Classification Algorithm Comparison for Pulsar Neutron Star Candidates

Kunho Kim ak26252@uga.edu

Abbie Thomas agt96353@uga.edu

Abstract

This paper compares classification algorithms for identifying pulsars from a set of pulsar candidates. Pulsars are neutron stars which offer a wealth of information for astronomers. The objective of this experiment is to compare two different pulsar classification algorithms: logistic regression and various support vector machines (SVMs). We aim to reach an accuracy of at least 95% for each algorithm. The data set, HTRU2 contains 17,898 instances of pulsar candidates with eight input attributes and one target attribute for each instance. The data will be preprocessed and optimized separately for the two classification algorithms. These algorithms will be analyzed for significance separately and compared based on the accuracy of the model. The results of this experiment concluded that the SVM using Gaussian RBF was the most accurate model with an accuracy of 99.85% at its most efficient. The other SVM models as well as the logistic regression model were also successful in classifying pulsar candidates with accuracies ranging from 88.44% to 98.30%.

1. Introduction

A pulsar is a type of neutron star which rotates quickly, is highly magnetized, and whose radio emissions are detectable on Earth (Lorimer & Kramer, 2012). According to Lorimer and Kramer, pulsars provide astronomers with information about neutron star physics, general relativity, the Galactic gravitational potential and magnetic field, the interstellar medium, celestial mechanics, planetary physics, and cosmology. Zaven Arzoumanian of NASA's Goddard Space Flight Center said, “You can time the pulsations of pulsars distributed in many directions around a spacecraft to figure out where the vehicle is and navigate it anywhere.” Needless to say, these rare stars provide a wealth of knowledge when they are discovered. There are only around 2,000 known pulsars (Hille, 2017) – which means that detecting and identifying them is a difficult and important task.

The objective of this experiment is to compare two pulsar classification algorithms: logistic regression and various support vector machines (SVMs). Each algorithm will be optimized according to the machine learning framework used and compared by accuracy. We aim to reach an accuracy of at least 95% for each algorithm.

The HTRU2 Data Set contains 17,898 pulsar candidates, which were collected during the High Time Resolution Universe Survey (Lyon, 2017). Of this data set, 1,639 are positively identified pulsars – about 9.157% of the data set – and 16,259 are not pulsars – about 90.843% of the data set. Each row represents a pulsar candidate and each candidate has eight attributes: mean of the integrated profile, standard deviation of the integrated profile, excess kurtosis of the integrated profile, skewness of the integrated profile, mean of the DM-SNR curve, standard deviation of the DM-SNR curve, excess kurtosis of the DM-SNR curve, a skewness of the DM-SNR curve. All eight attributes are continuous numerical variables. Each row also contains a target attribute, the class. The class is a nominal variable of either one or zero – one representing pulsars and zero representing non-pulsars.

We will create a logistic regression model, which will be evaluated based on accuracy as well as on the receiver operating characteristic (ROC) curve. This will be compared to various SVM models – linear, Gaussian RBF, polynomial, and sigmoid. The SVM models will be evaluated

based on accuracy as well. Once both models are optimized, they will be compared based on the accuracy of the logistic regression model and the accuracy of the highest performing SVM model.

We found that the SVM Gaussian RBF model had the highest accuracy of 98.464%. The Gaussian RBF model was more accurate than the other SVM models as well as the logistic regression model, which had an accuracy of 98.128%. We also found that preprocessing the data in which the training set contained an equal representation of class one and class zero produced an accuracy of about 50% for all the models except Gaussian RBF, which had an accuracy of 99.848%. We will discuss why these models performed as they did as well as future steps for experimentation.

2. Data – Preprocessing

As discussed in the introduction, this data set is quite unbalanced, with less than 10% of the data being the positive target in class one and more than 90% of the data being the negative target in class zero. In order to compensate for this, we tried several different strategies and got varying results. Additionally, the original data set doesn't include column labels, which caused problems in converting the data to a Pandas data frame. As a solution for this, we added the correct column labels as the header row. This step made it easier to access the columns of the data frames in the code in later stages of our experiment.

We decided to include all of the eight features in the data set. There were several motivations for this, but the main reasons were that including all eight features gave us the best results without sacrificing a large amount of time. More about this will be discussed in the future work and experiments section, but we currently believe that all eight of these features hold significance in predicting pulsars.

The linear regression model did not require significant preprocessing. We considered standardizing the data as we did with the SVM algorithm, but found that the accuracy actually decreased and it is widely accepted that standardization does not improve the success of logistic regression. We also ran recursive feature elimination to rank the attributes but found that each attribute was ranked equally according to the output. Based on this, we decided that any preprocessing for logistic regression was unnecessary or overall unsupportive of the model.

Since in SVM the algorithm is trying to maximize the distance between the separating hyperplane and the support vectors, the features cannot have widely varying and large values. Because this can result in a less accurate model, we decided to use scikit-learn's StandardScaler. By using this scaler, we're able to make each feature have a mean of zero and unit variance of one.

To train the model, we did a standard 80% training, 20% testing split of the data set (method one). More on this will be explained in the experiments section. However, we also tried different methods of splitting the data set before fitting the model. One such way was to split the current data set into two separate data sets: the rows of class zero and the rows of class one (method two). Since there is a significantly more instances of class zero than class one, the class zero data set was further reduced by choosing a random sample. This random sample's size is the same size as the class one data set so that the two can be combined without error. These two data sets were then each split using the 80/20 method mentioned earlier. The 80% splits were appended together to form the modified testing set and the 20% splits were appended to form the modified training set. The result of this preprocessing step was a training and testing set that had an even distribution of class one and class zero.

