IJHSR
ijhighschoolresearch.org
■ RESEARCH ARTICLE

Random Forest Identification of Pulsars

Ankhita Ravishankar Sathanur

Eastlake High School, 400 228th Ave NE, Sammamish, WA 98074, USA; ankhitas@hotmail.com
Mentor: Kristen Surraro

ABSTRACT: This study explores the implementation of a random forest classifier to identify pulsar signals from a large sample of survey data. Pulsars are a unique type of rotating neutron star that emit pulses of radio emission in beams that sweep across Earth, allowing for the detection of their repetitive pulses. Traditionally, pulsar candidates have been identified through manual signal processing. As data volumes increase, automated methods, like artificial neural networks, have been proposed. In this study, the random forest classifier– an algorithm that takes the majority output of multiple decision trees –was used to separate pulsar signals from radio frequency interference (RFI) and other noise. 1,639 real pulsar examples and 16,259 samples of RFI/noise from the HTRU2 survey were used to create the model. Features of the data used include the mean, standard deviation, excess kurtosis, and skewness of the integrated pulse profile and DM-SNR curve. The model demonstrated a 95% accuracy in identifying pulsars. The excess kurtosis, skewness, and mean of the integrated profile were determined to be the most critical factors in differentiating between pulsars and interference. This tool could filter data from future surveys to reduce the number of candidates that need to be processed by humans.

KEYWORDS: Robotics and Intelligent Machines; Machine Learning; Random Forest; Physics and Astronomy; Pulsars.

■ Introduction

Pulsars:

Pulsars form from neutron stars. A neutron star is a leftover core from when a star with a mass between 8 and 20 solar masses dies in an explosive supernova. This core is extremely massive, around 1.5 to 3 times the sun's mass, and very small, typically only around 20 to 24 km in diameter (roughly the size of a small city block).[1] These features contribute to a pulsar's high density, comparable to that of atomic nuclei, second only to black holes.

One of the most notable features of pulsars is their rapid spin rate. This spin is left over from the original rotation of the living star from which the pulsar formed. As a star collapses into a neutron star, its spin rate increases dramatically: as the size of the star shrinks significantly, the rotation rate must increase to conserve angular momentum.

Pulsars are also highly magnetic, with a magnetic field 100 million to 1 million billion times stronger than Earth's magnetic field.[1] A neutron star with the right combination of extreme magnetism and rapid spin is described as a pulsar.

The light pulsars emit can be attributed to the rotating magnetic field. This creates an electric field where charged particles move and form an electric current. As these charged particles are accelerated to high speeds by the magnetosphere, the region above the surface of a pulsar, the pulsar radiates visible, radio, and x-ray light. Pulsars emit these beams of light from their north and south magnetic poles, which typically differ from their axis of rotation. This allows their beams of light to be viewed as pulses from Earth, as the coherent light (emitted like a laser rather than a lightbulb) sweeps across Earth. However, if the magnetic poles aligned with the rotation poles, the light would be viewed as a steady stream rather than as pulses, as the beams would not move as the star rotates.

Pulsars are valuable tools in studying the universe. For one, they are helpful in studying extreme states of matter due to their high density, which creates what astronomers call a "nuclear pasta" as atoms arrange in unusual shapes and patterns. For example, within neutron stars, atoms arrange themselves in patterns such as flat sheets, spirals, and small nuggets that are not seen anywhere else.[2]

While pulsars reduce speed as they lose energy to light, their rotation rate changes at a slow, almost imperceptible rate. They are, therefore, so dependable that astronomers can predict when a pulse will occur years in the future with an accuracy of 100 nanoseconds. This accurate, reliable spin rate makes pulsars helpful in finding exoplanets, as a nearby planet will often cause noticeable disturbances in a pulsar's spin. In fact, the first planet discovered outside our solar system was discovered orbiting a pulsar. The constant spin of pulsars as they move through space also makes them a useful tool in measuring cosmic distances as they move through space while blinking a known number of times per second. Finally, pulsars are useful in searching for gravitational waves, which also create disturbances in the regularly timed pulses.

Pulsar Detection:

Pulsars are extremely rare, and only around 2,000 have been detected to date. A typical method for detecting pulsars is all-sky surveys, where a telescope scans the entire sky and looks for light that flickers over time. The Parkes radio telescope in Australia has discovered the majority of pulsars. Still, other radio telescopes, such as the Arecibo telescope in Puerto Rico, the Green Bank telescope in West Virginia, the Molonglo telescope in Australia, and the Jodrell Bank telescope in England,

DOI: 10.36838/v5i4.3

have also made contributions. In addition, the Fermi Gamma Ray Telescope has also detected gamma-ray-emitting pulsars.

Each pulsar has a unique spin rate and radio pulse profile, both of which are used in their detection. Traditionally, pulsar candidates have been identified through manual signal processing, where humans visually identify the emission spectrum from the data collected by telescopes.[3] Not only is this process time-consuming and mentally demanding, but it also introduces human error into pulsar identification. However, as the technical capabilities of pulsar searches continue to grow and change - for example, through increasing bandwidth, sky coverage, sensitivity, and, most notably, frequency resolution - the number of candidates has risen.[4] This makes manual processing no longer feasible. In response, some data filtering tools have been created to filter out most of the noise and interference before signals reach human eyes. However, even these filtering methods are no longer practical, as they return more viable candidates than can be manually processed. The lack of an efficient and accurate pulsar classification method has created this 'candidate selection problem'.[4]

Machine Learning:

Machine learning is a branch of computer science and artificial intelligence that uses data and algorithms to imitate how humans learn.[5] It allows computers to learn without being explicitly programmed in a way that continuously improves their accuracy.

The learning system of a machine learning algorithm can be split into a decision process, an error function, and a model optimization process.[5] In the decision process, the model processes training data to create the model and then uses the patterns and correlations it identified in the training process to make a decision, for example, a prediction or classification, about new test data.[6] An error function refers to the process of assessing the accuracy of the model. Finally, the model optimization process describes fine-tuning the algorithm to fit the training data better to attain a higher accuracy in the testing phase.

