Application of machine learning methods to identify and categorize radio pulsar signal candidates

Serena Debesai (sdebesai), Carmen Gutierrez (cguti), and Nazli Ugur Koyluoglu (nazli)

June 12, 2020

1 Abstract

A pulsar star is a high mass star that collapses into a neutron star and spins increasingly rapidly, emitting intense electromagnetic radiation. The beams of energy emitted by the pulsar star are detected as radio waves by our telescopes. Their unique integrated electromagnetic radiation profile and strongly periodic received signal allows for classification and flagging of pulsars from a set of noise and Radio Frequency Interference (RFI) signals. However, radio pulsar surveys produce many more pulsar candidate signals than can be inspected by human experts by hand. Here, we investigate the application of various machine learning methods to the problem of automatic pulsar identification.

2 Introduction

A pulsar is a high-mass star that collapses into a dense magnetized neutron star and spins increasingly rapidly, emitting intense electromagnetic radiation from its magnetic poles. As the star and its poles rotate, the beams of energy radiated from the poles pass over us, resulting in a strongly periodic radio signal being received by radio telescopes on Earth. This is sometimes likened to a "lighthouse" effect, because the radiation can only be observed when a beam is pointed towards us, much like the way a lighthouse can be seen only when the light is pointed in the direction of an observer. The beams of energy emitted by the pulsar star are detected as extremely regular pulses (in the range between milliseconds and seconds) in radio signal (hence the name "pulsar") by our telescopes. Pulsars are of great scientific interest because of their highly regular signal; once a pulsar has been identified, deviations from its normal period can be used as evidence of gravity waves, dark matter, and other cosmic phenomena.

Though traditional pulsar search algorithms can identify periodic signals to be inspected by human experts, they are not adept at telling the difference between astronomical sources of periodic signal, like pulsars, and Radio Frequency Interference, or RFI, that is produced by such things as FM radio. Additionally, radio pulsar surveys produce many more candidate signals than can be reasonably inspected by hand, and astronomers often have to turn to sources of voluntary help, such as high school students who volunteer with the Pulsar Search Collaboratory, to help flag RFI and likely pulsars. This is a very labor-intensive process, and is ideally suited to the application of machine learning methods.

3 Related work

One of the first papers to explore the application of machine learning to pulsar classification was Eatough et al. (2010), who used a training data set of 259 known pulsars and 1624 non-pulsars, each with between 8 and 12 features (they note that the inclusion of the 4 extra features had little effect on the success rate). They implemented an ANN, which was groundbreaking but which when tested on a repository of 2.5 million candidates only recovered 92% of the known pulsars in the set. This recovery rate, Ford (2017) notes, was not acceptable enough to entrust the algorithm with classification without human supervision.

Bates et al. also attempted a similar classification, using 22 features; the use of more features did not seem to correlate with more success, and researchers have speculated that Bates and Eatough were not rigorous or selective enough in their choice of features. Morello et al. (2014) employed more advanced techniques, achieving improved accuracy even with a smaller data set than Eatough and Bates. For example, Morello et al. attempted to upsample the data to correct for the severe imbalance between non-pulsars and pulsars, but Ford notes that they may have also upsampled the validation data, resulting in an inflated accuracy.

Lyon et al. (2016), whose data set we used, has done work relating to optimal feature selection, resulting in the 8 simple statistical features used as features in this paper.

All of the papers mentioned above used statistics distilled from the pulse plots, not the data of the pulse plots themselves. This loss of information, which also happens in the data we used, could be detrimental to accuracy. Zhu et al. (2014) attempted to implement an algorithm that learned directly from the pulse plots, to some success. Note that we are using the simple statistics used by Lyon; however, we will be implementing more advanced and numerous machine learning techniques in an attempt to improve the accuracy of algorithms trained on this data set.

Ford(2017) made use of Support Vector Machines (SVMs) to improve classification, and did much to address the problems that arose in previous papers. Ford suggested Random Minority Oversampling (ROS) and Random Majority Undersampling (RUS) as well as Synthetic Minority Oversampling Technique (SMOTE) developed by Machine Learning researchers as possible solutions to the imbalanced dataset; his expertise in machine learning methods were a welcome addition to the current body of research relating to automated pulsar identification. However, Ford primarily focused on using SVMs, while we will be expanding on his work and applying newer and more varied methods.

4 Dataset

The dataset we used during our investigation was provided courtesy of Dr. Robert Lyon of the University of Manchester School of Physics and Astronomy in the United Kingdom and accessed via the University of California, Irvine Machine Learning repository (Lyon). Data collection and primary data analysis was conducted by the High Time Resolution Universe Survey (HTRUS).

