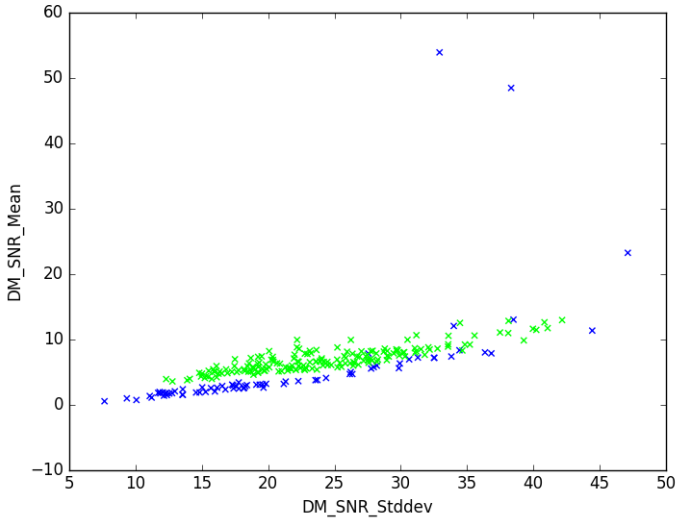

Figure 8: Unrealistic simulated MSP data (green) compared to HTRU2 MSPs (blue)

When creating the simulated MSP data, the way in which the data was processed was very important with regard to how realistic it appeared in the feature space compared with real MSPs. Any datasets which had not been generated in the exact same way as the original HTRU2 data were easily distinguishable from the real data, particularly in their DM-SNR curve features. This is shown in Figure 8, where most simulated MSPs can clearly be separated from real MSPs by a line bisecting the lines followed by each group. These differences adversely affect the model being used to classify real MSPs. The final simulated dataset, consisting of 744 simulated MSPs, did not deviate from the feature trends shown by the real pulsars upon visual inspection, and, within those trends, closely aligns with the regions occupied mostly by the real MSPs (see Figure 18 in the Appendix). Consequently, they were assessed to be realistic in the Lyon feature space.

For classifying the real MSPs, the training dataset, containing everything except the real MSPs, underwent cross-validation, with the noise being class-0 and all remaining pulsars (including simulated examples, where relevant) being class-1, with the number of folds set at $k = 5$. The real MSPs were then used as the testing set, using the model created by the cross-validation of the training data. Since the testing set was made up purely of class-1 objects (the real MSPs), the accuracy and recall of algorithms applied to that set are exactly the same by definition. This was done as part of the overall emphasis on the pulsars in the data.

A test of significance was carried out on the HTRU2 results generated before and after added the simulated MSPs to the training data, in the form of a student's t-test. A student's t-test was chosen as it controls for sample sizes, which is important, given

Classifier	R **for HTRU2 data**	R **for HTRU2 + sim.**	R **for corrected HTRU2**
CART_tree	0.301 ± 0.042	0.594 ± 0.023	0.698 ± 0.026
MLP	0.383 ± 0.027	0.603 ± 0.012	0.695 ± 0.020
NaiveBayes	0.380 ± 0.000	0.465 ± 0.000	0.561 ± 0.000
SVM	0.487 ± 0.013	0.682 ± 0.008	0.786 ± 0.008
Random Forest	0.293 ± 0.058	0.563 ± 0.039	0.698 ± 0.008
AdaBoost	0.358 ± 0.016	0.558 ± 0.008	0.660 ± 0.010

Table 1: Recall on HTRU2 MSPs (class 1) using the HTRU2 noise (class 0) and non-MSPs (class 1) as training data

Classifier	R **for HTRU1 data**	R **for HTRU1 + simulated data**
CART_tree	0.819 ± 0.040	0.859 ± 0.025
MLP	0.816 ± 0.012	0.854 ± 0.034
NaiveBayes	0.811 ± 0.000	0.824 ± 0.000
SVM	0.981 ± 0.007	0.986 ± 0.000
Random Forest	0.822 ± 0.015	0.846 ± 0.039
AdaBoost	0.789 ± 0.028	0.792 ± 0.008

Table 2: Recall on HTRU1 MSPs (class 1) using the HTRU1 noise (class 0) and non-MSPs (class 1) as training data

the differences in pulsars numbers before and after adding the simulated MSPs. This generates a result, known as the p-value, using the two results and their respective standard deviations. The threshold p-value was chosen as $p_{thres} = 0.05$, which is a typical value and indicates that rejection of the null hypothesis is acceptable to a maximum error of 5% , with the p-value representing the actual calculated error. Hence, if the significance test generates a p-value less than p_{thres}, the null hypothesis is rejected and the difference between results is significant [23]. Having used HTRU2 as a dataset to optimize the simulated data, the simulated data was added to HTRU1 and the algorithms were then applied to the resulting dataset. After adding simulated MSPs to the training set, the recall on HTRU2 increased significantly compared with using just raw observational data (see Table 1), while the recall on HTRU1 did not change significantly (see Table 2).