Machine learning can be subdivided into supervised and unsupervised machine learning. Supervised machine learning is a subset of machine learning where the machine uses the training examples with labels or targets. These labels help the algorithm to correlate features and ultimately identify patterns in the dataset. Supervised learning is instrumental in solving problems that fall under the categories of classification or regression. In classification problems, the machine predicts the most probable category, class, or label for new examples, while in regression problems, the machine predicts the value of a continuous response variable. Methods used in supervised learning include neural networks, linear regression, logistic regression, random forest, and support vector machines, and other methods.[5]

Unsupervised learning refers to a model training process where the training data do not have attached labels. Instead, the machine searches for patterns in the data.[5] Unsupervised learning is useful in solving problems that require clustering, as the machine can group data points with similar features.

Decision Trees:

A decision tree classifier is a supervised machine learning algorithm that uses a set of rules to make decisions, much like humans.[7] The original collection of training data is entered at the start, called the root node. Decision trees use dataset features to create a list of yes/no questions. Each question is marked with a node. The questions continuously split the dataset into smaller subsets, where all the data points that correlate to the answer 'yes' branch into one group, and the remaining data points branch to create a second group.[7] In this fashion, the data becomes organized into a tree structure. This splitting of nodes continues until there are no more rules to apply or no data points left. These final nodes are called leaf nodes. At this point, each leaf node must be assigned a class. If the data in a leaf node is fully isolated by class (all of the data points in the training set are in groups that only contain the same label), it is referred to as a pure leaf node, where the class is assigned as the common label. However, leaf nodes are often not 100% pure and are thus called mixed nodes. In this case, the algorithm assigns the most common class among the data points in the node as the common label.[7]

Ideally, a decision tree will have the smallest number of splits possible while maintaining the highest accuracy. However, this approach could be more computationally infeasible, mainly as larger datasets are used and the time to build the tree grows. The next best approach in creating the best tree is through the greedy approach, which attempts to make the locally optimal decision at the current node rather than the best decision for the tree overall.[7] In making the best split, decision trees aim to divide the dataset into the smallest subset possible, so ultimately, the goal is to minimize the loss function. Loss functions are mathematical equations, known as criteria, that calculate the information gained when a node is split. Criteria are used to decide which features are most efficient to split on. The three main loss functions used in decision tree algorithms are Gini, Entropy, and Log Loss. All three functions are measurements of error.

Decision trees are useful in solving both classification and regression tasks. They allow for easy interpretability, as decision trees are simple, easily visualized, and understood. They are also useful for their data robustness since they can process numerical, categorical, and Boolean data. Decision trees also readily provide information about the data and the relative importance of various features. The most important features are used to split nodes higher up in the tree by the need to create the most efficient splits possible.

Random Forest Classification:

Random forests are a collection of independent decision trees that act as an ensemble.[8] When a test data point is run through a random forest, each tree will make its own class prediction based on its unique model. The class with the most votes (the class the majority of the trees predict) becomes the entire forest's prediction. The underlying reasoning is that a group of relatively uncorrelated trees acting as a group will outperform any individual tree.[8,9]

The low correlation between individual trees is key to the accuracy of the random forest. It ensures that an error in one

DOI: 10.36838/v5i4.3

tree will not be matched in other trees and that the random forest is collectively protected from individual errors.

Two methods random forests use to ensure this diversity in trees are bagging and feature randomness.

Bootstrap Aggregation, also known as bagging, is the process where each tree uses a unique dataset sample to build its tree through random sampling with replacement. Rather than splitting the entire dataset into smaller chunks for each tree, random sampling with replacement allows each tree to be trained on a dataset equal to the size of the entire dataset. However, instead of feeding each tree the entire original dataset, each tree chooses a random sample of data points (equal to the total number of data points in the dataset) with replacement.[8] Since the trees sample with replacement, duplicate data points can be sampled and used to train a single tree.

The second method used to ensure a low correlation between trees is feature randomness. When splitting a node in a regular decision tree, all the features of the dataset are considered. The split is made using the feature that creates the greatest separation between the two resulting branches. However, in random forest trees, each tree can only determine the best split from a randomly selected subset of the dataset features. This allows for even more variation among the trees in the forest.[8]

Ultimately, random forests serve as highly accurate and efficient models for solving classification and regression problems and can generally produce better results than any single decision tree.[9]

Machine Learning in Pulsar Identification:

Various machine learning methods have recently been proposed to address the growing need for automation in pulsar classification. However, the majority of these algorithms are based on artificial neural networks (ANNs).

ANNs are computing systems inspired by the biological neural networks used for decision-making in the human brain. They are instrumental when solving problems related to pattern recognition and classification, approximation, optimization, and data clustering.[10] ANNs use an extensive collection of units or nodes that work as artificial neurons, and each act as individual simple processors that operate in parallel. Neural networks contain an input layer, one or more hidden layers, and an output layer. Each node is connected to every other neuron in the layers above and below it using a connection link, and each has a weight threshold.[10,11] If the output of any individual node is above this threshold value, that node is activated and sends data to the next layer of the network. If the threshold is not met, no data is passed along to the following network layer.[10]

ANNs have been frequently proposed as a solution to automating the classification of pulsars. For example, Bates *et al*. (2012) developed a neural network that could detect 85% of real pulsar candidates over two years of data from the mid-latitude portion HTRU survey. Candidate parameters used included the pulse period in milliseconds, the pulse width, the DM in cm^{-3} parsecs, the signal-to-noise ratio of the detection, and a unique $\chi 2$ value calculated from fitting the pulse profile with a sine function.[12] However, their study did not utilize a representative sample of the pulsar population during the training process. In addition, they used a single artificial neural network to detect different types of pulsars, likely contributing to the lower accuracy.