The data contains a total of 17,898 examples, 16,259 of which are spurious examples caused by Radio Frequency Interference (RFI) or noise, and 1,639 real pulsar examples identified by human annotators.

4.1 Features

Each example is described by 8 continuous variables that describe a longitude-resolved version of the signal that has been averaged in both time and frequency, and 1 class variable. For real pulsars, the class label is 1; otherwise, the class label is 0. The 8 features were obtained by the HTRUS using classical pulsar candidate analysis, which involves creating and analyzing the (folded) integrated pulse profile and the Dispersion Measure/Signal-to-Noise Ratio (DM/SNR) curve. These 8 features are:

1. Mean of the integrated profile
2. Standard deviation of the integrated profile
3. Excess kurtosis of the integrated profile
4. Skewness of the integrated profile
5. Mean of the DM-SNR curve
6. Standard deviation of the DM-SNR curve
7. Excess kurtosis of the DM-SNR curve
8. Skewness of the DM-SNR curve

This yields an array-like csv data-set of dimension 17,898 x 9, where the first column corresponds to the label (0 or 1) and the 8 remaining columns correspond to the features.

5 Supervised Learning Methods

First, we implemented a Gaussian Discriminant Analysis algorithm, which acted as our baseline for the supervised learning methods. This algorithm was provided with the entire dataset. We then applied Random Forest, first using Python's scikit-learn library and then using a random forest algorithm implemented from scratch.

5.1 Gaussian Discriminant Analysis

Gaussian Discriminant Analysis (GDA) models $p(x|y)$, the probability of a feature value given a value for the outcome as a multivariate normal distribution. It then finds the vector means of the two classes, and chooses a linear decision boundary to separate them. This model is described as:

$$y \sim Bernoulli(\phi)$$
$$x|y = 0 \sim (\mu_0, \Sigma)$$
$$x|y = 1 \sim (\mu_1, \Sigma)$$

The optimal values of these parameters $(\phi, \mu_0, \mu_1, \Sigma)$, are as follows:

$$\begin{aligned}
\phi &= \frac{1}{n}\sum_{i=1}^{n} 1\{y^{(i)=1}\} \\
\mu_0 &= \frac{\sum_{i=1}^{n} 1\{y^{(i)} = 0\}x^{(i)}}{\sum_{i=1}^{n} 1\{y^{(i)} = 0\}} \\
\mu_1 &= \frac{\sum_{i=1}^{n} 1\{y^{(i)} = 1\}x^{(i)}}{\sum_{i=1}^{n} 1\{y^{(i)} = 1\}} \\
\Sigma &= \frac{1}{n}\sum_{i=1}^{n}(x^{(i)} - \mu_{y^{(i)}})(x^{(i)} - \mu_{y^{(i)}})^T
\end{aligned}$$

We chose to use this as our baseline because it is suited to classifying our data into two classes as pulsar and non-pulsar. We chose this over a Naive Bayes model because our features are continuous, and not discrete, values. We also chose GDA as our baseline over discriminative learning algorithms such as logistic regression because it is asymptotically efficient on data that is indeed normally distributed, and also because it makes stronger modelling assumptions.

5.2 Random Forest

Random Forest is a classification algorithm that uses a large number of uncorrelated decision trees (which will be discussed shortly) to come up with a consensus prediction on each example. The use of *uncorrelated* decision trees is very important because the variance between the models of each decision tree reduces the effect of individual errors made by each model, thereby reducing the overall prediction error.

A decision tree is a tree-like series of nodes, or questions we might ask of the data, that divides the data into binary groups at each node. Each decision tree is provided with a complete (e.g. in the case of the pulsars, complete means a dataset containing 17,898 examples) dataset which has been randomly chosen, with replacement, from the original dataset. The ability to replicate training examples encourages variation among the decision trees and makes for a better prediction algorithm.

In order to divide the data, each decision tree is provided with a random subset of the features from which it can choose (the random selection of the subsets also encourages variation among the trees) to achieve the greatest separation of the data.

To choose the ideal split feature and separation threshold (i.e. the question we ask of the data to maximally separate the data) from the subset, we must iterate through each feature, and within each feature, through each represented feature value. It is important to note that since our features are continuous, we must iterate over each represented value of each feature. For each, we calculate the entropy, which is the measure of how much separation we achieve by using a certain separation feature and feature value. Entropy is defined as :

$$E(S) = -\sum_{i=1}^{n} p(x^{(i)})log(p(x^{(i)}))$$

where $p(x)$ is the fraction of examples of a given class. From this, we can calculate the information gain, which is defined to be the :

$$IG = E(parent) - \frac{N_{left}}{N}E(left) - \frac{N_{right}}{N}E(right)$$

where $E(parent)$ is the entropy of the parent node, before separation, $E(left)$ and $E(right)$ are the entropy of the left node and right node, respectively, and $\frac{N_{left}}{N}$, $\frac{N_{right}}{N}$ are the fraction of examples sorted into the left and right nodes, respectively. By choosing the feature and separation threshold that maximizes the information gain, we can create decision trees that maximally separate the data into the two binary subtrees.