A program was written that uses the combined HTRU2 and simulated training dataset to determine the effect of varying the MSP cutoff period threshold, P_{MSP}, on the MSP recall using the new definition. This was carried out by entering a quantity and range of P_{MSP} values as arguments, then plotting the mean recalls against P_{MSP}, with error bars, for each individual classifier. The variation in definition was applied to real HTRU2 pulsars only. This was carried out for both the newly-defined MSPs and all pulsars, as a reference. The results are shown in Figure 9 and show that the recall on real MSPs, as defined by the threshold, is not significantly affected by the exact position of the threshold.
A final set of programs was written such that circa 90% of noise below P_{MSP} was taken out of the training set. This did not significantly affect the classification results on the resulting training dataset nor on those for the MSP testing dataset.

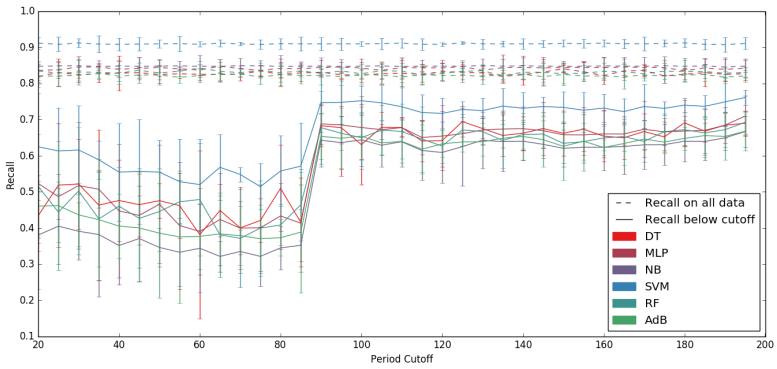

Figure 9: Recall on HTRU2 MSPs for variation in the MSP cutoff period

5.2 FRBs

During feature selection, the most useful features for distinguishing FRBs were the SNR, the boxcar width and the DM. This is not surprising, as these are the defining features of FRBs. The other features examined the number of nearest-neighbour data points for a given data point for different time limits on the definition of nearest-neighbours. These intervals were on the order of a few seconds of observation time. The reasoning behind this was that, because of the transient nature of FRBs, they may be more likely to be isolated points in feature space, while terrestrial noise tends to arrive in groups of detections at similar times. However, their JMI ranks varied significantly between datasets and individual runs, due to their JMI values being very small and subject to random fluctuations. Therefore, only the SNR, DM and boxcar width were taken forward as useful features.

The classifier evaluation programmes used with the pulsar datasets were updated and added to, in order to use the new format of data (.dat files, as opposed to the .arff format previously employed with pulsars) and the unsupervised algorithms. The results were then displayed visually as ROC curves (one for each classifier), as shown in Figure 10, with a corresponding value of the AUC also calculated for each curve (see Table 3). The Isolation Forest classifier, which is the best performing unsupervised classifier, is plotted in both panels for comparison with the supervised (right-hand panel) and other unsupervised (left-hand panel) classifiers.
Having established Isolation Forest as the most promising unsupervised classifier, a plot of precision against recall was created (see Figure 11). Isolation Forest is plotted twice, the second time after cutting out all data points with DM < 500 cm^{-3}pc, alongside Random Forest, AdaBoost and NaiveBayes. After applying the cutoff, the performance of Isolation Forest improves such that it begins to outperform NaiveBayes at higher recall values in terms of precision, although it remains inferior to the other supervised classifiers.

To visualise the behaviour of the classifiers and their subsequent decision boundaries, contour plots of the data in the SNR vs. DM plane were created, with each axis scaled such that the data have zero mean and unit variance for each feature. The results are plotted in Figure 15. The lighter regions represent areas of (2D)

Classifier	AUC value for FRB data
CART_tree	0.807 ± 0.078
MLP	0.999 ± 0.001
NaiveBayes	0.990 ± 0.008
SVM	0.982 ± 0.035
Random Forest	0.928 ± 0.059
AdaBoost	0.996 ± 0.009
Isolation Forest	0.990 ± 0.009
PCA	0.961 ± 0.042
One-class SVM	0.958 ± 0.085
Elliptic Envelope	0.057 ± 0.003

Table 3: AUC values for FRB dataset.