Eatough *et al*. (2010) also implemented a neural network to re-analyze Parkes Multibeam Pulsar Survey (PMPS) data. They were able to discover a previously unknown pulsar. This neural network was trained using a particular set of scores to identify credible pulsar candidates automatically. The tool recovered 92% of pulsars present in a test sample of approximately 2.5 million candidates. Shortcomings included the poor training of the ANNs on MSPs, unbalanced training sets, and abnormal candidate plots generated by search software, which made it unlikely that the tool would identify MSPs.[13]

Morello *et al*. (2014) developed a third neural network, the Straightforward Pulsar Identification using Neural Networks (SPINN), to process HTRU survey data. The algorithm identified every known pulsar in the southern survey data with a false positive rate of only 0.64%. It also identified four new pulsars in re-processing the intermediate galactic latitude area of HTRU, three of which have spin periods shorter than five milliseconds. Pulse features used in developing SPINN include the S/N of the folded profile, which is a measure of signal significance, the ratio between period and DM, the intrinsic equivalent duty cycle of the pulse profile, which is the ratio of a pulsar's pulse width in seconds to its spin period, a measure of the validity of DM, the persistence of the signal through time, and a measure of the variability of the pulse shape during the observation.[14]

This study aims to explore the implementation of another machine learning tool, the Random Forest Classifier, as a solution to the pulsar 'candidate selection problem. In addition to ANNs and other previously proposed tools, the Random Forest Classifier can provide an efficient and accurate solution to identifying pulsar signals from a large sample of survey data.

Methods

Dataset:

In developing our tool, we used a sample of pulsar candidates from the South High Time Resolution Universe Survey (HTRU2).[15] Data for the southern hemisphere portion of the HTRU survey was collected using the Parkes Multibeam Receiver. The dataset used came from the UCI Machine Learning Repository. The dataset contains 17,898 data points including 1,639 real pulsar examples and 16,259 samples of RFI/noise. The disparity between the number of real pulsar examples and the interference signals speaks to the rareness of true pulsar signals. Features of the data we used included the mean, standard deviation, excess kurtosis, and skewness of the integrated pulse profile and DM-SNR curve.

The integrated pulse profile is a superposition of hundreds of thousands of individual pulsar pulses. The integrated profile is unique to each pulsar and can be recreated anytime.[16]

The Dispersion-Measure-Signal-to-Noise-Ratio curve, or the DM-SNR curve, accounts for the dispersion of pulses. Radio pulses arrive at different times across different radio frequencies due to the ionized interstellar medium radio signals travel through before they reach Earth.[17] This delay across frequencies is referred to as dispersion. Astronomers fit the shape

of the delay when creating the pulse profile to compensate for its effect; however, there remains uncertainty with the fit. This uncertainty is expressed through a DM-SNR curve.

The mean, standard deviation, excess kurtosis, and skewness of both the integrated pulse profile and the DM-SNR curve were used as features of the radio signals in training the machine learning model.

The mean refers to the average. Mathematically speaking, it is the sum of a collection of values divided by the number of values in the collection. In reference to the integrated profile, the mean refers to the average pulse energy associated with the profile. The mean of the DM-SNR curve is the average of the curve.

The standard deviation measures how much individual data points vary from the mean. For the pulse profile and DM-SNR curve, it measures how much individual pulses differ from the mean.

The kurtosis and skewness both refer to the shapes of the curves. Excess kurtosis describes how tailed distribution is relative to a normal distribution of data. As in Figure 1, a curve with fewer outliers would appear thin-tailed and have a negative kurtosis (less than 3), while a curve with many outliers would look fat-tailed and have a positive kurtosis (greater than 3).[18] A positive skew indicates that the curve is left-modal and skewed to the left, and a negative skew suggests that the curve is right-modal and skewed to the right, as seen in Figure 2.

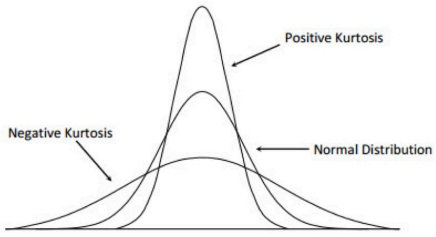

Figure 1: Depiction of excess kurtosis.[19]

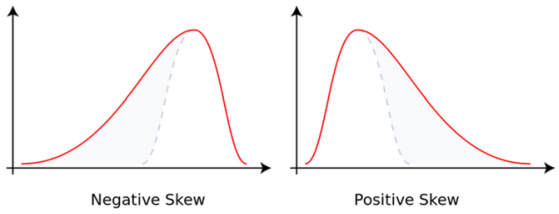

Figure 2: Depiction of skew.[20]

Python Libraries and Packages:

In creating the random forest classifier, we employed several python packages. Pandas, an open-source data analysis and manipulation tool, was used to format and visualize the data from a CSV file. The Scikit-Learn (sklearn) python machine-learning library was used in creating the bulk of the program. The method train_test_split was used from sklearn.model_selection to split the dataset into training and testing data, with 70% of the data used for training and 30% used for testing. The proportion of pulsars to noise was maintained during the split into testing and training data, and the method RandomForestClassifier was used from sklearn.ensemble to build the random forest. The metrics package from sklearn was used to calculate the accuracy of the random forest classifier, and the tree package was used to visualize individual estimators (a single decision tree). Finally, the matplotlib.pyplot library was used to create graphs and plots to visualize the impacts of various hyperparameters on the accuracy of the classifier. The matplotlib and seaborn libraries were also used to visualize relative feature importance.

Dataset Balancing:

The dataset used to build the random forest classifier is extremely imbalanced with roughly 90% noise data and only 10% real pulsar data. This has implications for the accuracy of the model, as generally, machine learning models tend to ignore and perform worse on the minority class. Again, this has consequences for the model as it identifies pulsars, the minority class. While testing dataset balancing methods, all hyperparameters of the random forest classifier were kept at their default values to ensure consistent results.