The decision trees are built recursively, with the stopping criterion being either when the tree has reached a predetermined maximum number of nodes or when the number of features in each subdivision has reached a predetermined minimum number.

6 Unsupervised Learning Methods

After implementing supervised learning methods, we applied unsupervised learning methods to further categorize the true pulsar stars in our dataset. For this reason, the following unsupervised learning methods were trained only on the 1,639 positive examples. First, we implemented K-means clustering, which acted as our baseline for unsupervised learning. We then applied Self-Organizing Map, which is a clustering algorithm that uses neural networks, which we implemented from scratch using TensorFlow.

6.1 K-means

The first of the unsupervised algorithms, K-means clustering, is meant to find groupings in the data. The algorithm is initialized with k random vector means $\mu_1, \mu_2, ..., \mu_k$ (random meaning each are assigned to a random data point in the set); it then assigns every training example to the mean that is closest to it. It then reassigns each mean to the mean of the examples that were assigned to it, and begins again by assigning every training example to the new mean that is closest. The algorithm finishes when no training examples are reassigned to a different mean during an iteration.

6.2 Self-Organizing Map

Self-Organizing Maps (SOMs) employ artificial neural networks for unsupervised learning by clustering the multi-dimensional input data on a two-dimensional grid. The neural network has input neurons for each feature, and output neurons (also called map neurons) for each cell in the two-dimensional grid that represent the data received from the input neurons. Each input and output (map) neuron pair is connected to each other, while the output neurons are not connected to each other (this ensures independent learning for each cell on the grid).

Similar to artificial neural networks used for supervised learning, each connection between an input and output neuron has a corresponding weight. However, unlike supervised learning, the weights are not learned through back-propagation using known true labels, and are not propagated forward. Instead, in the context of SOMs, the weights represent a direct relationship between input and output neurons.

For a k-dimensional input data with k features, each of the k weights associated with an output neuron together compose a vector in the feature space. This makes it possible to talk about the Euclidean distance between a single input vector and an output neuron. The output neuron (x, y) (coordinates on the two-dimensional grid) closest to a given input vector $x^{(i)}$ is called the Best-Matching Unit (BMU), and is calculated as follows:

$$(x, y)_{\text{BMU}} = \arg\min_{(x,y)} \sqrt{\sum_{j=1}^{k} (x_j^{(i)} - W_{xyj}))^2}$$

, where W is a tensor containing the weights of connection between input neuron (feature) j and output neuron (x, y). After calculating the BMU, the SOM "pulls" the two-dimensional map toward the input vector. It does this by first calculating a radius σ around the BMU that defines a neighborhood of cells; this radius is decayed exponentially in every epoch. Then, it modifies the weight associated with each cell (output neuron) in this neighborhood to make them even closer to the input vector in question, with cells closer to the BMU being modified more. The new weights are calculated as follows:

$$W^{(t+1)} := W^{(t)} + L^{(t)}\Theta^{(t)}(X^{(i)} - W^{(t)}),$$

where $W^{(t)}$ and $W^{(t+1)}$ are the current and updated weight tensors; $L^{(t)}$ is the current learning rate, which is also decayed exponentially in every epoch; $X^{(i)}$ is the broadcasted input vector $x^{(i)}$; and $\Theta^{(t)}$ is the measure for how much each cell will be modified, which is calculated as follows:

$$\Theta^{(t)} = \exp -\frac{D^2}{2(\sigma^{(t)})^2},$$

where D is the Euclidean distance between a cell and the BMU on the two-dimensional grid.

Once this process is done and the weights are updated for each input vector, the two-dimensional grid map is modified in a way that certain neighborhoods of cells are pulled "closer" to certain input vectors, which define their clusterings. After running this algorithm for several epochs, the output is a mapping of each input vector to the "closest" output neuron. Output neurons with non-empty mappings represent "clusters", and the k weight values associated with them together represent their "centroids" in the feature space.

6.3 Principal Component Analysis

We used Principal Component Analysis (PCA) in order to better represent the high-dimensional data and visualize the results of unsupervised learning. After running the unsupervised learning algorithms on the full dimensionality of the positive-labeled input data and obtaining clusters, we projected this data onto its 2 principal components to make it possible to visualize the clusters on a two-dimensional plot.