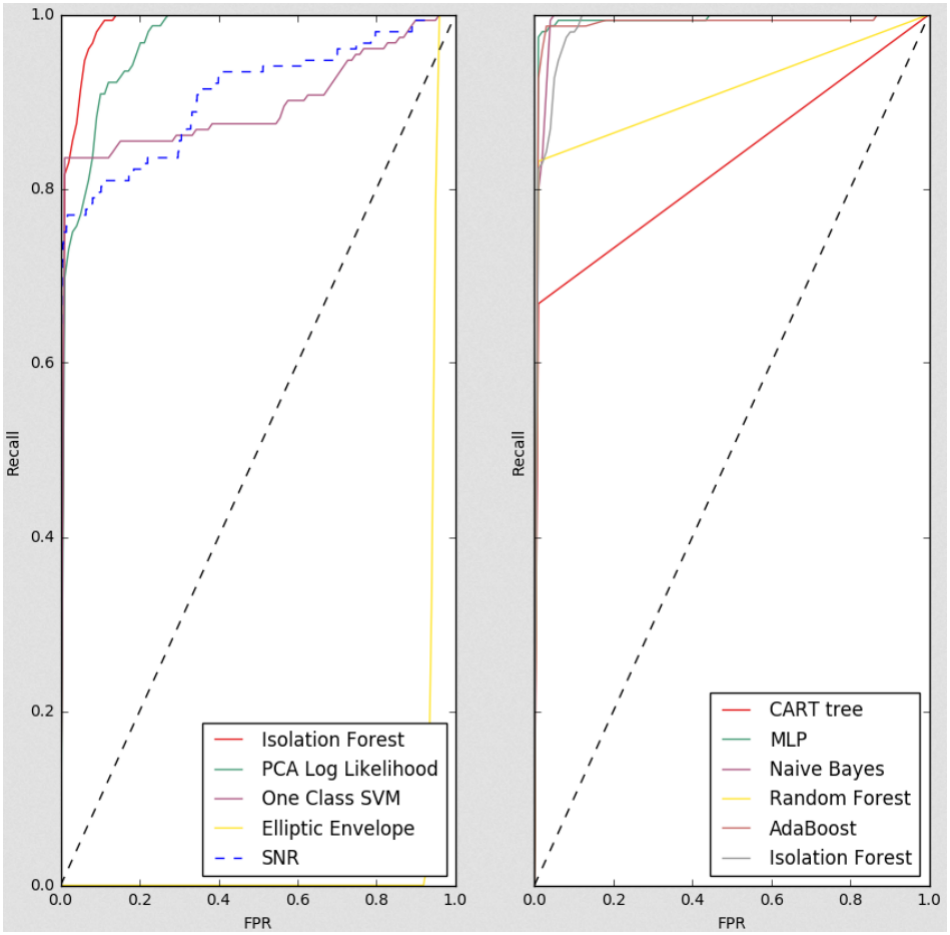

Figure 10: ROC curve for FRB data of supervised (R) and unsupervised (L) classifiers, with Isolation Forest in both figures. The SNR has been included as an additional reference for the suitability of classifiers

feature space in which anomalous (non-noise) data is determined to be prevalent by that particular classifier. Of the unsupervised classifiers only Isolation Forest and one-class SVM are able to separate FRBs and noise adequately, while the only supervised classifier to do so without overfitting on FRBs is the Gaussian NaiveBayes. The other supervised classifiers (which require both classes in the training data) ef-

fectively "learn" that the (simulated) FRBs have a small DM range compared with the noise data. Some classifiers also make use of the fact that the FRBs also have a lower limit on their SNRs relative to the noise.

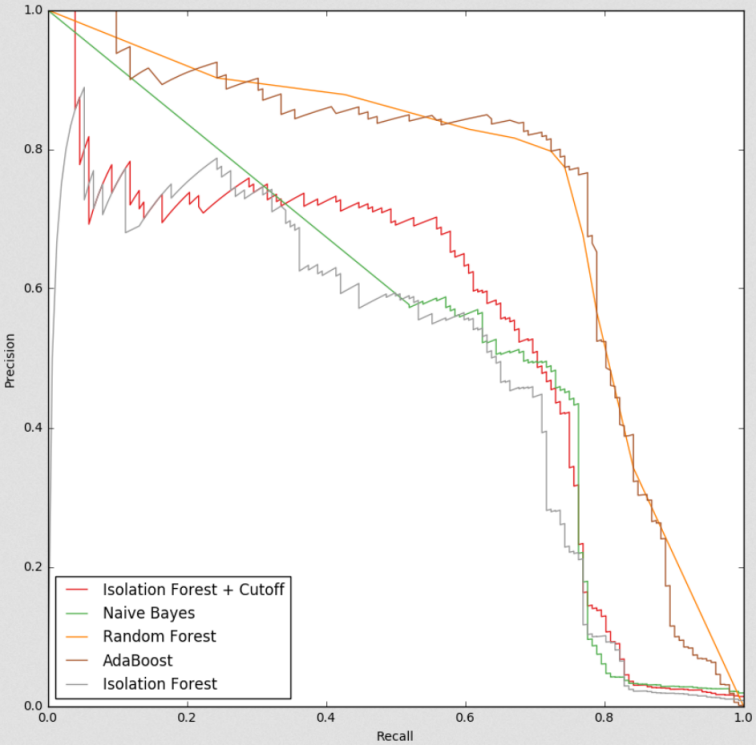

Figure 11: FRB data precision-recall plot of Isolation Forest (with and without cutting out all points with DM < 500 cm^{-3}pc, both ensemble supervised classifiers and NaiveBayes

6 Discussion of classification results

6.1 Pulsar results

The Lyon feature set is better for classifying pulsars by machine learning in this project partly because there are fewer features than is the case for the Lyon features, which makes the processing less complicated and quicker (i.e., it is less prone to the so-called "curse of dimensionality"). Secondly, they are generic statistics-based features which directly relate to existing classification techniques, as they are simply the four lowest statistical moments of two of the graphics already used by human researchers to classify objects (see Figure 1). By contrast, some of the Thornton features are speculative, such as χ^2 results of fitting the pulse profile to various models, or are insensitive to some details, such as the feature using the area under the pulse profile, which can take the same value for a very narrow, tall profile or a broad, short profile, and which therefore requires additional features to extract the details, which is more inefficient. Thirdly, the variety of pulse profiles shown by Pilia et al. [5] indicates that speculative fittings may be even less efficient as features, since most will be redundant for any given pulsar and the pulsar may not conform adequately to any of them.