Initially, the model was built on the raw, unbalanced dataset, which contained 1,639 real pulsar examples and 16,259 samples of noise, to analyze the effects of data imbalance on the model. The resulting training and testing confusion matrices are shown in Figure 3.

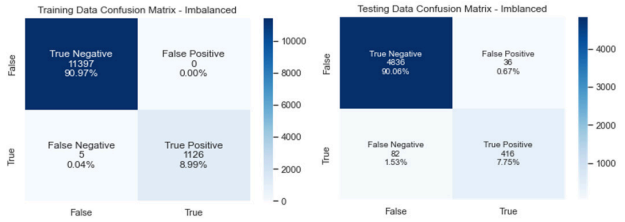

Figure 3: Confusion matrices resulting from the imbalanced dataset.

The first method of dataset balancing tested, referred to as using a random subset of the noise data, involved cutting out noise data to the number of pulsar data before splitting it into training and testing data and building the model. This was achieved using the .sample() method through Pandas, which selected a random sample of 1,639 data points of noise data, equivalent to the number of pulsar data points included in the dataset. The resulting training and testing confusion matrices are shown in Figure 4.

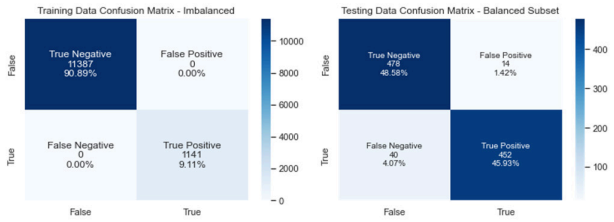

Figure 4: Confusion matrices resulting from a balanced dataset that was achieved by randomly sampling a subset of noise data equivalent to the pulsar data.

The final dataset balancing method was the Synthetic Minority Oversampling Technique (SMOTE). This method addressed imbalance by oversampling the minority class. This is done by synthesizing new data points from existing examples. The resulting training and testing confusion matrices are shown in Figure 5.

DOI: 10.36838/v5i4.3

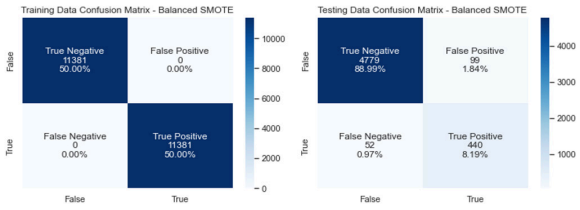

Figure 5: Confusion matrices resulting from balanced dataset achieved using SMOTE (oversampling minority class by synthesizing minority data points based on existing samples.)

Detailed analysis was performed to determine each balancing approach's strengths and weaknesses and compare the balanced approaches to the imbalanced method. In doing so, the blind approach was used as a baseline for comparing balancing method performance. It was derived as such: using the imbalanced dataset, if all cases are predicted as noise, the resulting accuracy is 16,259 noise cases /17,898 total cases = 0.908426 in the full dataset. As shown in Figure 6, using this blind approach, a method that misses all pulsars, 100% of actual pulsar cases will be recognized incorrectly as noise, and 90.84% of all test cases predicted as noises will be real noise. Thus, 0% of actual pulsar cases will be recognized correctly as pulsars, as the method will not predict any pulsars.

	Predicted Noise/ Actual Noise (%)	Predicted Pulsar/ Actual Pulsar (%)	Real Noise/ Predicted Noise (%)	Real Pulsar/ Predicted Pulsar (%)	Average (%)
Imbalanced	99.26	83.53	90.54	92.04	91.3425
Balanced using SMOTE	97.97	89.43	90.83	81.63	89.965
Balanced using Subset of Noise	97.15	91.87	90.54	97	94.14
Blind (Imbalanced)	100	0	90.84	N/A	63.61333333

Figure 6: Comparison of balancing methods, using SMOTE and a subset of the noise data, to imbalanced methods, and their performance in four categories: predicting actual noise, predicting actual pulsars, the accuracy of all predicted noise and accuracy of all predicted pulsars.

In analyzing the balanced methods, using a random subset of the noise data, on average, provided better results than the imbalanced model. Using SMOTE, on average, provided comparable results to the imbalanced model.

Balancing using SMOTE and a random subset of the noise data provides comparable results when classifying noise data; SMOTE is only marginally more accurate. Using SMOTE, 97.97% of actual noise cases were recognized correctly as noise, comparable to 100% by the blind approach and 99.26% using an imbalanced dataset. Furthermore, using SMOTE, 90.83% of all test cases predicted as noise were real noise, comparable to 90.84% by the blind approach and 90.54% by the imbalanced method. However, SMOTE needed to be more accurate in classifying pulsar data than balancing using a random subset of the noise data. 89.43% of actual pulsar cases were recognized correctly as pulsars, which is still significantly higher than 0% by the blind method and 83.53% by the imbalanced method but is less accurate than 91.87% using a random subset of the data. Using SMOTE, 81.63% of all test cases predicted as pulsars were real pulsars, which is significantly less accurate than 92.04% by the imbalanced approach and 97.00% using a random subset of the noise data.

Balancing using a random subset of the noise data is more accurate in classifying pulsar data. Using a random subset, 91.87% of actual pulsar cases were recognized correctly as pulsars, higher than 83.53% by the imbalanced model and significantly higher than the 0% by the blind approach. In addition, 97.00% of all test cases predicted as pulsars were real pulsars, over 15% higher than the 81.63% using SMOTE. However, it needed to be more accurate in classifying noise data. 97.15% of actual noise cases were recognized correctly as noise, which is notably less than 99.26% by the imbalanced approach and 100% by the blind approach. Also, 90.54% of all test cases predicted as noise were real noise, equal to 90.54% using the imbalanced method and comparable to 90.83% using SMOTE and 90.84% by the blind approach.

Cumulative averages show that using a subset of the noise data is roughly 4.175% more accurate overall than using SMOTE, 2.798% more accurate than the imbalanced method, and 30.527% more accurate than the blind approach.