It is important to note that PCA was not used before or during unsupervised learning, and was only employed for visualization purposes.

Principal Component Analysis (PCA) first normalizes the given data and then identifies the first k principal components of the data, as the eigenvectors of the covariance matrix Σ corresponding to the largest k eigenvalues.

7 Experiments and Results for Supervised Learning

Our dataset includes 17,898 total examples, 1,639 of which are in the pulsar star (positive) class. We upsampled the positive examples during training so that approximately half the training examples have positive labels. The validation and/or test sets that we evaluated are models on were not upsampled.

7.1 Gaussian Discriminant Analysis

We implemented GDA as the baseline for our supervised learning methods. We held out a test set and evaluated our model on the rest of the data using K-fold cross validation with $k = 5$. After the optimization process was over, we evaluated our model on the test set.

Within each fold, we upsampled positive examples the training set only and evaluated on the validation set. We achieved an average accuracy of 96.66% and an average recall (true positive rate, TPR) of 85.73% across the 5 folds. The average difference between validation and training mean squared error was $3.50 \cdot 10^{-4}$; however, it is important to note that for 3 of our folds, this difference was less than zero.

We then upsampled all positive examples in the data excluding the test set and evaluated on the test set. We achieved an accuracy of 96.28% and a recall of 83.43%. The difference in validation and training error was $4.12 \cdot 10^{-3}$.

Figure 1 shows the true positive and false positive rates of predictions from this model, plotted against probability thresholds.

7.2 Random Forest

We implemented Random Forest as our target supervised learning method, first using the RandomForestClassifier in scikit-learn and optimizing over hyperparameters using RandomSearchCV in scikit-learn, which does a randomized search over hyperparameters. The optimized hyperparameters were:

- n_estimators: 200
- min_samples_split: 5

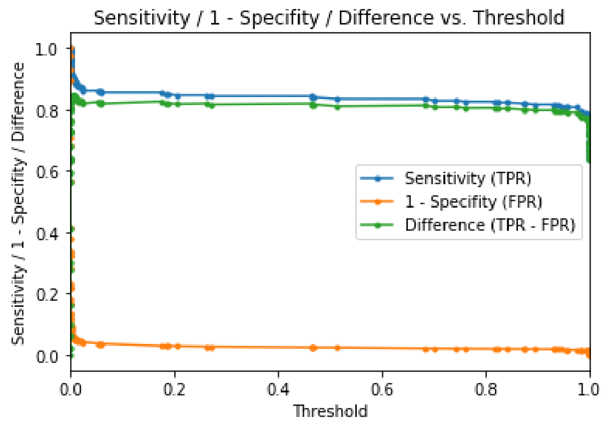

Figure 1: Recall (TPR) and 1−Sensitivity (FPR) values corresponding to the GDA model's predictions on the test set, plotted against probability thresholds

- max_features: 4
- max_depth: 110

We held out a test set and evaluated our model on the rest of the data using K-fold cross validation with $k = 5$. After the optimization process was over, we evaluated our model on the test set.

Within each fold, we upsampled positive examples the training set only and evaluated on the validation set. We achieved an average accuracy of 98.02% and an average recall (true positive rate, TPR) of 85.31% across the 5 folds. The average difference between validation and training mean squared error was $1.85 \cdot 10^{-2}$.

After observing the plots of the true positive and false positive rates of predictions from this model against probability thresholds, we decided to try out using a decision threshold of 0.2 rather than 0.5 to achieve higher recall. This indeed resulted in an average recall of 90.00% across the 5 folds.

We then upsampled all positive examples in the data excluding the test set and evaluated on the test set. We achieved an accuracy of 97.93%. While the recall was 83.13% for a 0.5 threshold, it was 88.00% for a 0.2 threshold. The difference in validation and training error was $1.92 \cdot 10^{-2}$.

Figure 2 shows the true positive and false positive rates of predictions from this model, plotted against probability thresholds. Here, it is possible to see that 0.2 is the threshold that maximizes the difference between the true-positive rate, which is desired to be high, and the false-positive rate, which is desired to be low.