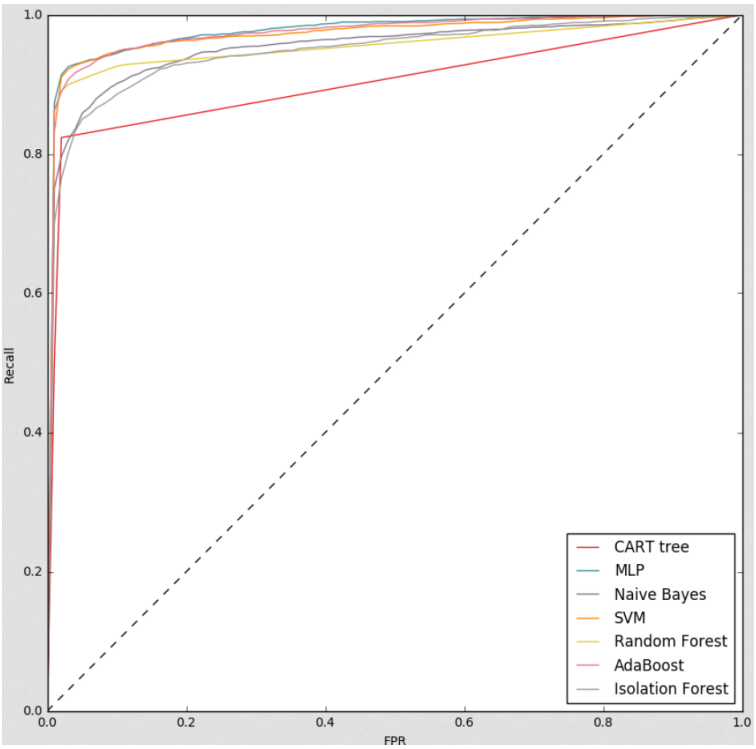

Figure 12: ROC curve of Isolation Forest together with the supervised algorithms, applied to HTRU2

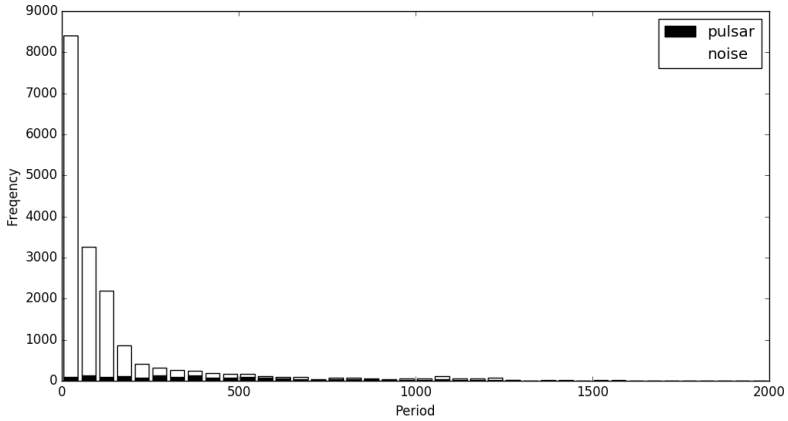

Figure 13: Histogram of HTRU2 noise (white) and pulsar (black) data, showing class imbalance in favour of noise data, particularly at low periods

The addition of simulated data clearly and significantly improves the performance of the classification algorithms on the HTRU2 data. This is not only due to the realistic appearance of the data in the feature space, but also to the impact of adding a large number (744) of simulated MSPs to the real MSP population (71), as it counters the dominance of non-millisecond pulsars within the dataset of classified pulsars, as well as noise within the full dataset (see Figure 13).

The performance of the NaiveBayes classifier was the poorest of the classifiers overall and the standard deviation effectively zero, which is indicative of an underlying problem. The NaiveBayes classifier assumes the features it uses to be independent of one another, which is clearly contradicted by the aforementioned trends in the data.

The performance of the SVM classifier was the best overall, which is likely due to its main strength of classifying non-separable data (see Section 2.4), of which the datasets in this project are examples (Figure 7 in the Appendix is one example of the substantial mixing of data points from both classes in HTRU2).

HTRU1 was not improved by any level of significance by the addition of the final simulated data. However, the classifiers performed very well without adding the simulated data, leaving little room for improvement. Also, had the simulated data been completely unrealistic for HTRU1, its addition would have been expected to cause the results to deteriorate.

It must be noted, however, that HTRU1 and HTRU2 were processed using different pipelines methods and codes, which may explain how two sets of data from the same survey can produce such different performance levels from the same classifier algorithms and for every one of those algorithms. HTRU1 was processed using the PEASOUP pipeline, detailed by Morello et al. [12], while HTRU2 was processed using PDMP by Thornton [15]. The surprisingly high significance of how exactly a dataset was processed upon the data itself had already been witnessed when processing the simulated datasets (see Section 6).

The effect of varying the period threshold for defining MSPs is not significant, which is surprising, since MSPs are very clearly defined from an astrophysical perspective (see Introduction). The increase in recall in the 80-100 ms cutoff period interval is not significant for each classifier individually, but its transcendence of every algorithm in Figure 9 suggests it is significant overall. However, the simultaneous increase in the MSP-to-noise candidate ratio (see Figure 14 in Appendix) suggests than the significance of the effect is due to the sudden appearance of pulsar data, as the period increases, which allows the classifiers to better separate the classes overall. This could also be due to localised lessening of noise dominance (class imbalance) as more pulsars are added to the MSP dataset.