The method train_test_split was used from sklearn.model_selection to split the dataset into training and testing data, with 70% of the data used for training and 30% used for testing. Next, the dataset was divided after balancing methods were applied. In the imbalanced method, the proportion between noise and pulsars was maintained, and 70% of each of the noise and pulsar cases was used for training, and 30% of the noise and pulsar cases were used for testing. In the approaches where the dataset was balanced, the entire dataset was split into 70% training data and 30% testing data. As a result, training and testing data were half pulsar data and half noise data.

Hyperparameter Evaluation Metrics:

Hyperparameters are adjustable parameters set before algorithm training, making it possible to control the model training process.[22] Hyperparameters were tuned after balancing the data using a subset of the noise data, which was the most accurate dataset balancing method overall.

In assessing the accuracy of the model as different hyperparameters were tuned, the evaluation metric *Classification Accuracy* was selected. Classification accuracy is defined as the ratio of the number of correct classifications, where the class predicted by the model is the same as the class given in the input dataset, to the total number of predictions made by the model. This ratio is bound between 0 and 1.[22] This evaluation metric was selected since the task was binary in nature (pulsar vs. non-pulsar), and all the predictions and prediction errors were therefore equally important. The results from this metric are illustrated as a plot.

The model's accuracy relative to various hyperparameters was also visualized through a *Receiver Operating Characteristic Curve* (ROC curve). A ROC curve is a valuable evaluation metric for supervised binary classification problems. In creating a ROC curve, the true positive rate (TPR) is plotted against the false positive rate (FPR). It allows the tradeoff between sensitivity, the TPR, and specificity, the FPR to be visualized. As in Figure 7, the true positive rate is the ratio of "true" events correctly identified by the algorithm to the total number of "true" events in the testing dataset. The false positive rate is the ratio of "false" events that were incorrectly classified as "true" events to the total number of "false" events.[22] In the best-case scenario, the true positive rate should be 1, while the false positive rate should be 0. For example, the true positive rate in pulsar classification refers to the number of pulsars correctly identified as pulsars. In contrast, the false positive rate refers to the number of non-pulsars wrongly identified as pulsars. ROC curves are especially useful in comparing the performances of

different supervised learning algorithms, in this case various random forest classifiers with different hyperparameters, by selecting the algorithm with the greatest area under the curve.

$$TPR = \frac{TP}{TP + FN} \quad FPR = \frac{FP}{FP + TN}$$

Figure 7: True positive rate and false positive rate equations. TP represents true positive, FN represents false negative, FP represents false positive, and TN represents true negative.[23]

The greater the area under the ROC curve (AUC), the more accurate the classification model. Using integral calculus, the AUC is found by calculating the area under the curve from (0,0) to (1,1). AUC is a value from 0.0 to 1.0. An AUC of 0.0 means the model has classified 100% of the test data incorrectly, while an AUC of 1.0 means that 100% of the data was classified appropriately.[23]

In using both metrics, the aim was to understand the model's accuracy in identifying pulsars and to ensure that the false positive identification rate remained low. If the false positive value was to grow, and increasing numbers of false signals were incorrectly identified as real pulsar signals, valuable and expensive telescope time would be allotted to search in unnecessary places, thus defeating the purpose of this tool in reducing the time to identify pulsars accurately.

Hyperparameters:

Various hyperparameters were tested to increase the accuracy of the classifier. min_samples_leaf, min_samples_split, max_leaf_nodes, criterion, and n_estimators were the parameters tested. Each of these hyperparameters was tested individually to visualize the impact of increasing or decreasing their values. The remaining hyperparameters were kept at their default values during each test, with n_estimators = 100 across all tests. Several correlations were identified.

5.31 min_samples_leaf:

The min_samples leaf hyperparameter determines the minimum number of samples required to be at a leaf node, the final nodes that make up the base of each decision tree.[24] Values tested for min_samples_leaf were 1, 2, 5, 10, 50, and 100. As seen in Figure 8, at low values, from 1 to 10, the accuracies appeared to fluctuate but remained high in accuracy. However, the accuracy dropped at higher values. For example, setting min_samples_leaf to 5 gave the highest accuracy of 0.9502 and the highest AUC of 0.9482 (equal in accuracy to setting min_samples_leaf to 2), indicating that while the model is highly accurate, it is also maintaining the necessary low false positive rate. The observed results are likely because lower values of min_samples_leaf allow the tree to have more flexibility, create more splits, and have more individual leaf classes at the base of the tree, thus making it more accurate when tested.

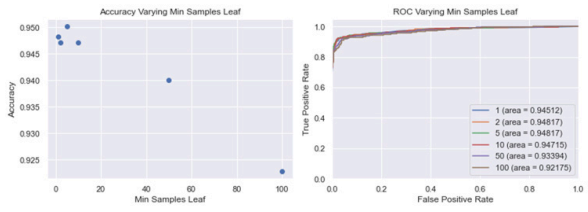

Figure 8: Accuracy plot and ROC varying min_samples_leaf hyperparameter.

The min_samples_split hyperparameter determines the minimum number of samples needed to split an internal tree node.24 The values 2, 5, 10, 50, and 100 were tested but demonstrated no clear correlation. As visualized in Figure 9, the AUC remained uniformly high between 0.94614 and 0.94817. Therefore, this parameter was not used in the final set of hyperparameters.