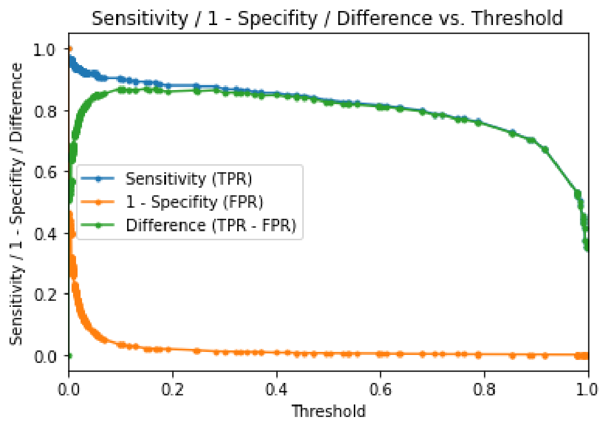

Figure 2: Recall (TPR) and 1−Sensitivity (FPR) values corresponding to the RF model's predictions on the test set, plotted against probability thresholds

Figures 3 and 4 show the Receiver Operating Characteristic (ROC) and Precision-Recall (PRC) curves, respectively.

Figure 5 shows the fraction of positive examples in each probability range, which assesses ability of the RF model to assign probabilities that align with these fractions.

Due to running time limitations, we were unable to use the Random Forest classifier we built from scratch to optimize for hyperparameters and compare with the SciKit model.

8 Experiments and Results for Unsupervised Learning

Our dataset includes 1,639 examples, which are all of the positive-labeled examples from the original dataset.

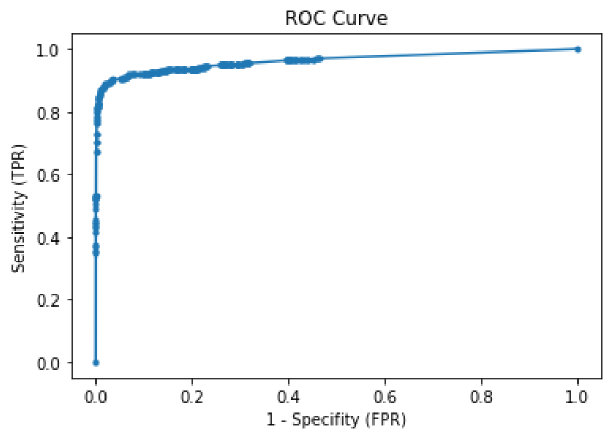

Figure 3: Recall (TPR) and 1−Sensitivity (FPR) values corresponding to the RF model's predictions on the test set, plotted against each other

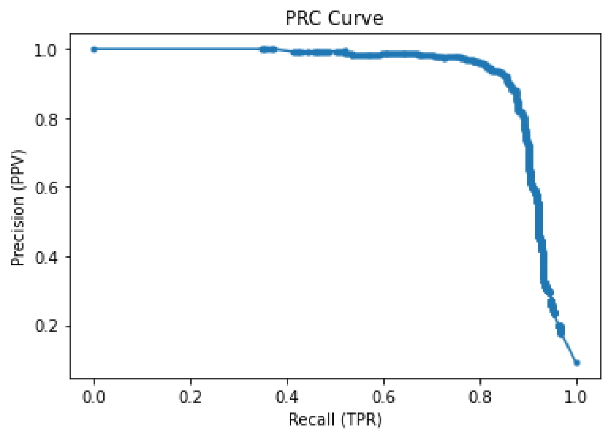

Figure 4: Precision (PPV) and Recall (TPR) values corresponding to the RF model's predictions on the test set, plotted against each other

While Principal Component Analysis was used for data representation and visualization purposes, it was not used before or during clustering, i.e. the clustering models were provided the full dimensionality of the data.

The Silhouette Coefficient was used to evaluate the clustering models. The Silhouette Coefficient for an example is calculated as the difference between the mean nearest-cluster distance for that sample and the mean intra-cluster distance, divided by the maximum of the two. The Silhouette Coefficient for a set of examples is the mean of the value computed for each example. A Silhouette Coefficient of 1 indicates good performance, while -1 indicates bad performance.

8.1 Principal Component Analysis

We calculated the correlations between each feature and the first two principal components, and these correlations are given in Table 1.

In Figure 6, we plotted the normalized correlation between each feature and both principal components. As shown in Figure 6, the features that correlated most strongly with the first component, in decreasing order, were the DM-SNR Profile Skewness, the DM-SNR Profile Mean, the Mean of the Integrated Profile, and the DM-SNR Profile Standard Deviation. Of these features, correlation between feature one and the Skewness of the DM-SNR Profile and the Mean of the DM-SNR profile were both at least twice as the correlation between the other features. The Skewness of the DM-SNR Profile had a positive correlation with the first component, while the Mean of the DM-SNR profile had a negative correlation with the first component. The features that correlated least with the first component, listed in increasing order, were the Excess Kurtosis of the Integrated Profile, the Standard Deviation of the Integrated Profile, and the Skewness of the Integrated Profile.