The effect of noise reduction was found to be statistically insignificant. This suggests that correctly classifying pulsars will depend more on the pulsar data rather than the noise data. Noise reduction is not particularly helpful in the context of classifying data in real time, as it requires the data to be classified beforehand. This, in turn requires data to be stored and then analysed closely. This process is not perfect, as shown in Table 1, where, for some classifiers, the misclassification of noise data as MSPs causes the classifier performance to deteriorate significantly.

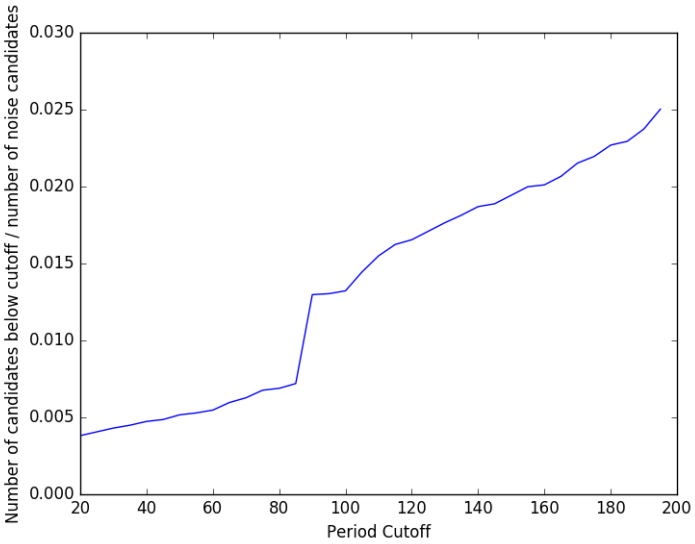

Figure 14: Ratio of MSPs to noise candidates as a function of P_{MSP}

6.2 FRB results

As expected, the DM, boxcar width and SNR give the highest JMI scores during testing for feature selection. However, the other features, relating to the tendency of data points to cluster together in terms of the time of observation, produced small JMI scores which were subject to fluctuations in both their absolute values and rankings. This is significant, as FRBs are single-detection events that are anomalous compared to known sources of noise and hence should ideally occupy a different region in feature space. It is possible, therefore, that the feature space made available by the processing pipeline is insufficient for reliable detection of FRBs, as no features, aside from those used to originally define FRBs, were found which were of any use in classifying the data.

As shown in Table 3, the AUC values of all the classifiers (except Elliptic Envelope) are within a 3σ threshold of each other, indicating that, for this dataset, supervised and unsupervised classifiers, in general, are equally useful. However, it should be noted that, when using a classifier algorithm, only a single point on the relevant curve is being used, as only a single threshold is employed throughout the classification process. It is therefore up to the user to compromise between a high recall and a low FPR. For a dataset such as the FRB dataset used in this project, the user should focus primarily on keeping the FPR low. This is because the accuracy of non-negatives (FRBs in this case) is effectively unknown, due to the paucity of real examples. On the other hand, the negatives (noise) are all real observation data and exist in a number large enough to be a representative sample.

One of the major drawbacks of supervised learning is that it requires representative samples of all relevant classes (noise and FRBs). This is not a problem for the noise - there is an abundance of accurate and representative noise samples to use for training. Since this is clearly not the case for FRBs, the use of supervised classifiers cannot be justified here. In Figure 15, with the exception of the Gaussian

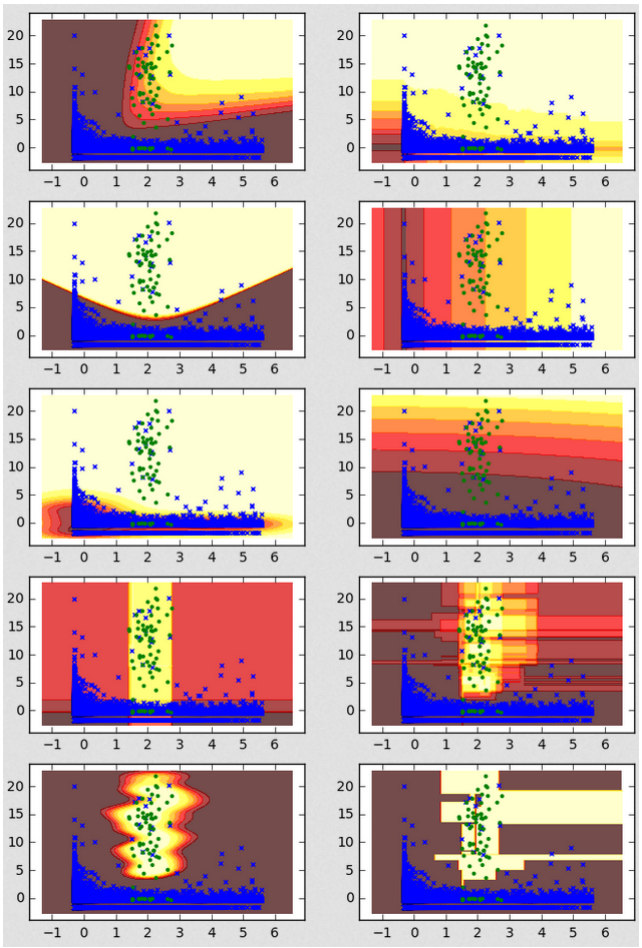

Figure 15: SNR-DM contour maps of FRB data for the following classifiers (clockwise from top left): MLP, Isolation Forest (with subsamples of 8000 data points and 100 trees), Elliptic Envelope, PCA, Random Forest, CART tree, SVM, AdaBoost, one-class SVM, Gaussian NaiveBayes.