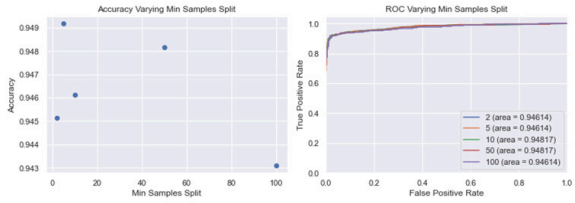

Figure 9: Accuracy plot and ROC varying min_samples_split hyperparameter.

max_leaf_nodes:

Max_leaf_nodes is the hyperparameter that sets the limit on the splitting of nodes to reduce the depth of the tree, which reduces overfitting. A greater value of max_leaf_snodes will create a deeper tree with more splits.[24] Values tested were 10, 20, 50, 100, 150, 200, 500, 750, and 1000. There was initially a positive correlation when plotting max_leaf_nodes versus accuracy. In Figure 10, as the number of leaf nodes grew, the accuracy also increased, with the greatest increase in accuracy between 10 and 20 leaf nodes. However, the accuracy dropped suddenly after 150 leaf nodes. The AUC at all tested values remained high (> 0.94), with the greatest AUC of 0.94817 when max_leaf_nodes was set to 150. As the max_leaf_nodes value grows, the tree can create more splits and become more constrained to the training data, allowing it to make more accurate decisions on the test data. However, suppose the value becomes too large. In that case, the tree can begin overfitting the training data to such precision that it cannot accurately address variances in the test data and thus begins making incorrect classifications, leading to the observed drop in accuracy.

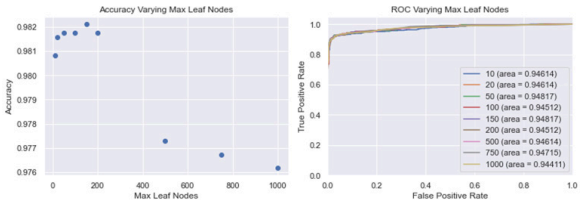

Figure 10: Accuracy plot and ROC varying max_leaf_nodes hyperparameter.

Criterion:

Criteria, as mentioned prior, are mathematical equations used to determine the best split at a node and are used to maximize the information gained with each division.[24] We tested the three main criteria, 'Gini,' 'Entropy,' and 'Log Loss,' to see which yielded the highest accuracy. Gini is calculated by subtracting the sum of the squared probabilities of each class from one.[25] Log Loss is a formula for calculating information gain, mainly used to train binary classifiers.[26] Entropy is a third criterion formula for calculating information gain. Lower entropy correlates to increased model accuracy.[27] The results in Figure

DOI: 10.36838/v5i4.3

11 showed that 'Gini' was the most accurate criterion for our model.

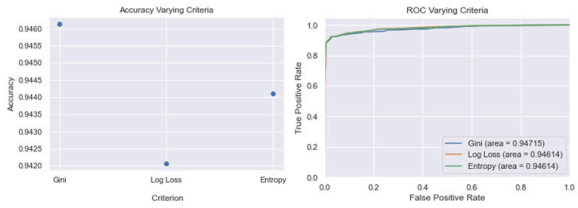

Figure 11: Accuracy plot and ROC varying criteria hyperparameter.

n_estimators:

n_estimators refers to the number of estimators, or independent decision trees, present in the random forest. There is a pre-existing understanding that as the number of estimators increases, the accuracy will also increase as errors in individual trees are more frequently canceled out. This occurs only at the expense of run time, as increasing the number of estimators will take the processing unit more time to build the random forest.[28] Since this correlation is well known, n_estimators was the last hyperparameter tuned after the other hyperparameters and their values were already optimized. The accuracy of the random forest was tested with 1, 2, 5, 10, 20, 50, 100, 200, 300, 400, and 500 estimators. As in Figure 12, plotting the resulting accuracies demonstrated that the accuracy appears to plateau after 50 estimators. The AUC remains high and fairly constant after 50 estimators, only slightly ranging from 0.9461 to 0.9502.

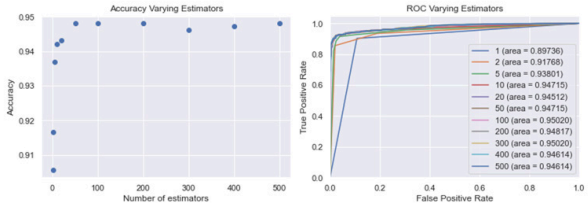

Figure 12: Accuracy plot and ROC varying n_estimators hyperparameter.

Results and Discussion

We looked at trends in the hyperparameter versus accuracy plots and ROCs to determine the parameters used in our final model. The final values were set as follows: n_estimators = 50, min_samples_leaf = 5, max_leaf_nodes = 150, criterion = 'gini.'

In assessing the model's accuracy using the evaluation metric Classification Accuracy as defined prior, our model demonstrated an accuracy of 0.9502. There are slight variations in the accuracy as the forest is rebuilt, as no two forests are identical.

The confusion matrices for the final model are shown in Figure 13. Analysis of the confusion matrices reveals that this model performs accurately on both noise and pulsar data. Using this model, 97.56% of actual noise cases were recognized correctly as noise, comparable to 100% by the blind approach. Using this algorithm, 90.84% of all test cases predicted as noise were real noise, equivalent to the accuracy of the blind approach. The algorithm is very accurate in classifying pulsar data, which is the advantage of using this algorithm over the blind approach, which does not identify any pulsars at all. 91.67% of actual pulsar cases were recognized correctly as pulsars, higher than 0% by the blind method. Furthermore, 97.41% of all test cases predicted as pulsars were real pulsars. Cumulative averages show that this algorithm provides 30.757% more accuracy than the blind approach.

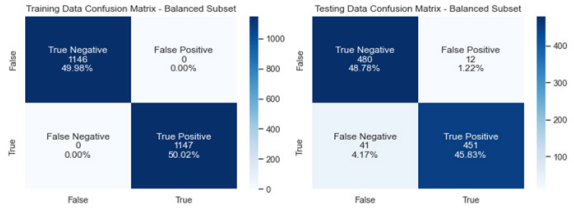

Figure 13: Confusion matrices resulting from balanced dataset achieved using random subset of noise sample balancing technique and optimal hyperparameters: n_estimators = 50, min_samples_leaf = 5, max_leaf_nodes = 150, criterion = 'gini.'

Decision tree 0 of the resulting random forest is depicted in Figure 14.