Feature	Component 1	Component 2
Mean IP	$3.17 \cdot 10^{-1}$	$-2.43 \cdot 10^{-1}$
Standard Deviation IP	$3.27 \cdot 10^{-2}$	$-2.07 \cdot 10^{-5}$
Excess Kurtosis IP	$-1.86 \cdot 10^{-2}$	$1.39 \cdot 10^{-2}$
Skewness IP	$-1.19 \cdot 10^{-1}$	$9.59 \cdot 10^{-2}$
Mean SNR	$-5.77 \cdot 10^{-01}$	$6.63 \cdot 10^{-1}$
Standard Deviation SNR	$-2.57 \cdot 10^{-1}$	$5.78 \cdot 10^{-2}$
Excess Kurtosis SNR	$4.87 \cdot 10^{-2}$	$8.25 \cdot 10^{-3}$
Skewness SNR	$6.954 \cdot 10^{-1}$	$6.99 \cdot 10^{-1}$

Table 1: First 2 principal components of the data represented in the feature space

The features that correlated most strongly with

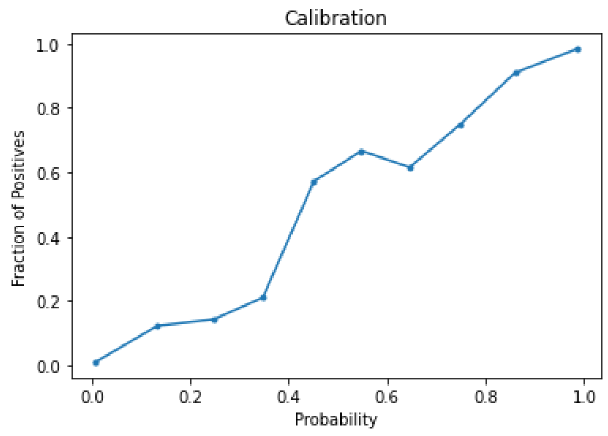

Figure 5: Fraction of samples with positive labels plotted against the 0.1-wide probability range they were assigned by the RF model

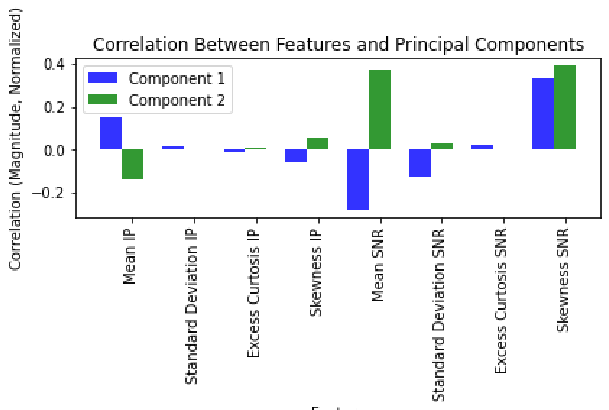

Figure 6: Correlations between each feature and the first 2 principal components of the data

the second component, in decreasing order, were the Skewness of the DM-SNR Profile, the Mean of the DM-SNR Profile, the Mean of the Integrated Profile, and the Skewness of the Integrated Profile. The features that had the weakest correlation with the second component, in increasing order where the Standard Deviation of the Integrated Profile, the Excess Kurtosis of the Integrated Profile, the Standard Deviation of the DM-SNR Profile, and the Skewness of the Integrated Profile.

Thus, we see that it is unclear whether the Mean of the Integrated Profile and the Skewness of the DM-SNR Profile, or the Mean of the DM-SNR Profile and the Skewness of the DM-SNR Profile, are positively or negatively correlated with one another; this is also true for the Mean of the Integrated Profile and the Skewness of the SNR Profile. The remaining features have extremely low correlation values with respect to both principal components and to every other feature (order of magnitude is less than 10^{-2}).

8.2 K-means

We implemented K-means as the baseline for our unsupervised learning methods, with k = 3. The K-means model outputted 3 clusters with centroids and counts as shown in Table 2.

Centroid	Count
$[34.4\ \ 37.4\ \ 4.44\ \ 24.3\ \ 99.1\ \ 72.1\ \ 0.432\ \ -0.330]$	621
$[69.2\ \ 39.2\ \ 2.39\ \ 10.6\ \ 20.6\ \ 48.5\ \ 3.71\ \ 18.8]$	974
$[94.9\ \ 45.2\ \ 0.940\ \ 2.35\ \ 1.35\ \ 12.8\ \ 14.5\ \ 256]$	44

Table 2: Centroids and counts of the 3 clusters produced by the K-means model

The Silhouette Coefficient for the resulting 3 clusters was 0.44.
We visualized these clusters on a 2-dimensional plot by projecting the data on the 2 principal components computed by PCA, as shown in Figure 8.