NaiveBayes, the supervised classifiers utilise the restricted (and arbitrary) range of DM values (and sometimes SNR values) in the simulated FRB population to define their decision boundaries. While this is technically desirable for a supervised classifier, it is, in reality, a sign of their unsuitability in the case of FRBs. This is because there is no indication of the simulated FRBs being representative of all possible FRB detections and there are no reliable FRB datasets, simulated or otherwise, upon which to train classifiers. This leads to a significant risk of a supervised classifier misclassifying an FRB as noise merely because the classifier has effectively overfitted on a non-representative FRB dataset. Therefore an unsupervised classifier, which only requires noise data to produce a model, should be used for screening FRB candidates.

Isolation Forest is the best performing unsupervised classifier overall, as shown by the left-hand ROC curve in Figure 10. The others perform almost as well (within the margin of error relative to Isolation Forest), with the exception of the Elliptic Envelope algorithm, which performs far worse than even a random guess. Regarding the precision-recall curve in Figure 11, it remains competitive with NaiveBayes, the

only supervised algorithm type (if Gaussian NaiveBayes is included) which does not base its decision boundaries on learning the DM limits of the simulated FRB data, as shown in Figure 15. By comparing the results from Figures 15 and 11 for the classifiers which appear in both, it is not surprising that the precision is worse for Isolation Forest and NaiveBayes, as the white areas in their contour plots include many more noise data points than those of AdaBoost and Random Forest.

Isolation Forest possesses a range of other properties which are desirable in the context of transient radio searches. Unlike some of the other unsupervised classifiers, it does not assume a particular distribution of noise data in feature space. It is computationally less complex (and therefore less expensive, and quicker, to run) than other classifiers. When training its optimisation function its complexity is of order $O(t\psi log(\psi))$, i.e. completely independent of the number of data points, in the dataset, n. In fact, the classifier often reaches acceptable performance levels for relatively small values of t and ψ [27]. When testing the model on the dataset, the complexity is of order $O(tnlog(\psi))$. Since t and ψ are determined by the user and initialised to fixed values before the algorithm is run, they can be regarded as constants once the algorithm starts running, so the complexity is effectively of order $O(n)$. This is less complex than for PCA ($O(n^2)$) and both types of SVM classifier ($O(n^{2-3})$) [22].

Isolation Forests, being an application of unsupervised learning, do not require their input data to be classified beforehand. During processing of observational data, this has the double advantage of saving time (as the processing pipeline no longer needs to classify raw data) and avoiding classification errors (such as those encountered in HTRU2 with the mislabelled "pulsars"). Isolation Forests can also be adapted for use in data streams (the continual addition of data into datasets in real-time).

The combination of these properties makes Isolation Forests a strong candidate for processing data for upcoming projects such as the SKA, in which data streaming will be common, due to its exceptionally high rate of data production (see Introduction). It would be particularly useful in transient searches as it is anomaly-based, but it also performs well for pulsar data, such as HTRU2 (see Figure 12), and it therefore could have applications beyond transient searches.

7 Conclusion

Using simulated pulsar data, if it demonstrates behaviour in feature space similar to that of observational pulsar data, does significantly improve classification using supervised-learning intelligent algorithms. For pulsars, supervised learning was highly useful due to the highly non-separable nature of the data (see Figure 7) and to the constraints on pulsars in the feature space. This allowed for the creation and use of viable simulated pulsar data, which significantly improves performance of the supervised classifiers on pulsar data.

For fast radio bursts, there is no known representative dataset of examples. FRBs by nature are anomalous readings which are otherwise unknown. Therefore they

should be searched for using unsupervised classifiers via anomaly scores. The Isolation Forest classifier was found to be the best such example.

References

[1] P. Scholz, L. Spitler, J. Hessels, S. Chatterjee, J. Cordes, V. Kaspi, R. Wharton, C. Bassa, S. Bogdanov, F. Camilo, *et al.*, "The repeating fast radio burst frb 121102: Multi-wavelength observations and additional bursts," *arXiv preprint arXiv:1603.08880*, 2016.

[2] E. Petroff, E. Keane, E. Barr, J. Reynolds, J. Sarkissian, P. Edwards, J. Stevens, C. Brem, A. Jameson, S. Burke-Spolaor, *et al.*, "Identifying the source of perytons at the parkes radio telescope," *Monthly Notices of the Royal Astronomical Society*, vol. 451, no. 4, pp. 3933–3940, 2015.