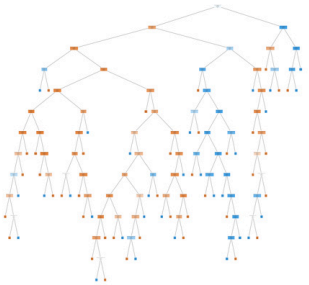

Figure 14: Full visualized decision tree 0 of the random forest.

Feature Importance:

As visualized in Figure 15, in ranking feature importance in differentiating between pulsars and non-pulsars on a scale from 0 to 1, the excess kurtosis of the integrated profile was ranked as the most important feature with a relative feature importance of 0.249734. Next was the skewness of the integrated profile at 0.193486, the mean of the integrated profile at 0.176090, the mean of the DM-SNR curve at 0.105585, the standard deviation of the DM-SNR curve at 0.099761, the skewness of the DM-SNR curve at 0.074150, and the excess kurtosis of the DM-SNR curve at 0.065950. Finally, the least important feature in determining whether a signal is from a true pulsar or a form of interference was the standard deviation of the integrated profile at 0.035243.

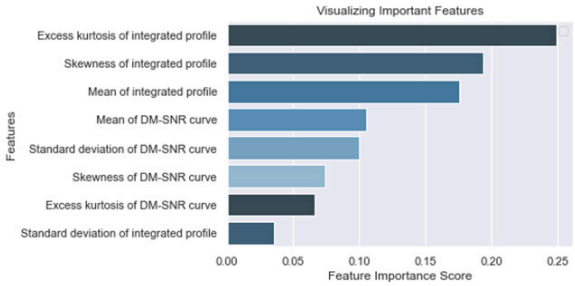

Figure 15: The ranked relative importance of radio pulse signal features.

Figure 16 provides an insight into how an individual tree creates splits to create the most significant separation between the pulsar and non-pulsar classes. The root node of this tree divided based on feature 4, the mean of the DM-SNR curve,

indicating that the mean of the DM-SNR curve was the most important feature out of the random subset of features used to build decision tree 0 in differentiating the pulsars from the non-pulsars in the random data points used to create that tree. Feature 5, the standard deviation of the DM-SNR curve, and Feature 3, the skewness of the integrated profile, were the following two most important features, as they were then used to split the second layer of the tree.

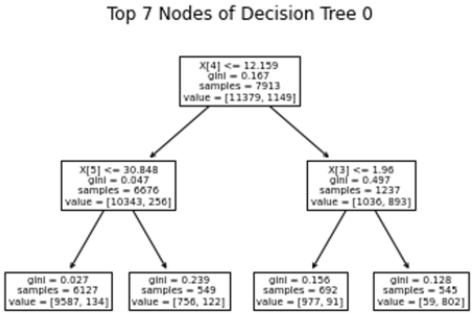

Figure 15: The ranked relative importance of radio pulse signal features.

The model is very robust and can produce results with similar accuracy even with fewer features. When removing the standard deviation of the integrated profile from the model, the accuracy remained very stable at 0.9512. When removing both the standard deviation of the integrated profile and the excess kurtosis of the DM-SNR curve from the model, the accuracy remained at 0.9471. With the three least important features, the standard deviation of the integrated profile, the excess kurtosis of the DM-SNR curve, and the skewness of the DM-SNR curve removed, the accuracy of the model stayed constant at 0.9502. This broadens the applications of this machine-learning model in pulsar surveys. The algorithm can be used with very high accuracies even in cases where not all eight features' values are recorded or known.

Conclusion

Our model demonstrates the effectiveness of random forest machine learning classifiers to solve the pulsar candidate selection problem. Our model was able to identify pulsars to an accuracy of 95% based on the mean, standard deviation, excess kurtosis, and skewness of the integrated pulse profile and DM-SNR curve of radio pulse signals from the South High Time Resolution Survey.

The optimal hyperparameters used to achieve this accuracy were n_estimators = 50, min_samples_leaf = 5, max_leaf_nodes = 150, and criterion = 'gini.'

Our results demonstrate that the excess kurtosis, skewness, and mean of the integrated profile were the three most important factors in differentiating between real pulsar signals and interference or other noise.

Limitations:

While this model has demonstrated high accuracy in classifying real pulsar signals from radio data, it is yet to be trained to identify millisecond pulsars (MSPs) specifically. Thus, this tool is likely not useful in searching for MSPs.

Additionally, since the real labeled pulsars in the training data came from the signals emitted by typical pulsars, this model would not be able to successfully identify or flag novel and unusual phenomena that might otherwise be detected through manual signal processing by human eyes. Thus, human processing should still be involved in the pulsar identification process at some level, and automated methods such as the one proposed here should only act as a first pass over the data.

Future Work:

This tool can be applied to a broad spectrum of future surveys, for example those conducted by the future Low-Frequency Array (LOFAR), the Five Hundred Meter Aperture Spherical Telescope (FAST), and the Square Kilometer Array (SKA), all of which are expected to produce vast amounts of signal data.[12] It can be used as a first pass over the recorded data to significantly reduce the number of possible pulsar candidates that require manual observation. In addition, the high accuracy of this classifier decreases the likelihood of missing promising pulsar candidates during the first pass over the data.

To continue to improve this model, it should be trained on larger labeled datasets in addition to the one provided by the HTRU2 survey.

To increase the likelihood of identifying MSP candidates, a separate machine learning classifier should be used that is trained on a dataset where MSPs are explicitly labeled. That will ensure that the unique characteristics of MSPs will be clearly identified by the machine learning tool.

Acknowledgments

I want to thank Kristen Surraro for her guidance on this project.