8.3 Self-Organizing Map

We implemented SOM from scratch with TensorFlow, and applied it on the dataset using a 5x5 output map grid and 100 iterations. The output had 3 non-empty grid cells, i.e. 3 clusters with centroids and counts as shown in Table 3.
The Silhouette Coefficient for the resulting 3 clusters was 0.44. We visualized these clusters on a 2-dimensional plot by projecting the data on the 2 principal components computed by PCA, as shown in Figure 9.

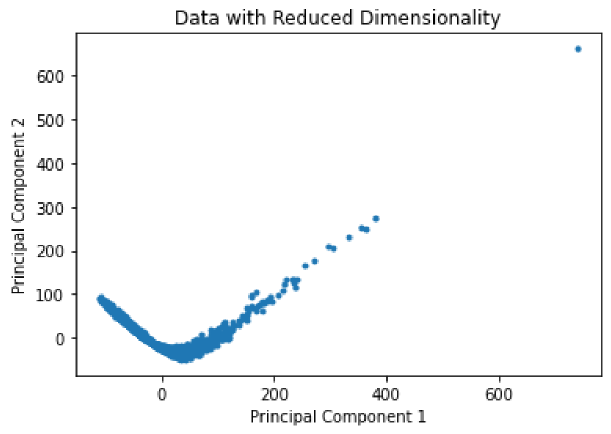

Figure 7: Visualization of PCA-transformed data with reduced dimensionality

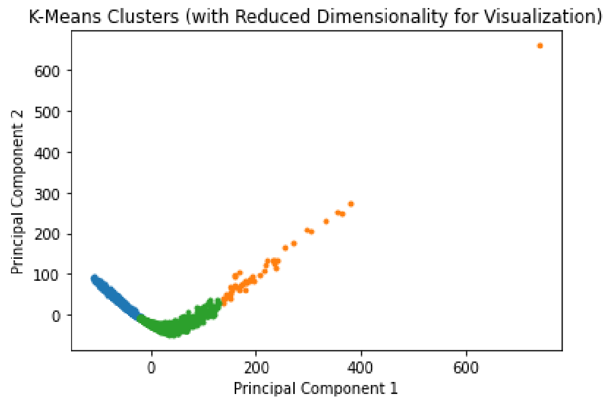

Figure 8: K-means clusters, visualized using PCA transformation

Centroid	Count
$[50.3\ \ 39.8\ \ 3.64\ \ 19.4\ \ 114\ \ 82.1\ \ 0.0525\ \ -0.949]$	517
$[69.5\ \ 39.4\ \ 2.39\ \ 10.8\ \ 28.7\ \ 57.7\ \ 2.89\ \ 11.7]$	1079
$[93.2\ \ 45.1\ \ 1.05\ \ 2.86\ \ 1.33\ \ 12.7\ \ 14.8\ \ 271]$	43

Table 3: Centroids and counts of the 3 clusters produced by the SOM model

9 Discussion

9.1 Discussion for Supervised Learning

We aimed to optimize for recall throughout training and validating our supervised models, since it is more costly to discard a true pulsar star, i.e. a sample of the minority class, than to wrongly classify a signal's source as a pulsar star. This is because pulsar stars are rare among very large datasets, and it is in the best interest of scientists to include a potential candidate in their scope, as long as the size is manageable (which is ensured by the discarding of the majority of the data).

With this objective, we tuned both the hyperparameters for learning and the treshold for prediction to maximize recall, above the baseline of 85.73% achieved by the GDA model. As can be inferred from the flat curves in Figure 1, recall in the GDA model was not susceptible to threshold changes. However, the corresponding plot for the Random Forest model, shown in Figure 2, indicated an optimal threshold of approximately 0.2, where recall is nearly maximized and the false-positive rate is also nearly minimized. Evaluating the Random Forest model on the test set, using the hyperparameters set to maximize recall, we achieved a recall of 88.00%, which is higher than our baseline.