[3] M. Keith, A. Jameson, W. Van Straten, M. Bailes, S. Johnston, M. Kramer, A. Possenti, S. Bates, N. Bhat, M. Burgay, *et al.*, "The high time resolution universe pulsar survey–i. system configuration and initial discoveries," *Monthly Notices of the Royal Astronomical Society*, vol. 409, no. 2, pp. 619–627, 2010.

[4] C. Ng, "Conducting the deepest all-sky pulsar survey ever: the all-sky high time resolution universe survey," *Proceedings of the International Astronomical Union*, vol. 8, no. S291, pp. 53–56, 2012.

[5] M. Pilia, J. Hessels, B. Stappers, V. Kondratiev, M. Kramer, J. Van Leeuwen, P. Weltevrede, A. Lyne, K. Zagkouris, T. Hassall, *et al.*, "Wide-band, low-frequency pulse profiles of 100 radio pulsars with lofar," *Astronomy & Astrophysics*, vol. 586, p. A92, 2016.

[6] W. Zhu, A. Berndsen, E. Madsen, M. Tan, I. Stairs, A. Brazier, P. Lazarus, R. Lynch, P. Scholz, K. Stovall, *et al.*, "Searching for pulsars using image pattern recognition," *The Astrophysical Journal*, vol. 781, no. 2, p. 117, 2014.

[7] M. E. Tipping and C. M. Bishop, "Mixtures of probabilistic principal component analyzers," *Neural computation*, vol. 11, no. 2, pp. 443–482, 1999.

[8] B. Bhattacharyya, S. Cooper, M. Malenta, J. Roy, J. Chengalur, M. Keith, S. Kudale, M. McLaughlin, S. M. Ransom, P. S. Ray, *et al.*, "The gmrt high resolution southern sky survey for pulsars and transients. i. survey description and initial discoveries," *The Astrophysical Journal*, vol. 817, no. 2, p. 130, 2016.

[9] A. Ahuja, Y. Gupta, D. Mitra, and A. Kembhavi, "Tracking pulsar dispersion measures using the giant metrewave radio telescope," *Monthly Notices of the Royal Astronomical Society*, vol. 357, no. 3, pp. 1013–1021, 2005.

[10] D. Lorimer, M. Bailes, M. McLaughlin, D. Narkevic, and F. Crawford, "A bright millisecond radio burst of extragalactic origin," *Science*, vol. 318, no. 5851, pp. 777–780, 2007.

[11] R. Lyon, B. Stappers, S. Cooper, J. Brooke, and J. Knowles, "Fifty years of pulsar candidate selection: From simple filters to a new principled real-time classification approach," *Monthly Notices of the Royal Astronomical Society*, vol. 459, no. 1, pp. 1104–1123, 2016.

[12] V. Morello, E. Barr, M. Bailes, C. Flynn, E. Keane, and W. van Straten, "Spinn: a straightforward machine learning solution to the pulsar candidate selection problem," *Monthly Notices of the Royal Astronomical Society*, vol. 443, no. 2, pp. 1651–1662, 2014.

[13] D. R. Lorimer, "Binary and millisecond pulsars," *Living Reviews in Relativity*, vol. 11, no. 8, p. 21, 2008.

[14] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *ACM SIGKDD explorations newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

[15] D. Thornton, "The high time resolution radio sky," 2013.

[16] W. van Straten and M. Bailes, "Dspsr: digital signal processing software for pulsar astronomy," *Publications of the Astronomical Society of Australia*, vol. 28, no. 01, pp. 1–14, 2011.

[17] G. Hobbs, R. Edwards, and R. Manchester, "Tempo2, a new pulsar-timing package–i. an overview," *Monthly Notices of the Royal Astronomical Society*, vol. 369, no. 2, pp. 655–672, 2006.

[18] S. Nogueira and G. Brown, "Measuring the stability of feature selection," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 442–457, Springer, 2016.

[19] R. Kohavi *et al.*, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Ijcai*, vol. 14, pp. 1137–1145, 1995.

[20] M. Bennasar, Y. Hicks, and R. Setchi, "Feature selection using joint mutual information maximisation," *Expert Systems with Applications*, vol. 42, no. 22, pp. 8520–8532, 2015.

[21] M. Kubat, R. C. Holte, and S. Matwin, "Machine learning for the detection of oil spills in satellite radar images," *Machine learning*, vol. 30, no. 2-3, pp. 195–215, 1998.

[22] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[23] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*, vol. 1. Springer series in statistics Springer, Berlin, 2001.

[24] P. J. Lisboa and M. J. Taylor, *Techniques and applications of neural networks*. Viking Penguin, 1993.

[25] G. H. John and P. Langley, "Estimating continuous distributions in bayesian classifiers," in *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, pp. 338–345, Morgan Kaufmann Publishers Inc., 1995.

[26] L. M. Manevitz and M. Yousef, "One-class svms for document classification," *Journal of Machine Learning Research*, vol. 2, no. Dec, pp. 139–154, 2001.

[27] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pp. 413–422, IEEE, 2008.

8 Appendix

8.1 Source code repository

https://github.com/lsgos/MPhysPulsars
Contains source code, datasets, simulated data, results in .png, .ods and .txt formats, Jupyter notebooks (.ipynb) and a copy of PulsarFeatureLab. All new codes for this computer-based project were written in Python 2.7.12.