References

1. Calla Cofield. What Are Pulsars? https://www.space.com/32661-pulsars.html.
2. Ravenhall, D. G.; Pethick, C. J.; Wilson, J. R. Structure of Matter below Nuclear Saturation Density. Physical Review Letters 1983, 50 (26), 2066–2069. https://doi.org/10.1103/ physrevlett.50. 2066.
3. Lyon, R. J.; Stappers, B. W.; Cooper, S.; Brooke, J. M.; Knowles, J. D. Fifty Years of Pulsar Candidate Selection: From Simple Filters to a New Principled Real-Time Classification Approach. *Monthly Notices of the Royal Astronomical Society* 2016, 459 (1), 1104–1123. https://doi.org/10.1093/mnras/stw656.
4. Lyon, R. J. Why Are Pulsars Hard to Find?, University of Manchester, 2016.
5. IBM Cloud Education. What is Machine Learning? https://www.ibm.com/cloud/learn/machine-learning.
6. Train and Test datasets in Machine Learning - Javatpoint https://www.javatpoint.com/train-and-test-datasets-in-machine-learning.
7. Bento, C. Decision Tree Classifier explained in real-life: picking a vacation destination https://towardsdatascience.com/ decision-ree-classifier-explained-in-real-life-picking-a-vacation-destination-6226b2b60575.
8. Yiu, T. Understanding Random Forest https://towardsdatascience.com/understanding-random-forest-58381e0602d2.
9. IBM Cloud Education. What is Random Forest? https://www.ibm.com/cloud/learn/random-forest.
10. IBM Cloud Education. What are Neural Networks? https://www.ibm.com/cloud/learn/neural-networks.
11. Artificial Neural Network - Basic Concepts - Tutorialspoint https://www.tutorialspoint.com/artificial_neural_network/artificial_neural_network_basic_concepts.htm.
12. Bates, S. D.; Bailes, M.; Barsdell, B. R.; Bhat, N. D. R.; Burgay, M.; Burke-Spolaor, S.; Champion, D. J.; Coster, P.; D'Amico, N.; Jameson, A.; Johnston, S.; Keith, M. J.; Kramer, M.; Levin, L.; Lyne,

DOI: 10.36838/v5i4.3

A.; Milia, S.; Ng, C.; Nietner, C.; Possenti, A.; Stappers, B. The High Time Resolution Universe Pulsar Survey — VI. An Artificial Neural Network and Timing of 75 Pulsars. *Monthly Notices of the Royal Astronomical Society* 2012, 427, 1052–1065. https://doi.org/10.1111/j.1365-2966.2012.22042.x.
13. Eatough, R. P.; Molkenthin, N.; Kramer, M.; Noutsos, A.; Keith, M. J.; Stappers, B. W.; Lyne, A. G. Selection of Radio Pulsar Candidates Using Artificial Neural Networks. Monthly Notices of the Royal Astronomical Society 2010, 407 (4), 2443–2450. https://doi.org/10.1111/j.1365-2966.2010.17082.x.
14. Morello, V.; Barr, E. D.; Bailes, M.; Flynn, C. M.; Keane, E. F.;van Straten, W. SPINN: A Straightforward Machine Learning Solution to the Pulsar Candidate Selection Problem. *Monthly Notices of the Royal Astronomical Society* 2014, 443 (2), 1651–1662. https://doi.org/10.1093/mnras/stu1188.
15. UCI Machine Learning Repository: HTRU2 Data Set https://archive.ics.uci.edu/ml/datasets/HTRU2.
16. Radhakrishnan, V.; Vivekanand, M. The Structure of Integrated Pulse Profiles. *Journal of Astrophysics and Astronomy* 1980, 1, 119-128.
17. Pulsar Dispersion Measure | COSMOS https://astronomy.swin.edu.au/cosmos/P/Pulsar+Dispersion+Measure#:~:text=The%20dispersion%20measure%20can%20be%20determined%20by%20the.
18. CFI. Kurtosis - Definition, Excess Kurtosis, and Types of Kurtosis https://corporatefinanceinstitute.com/resources/knowledge/other/kurtosis/.
19. How is the kurtosis of a distribution related to the geometry of the density function? https://stats.stackexchange.com/questions/84158/how-is-the-kurtosis-of-a-distribution-related-to-the-geometry-of-the-density-fun.
20. 6.1: Qualitative Data and Quantitative Data https://math.libretexts.org/Courses/Mount_Royal_University/ MATH_1150:_Mathematical_Reasoning/6:_Introduction_to_Statistics/6.1:_Qualitative_Data_and_Quantitative_Data.
21. Aniththa. Hyperparameter tuning a model - Azure Machine Learning https://docs.microsoft.com/en-us/azure/machine-learning/how-to-tune-hyperparameters..
22. Baron, D. Machine Learning in Astronomy: A Practical Overview. arXiv:1904.07248 *[astro-ph]* 2019.
23. Google. Classification: ROC Curve and AUC | Machine Learning Crash Course https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc..
24. Scikit-learn. 3.2.4.3.1. sklearn.ensemble.RandomForestClassifier — scikit-learn 0.20.3 documentation https://scikit-learn.org/stable/modules/generated/sklearn.ensemble. RandomForestClassifier.html.
25. Gini Index for Decision Trees: Mechanism, Perfect & Imperfect Split With Examples https://www.upgrad.com/blog/gini-index-for-decision-trees/.
26. What Is Log Loss in Machine Learning? https://pandio.com/what-is-log-loss-in-machine-learning/#:~:text=Log%20loss%20applies%20to%20the%20prediction%20process%20in (accessed 2022 -08 -27).
27. https://kodzilla.pl; dev@kodzilla.pl. Addepto https://addepto.com/what-is-entropy-in-machine-learning/#:~:text=Entropy%20is%20frequently%20used%20in%20one%20of%20the (accessed 2022-08 -27).
28. Fraj, M. B. In Depth: Parameter tuning for Random Forest https://medium.com/all-things-ai/in-depth-parameter-tuning-for-random-forest-d67bb7e920d.

Author

Ankhita Sathanur is a senior at Eastlake High School. She plans to major in computer science and physics and pursue a career in computational astrophysics. She hopes to continue researching pulsars and their unique role in astronomy, and she plans to continue to implement computing solutions in the field of astrophysics.