9.2 Discussion for Unsupervised Learning

Our results from PCA indicate that the the first and second principal components primarily measure the Skewness of the DM-SNR profile and the Mean of the DM-SNR Profile. In order to interpret the PCA-transformed data, we plotted the second principal component versus the first principal component (see Figure 7) and achieved a boomerang shape. For an example with a large DM-SNR Skewness, we would be hard pressed to predict the mean of the DM-SNR profile as this value decreases as principal component one increases, but also increases as principal component two increases, despite both components being increasing with an increasing DM-SNR Skewness. In other words, as the Skewness of the DM-SNR Skewness Profile increases, we could expect to

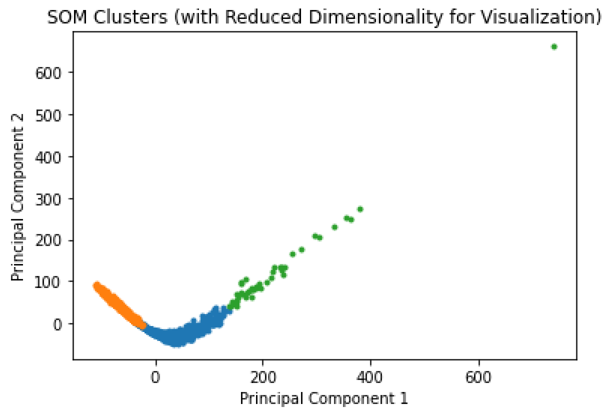

Figure 9: SOM clusters, visualized using PCA transformation

see the Mean of hte DM-SNR Profile increase or decrease. However, we would expect that examples with a large SNR mean should have a large SNR Skewness. However, for an example with a very large or small SNR Profile mean, we would expect a large DM-SNR Skewness.

In the first half of the boomerang shape where the plot has a negative slope, we see that the first principal component increases as the second principal component decreases, so we see that the Mean of the DM-SNR Profile also decreases. The Skewness of the DM-SNR Profile, however, it the behavior of the Mean of the DM-SNR Profile is unclear. At the minima, this trend changes however; we see that we the first and second principal components increase together, thus we would predict that the Skewness of the DM-SNR Profile increases, while the behavior of the Mean of the DM-SNR Profile is unclear. So we can conclude that for the examples in the section of the graph that is decreasing, the Mean of the DM-SNR Profile is the feature that determines the primary principal component, while for the examples in the section where the graph is increasing, the Skewness of the DM-SNR Profile is the feature that determines the primary principal component. The linear trend follows from the fact that the magnitude of correlation between both features for each component are similar. The magntiudes of the correlations between both of these features and each principal component are fairly similar, which explains why the curve, on either side of the minima, is fairly linear.

The unsupervised models were trained on the portion of the dataset comprised of true pulsar stars, and it is important to note that the dataset does not have any labels corresponding to different categories of pulsars. In general, pulsars are classified according to their spin frequency. This is information that is not represented in our data, since the integration involved in generating the "integrated pulse profile" eliminates information about frequency and only gives information about phase *in relation* to frequency. Thus, in our analysis of the outcomes of the K-means algorithm as well as the Self-Organizing Map, we were unable to gain much insight because the examples have been essentially anonymized in regards to frequency and other features used to classify pulsars. This problem is reflected on the clustering performances: the Silhouette Coefficients for both models were 0.44, with 1.00 as the target performance. The visualizations in Figures 8 and 9 also demonstrate the lack of distinct clusters.

Thus it would be of value to apply these algorithms to data sets that contained the unprocessed frequencies of the pulsar stars, and we plan to pursue this in the future. In addition to this, we would like to work on optimizing the Random Forest implementation written from scratch for run time in order to properly optimize for hyperparameters.

10 Acknowledgements

We would like to thank Prof. Moses Charikar, Prof. Chris Ré, Prof. Tengyu Ma, and the rest of the CS 229 staff for their support throughout the quarter.

11 Member contribution

12 Works Cited

Buckland, Mat. “Kohonen's Self Organizing Feature Maps.” Ai-Junkie, www.ai-junkie.com/ann/som/som1.html.

Ford, John M. 2017. Pulsar Search Using Supervised Machine Learning. Doctoral dissertation. Nova Southeastern University. Retrieved from NSUWorks, College of Engineering and Computing. https://nsuworks.nova.edu/gscis_etd/1001.

"Implementing Self-Organizing Maps with Python and TensorFlow." Rubik's Code, 27 Aug. 2018, rubikscode.net/2018/08/27/implementing-self-organizing-maps-with-python-and-tensorflow/.

R. J. Lyon, B. W. Stappers, S. Cooper, J. M. Brooke, J. D. 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 459 (1), 1104-1123, DOI: 10.1093/mnras/stw656

R. J. Lyon, HTRU2, DOI: 10.6084/m9.figshare.3080389.v1.

R. P. Eatough et al., "Selection of radio pulsar candidates using artificial neural networks," in Monthly Notices of the Royal Astronomical Society, vol. 407, no. 4, pp. 2443-2450, Oct. 2010, doi: 10.1111/j.1365-2966.2010.17082.x.