8.2 Risk assessment

1. Hazard identification:

(i) High computer usage and the subsequent risk of sight problems due to over-exposure to screens.

(ii) Electrocution

2. People at risk:

(i) MPhys participants

(ii) MPhys participants

3. Risk evaluation:

(i) The risk is adequately controlled due to industry standards and user guidance regarding computer screen usage.

(ii) The risk is adequately controlled due to grounding and insulation of wires.

4. Actions to be taken:

(i) Take regular breaks away from computer screens.

(ii) Follow standard guidelines for use of electrical equipment.

Last reviewed: 11th May 2017
Carried out by: Alex Lisboa-Wright, Lewis Smith.
Supervisor: Michael Keith.

8.3 Additional figures

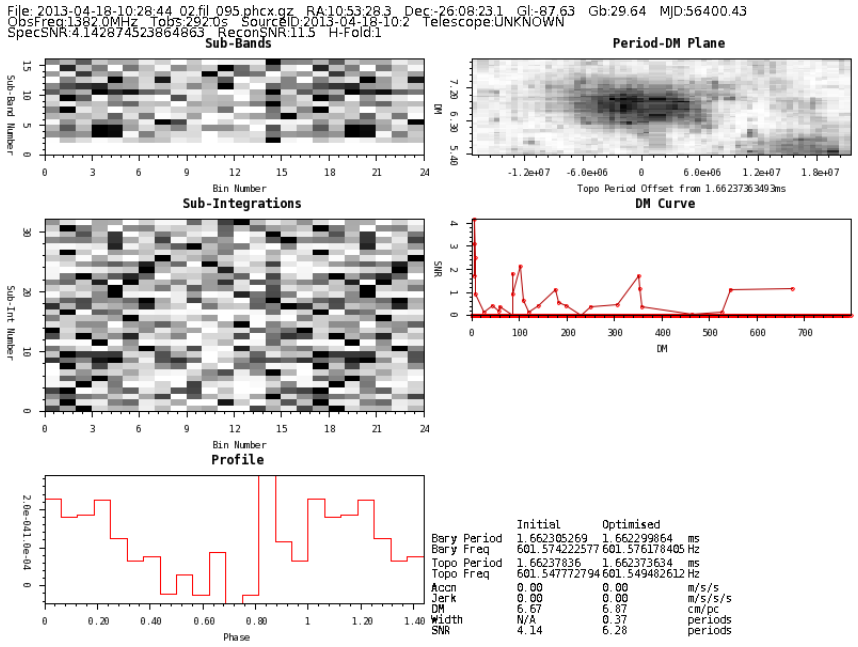

Figure 16: psrsoft image output for a non-pulsar (noise) data file

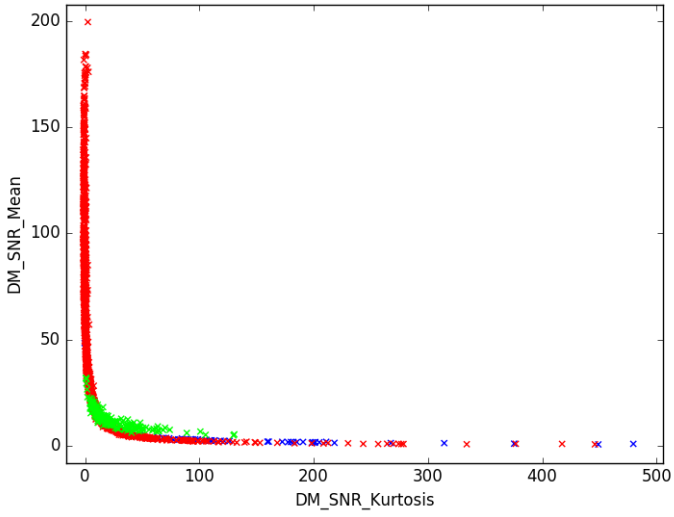

Figure 17: Unrealistic simulated MSP data (green) compared to HTRU2 MSPs (blue) and non-MSPs (red)

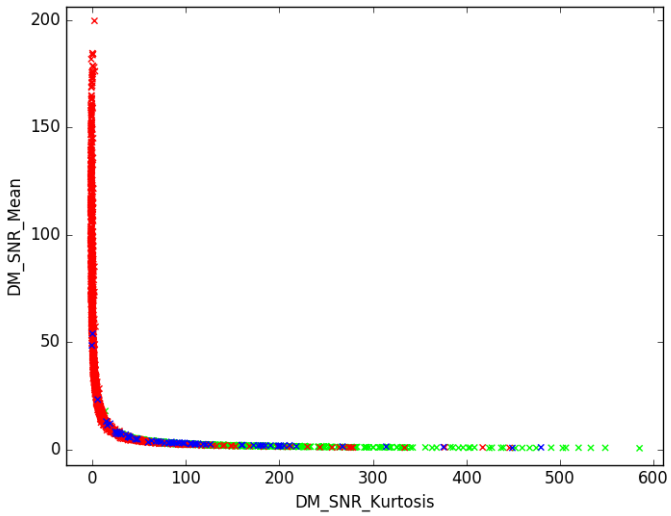

Figure 18: Final (realistic) simulated MSP data (green) compared to HTRU2 MSPs (blue) and non-MSPs (red)