We speculated that partitioning the data set with equal instances of both classes would mitigate the impact of having an unbalanced data set. The results of these partitioned data sets will be discussed further in the analysis section.

3. Experiments

We compared the accuracy of two machine learning algorithms for classifying pulsars from the HTRU2 data set: logistic regression and various support vector machine (SVM) algorithms.

Logistic Regression is a machine learning framework in which the coefficients of a linear function are learned in order to predict the class of a particular instance. Since each attribute of the data set is a continuous numerical variable and the problem is binary classification, logistic regression is well suited for this problem.

Using Jupyter Notebook, the data was read in as a Pandas data frame from a csv file. Because the original data set did not include column labels in the csv file, the first instance in the data set was used as each column label. We created practical column names and added the first instance to the data frame. Next, we saved the first eight columns – the input attributes – into an array and the last column – the target attributes – into an array. There was not a significant amount of preprocessing required for logistic regression. Using scikit-learn's recursive feature elimination (RFE), we determined that each input attribute was performing equally, with each having an RFE ranking of one. Additionally, we did not standardize the data set to a mean of zero and standard deviation of one for logistic regression. Though this was helpful for the SVM model, it is commonly accepted that standardization does not improve the accuracy of logistic regression models. We ran the standardized data through this model and found that the accuracy was lower than the unstandardized data.

The data was split so that 20% of the data was used for training and 80% was used for testing using scikit-learn's train_test_split from model_selection. This resulted in four NumPy ndarrays: input training, input testing, target training, and target testing. Now that the data was properly split into these four arrays, we used scikit-learn's Logistic Regression, with all default values, to fit input training and target training into a model.

Next, we looked at Support Vector Machines, which are commonly used for classification problems as well. They use hyperplanes to separate data points and formulate classes. The model will choose a hyperplane (or decision boundary) that maximizes the distance to the closest training data point of each class. Greater distance usually indicates lower generalization error in the classifier. To achieve more accurate hyperplanes in SVM, kernel functions may be used. These kernel functions allow the hyperplane to more accurately classify each data point. Depending on the data set and objective, certain kernel functions will perform better over others.

Since the basic SVM "one vs rest" model is great for binary classification[1], we decided to choose this decision shape. Additionally, we compared several different SVM algorithms with different kernel functions. We looked at the main four functions: linear, Gaussian RBF, polynomial, and sigmoid. Figure 1 shows an example of an SVM that uses a kernel trick. If it was using a linear function, the separating hyperplanes would be linear.

Scikit-learn's SVM package also allows the user to change the values for gamma and C. Simply put, C is used to control the error of the model and gamma is a parameter that gives more

[1] Both "one vs one" (OVO) and "one vs rest" (OVR) will achieve the same result in binary classification problems

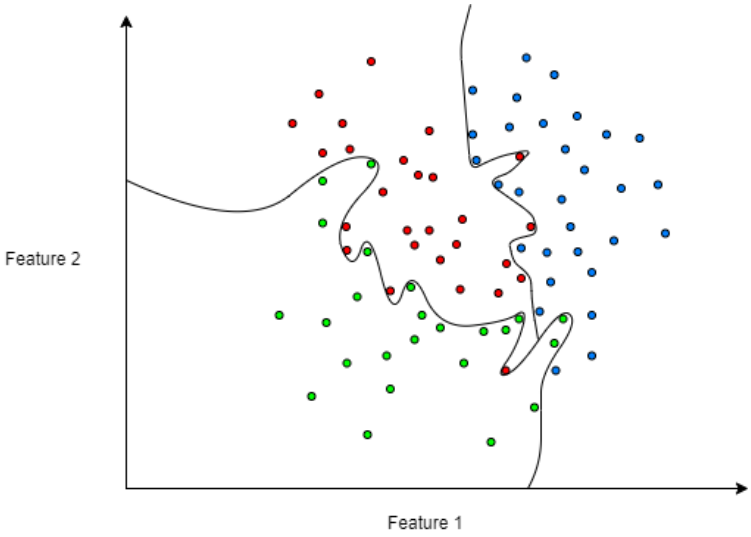

Figure 1: SVM using a kernel trick

curve and weight to the decision boundary. Usually these are powers of 10. The default values of the scikit-learn model are "C=1, gamma=1." Upon further experimentation, we found that these default values were the best performing in both timing and accuracy.

Using these parameters, we were able to build and fit each model for linear, Gaussian RBF, polynomial, and sigmoid SVM. The results of these models with the different preprocessing methods will be discussed in the next section.

4. Analysis

The logistic regression model produced coefficients [2.53108844e-02, -3.41166021e-02, 6.22899684e+00, -5.82307918e-01, -3.05535320e-02, 5.24445612e-02, -1.09857896e-02, -2.61182933e-03] and intercept [-8.07326827]. This model had an accuracy of 0.98128. Since the accuracy of this model was so high, we determined that this model was significant. We also decided to calculate the ROC curve for the logistic regression model using scikit-learn. The dotted curve represents a purely random classifier – a good classifier will be far from this curve. As seen in Figure 2, the logistic regression model we created for classifying pulsars stays very far from the dotted line, indicating that it is a significant model. Additionally, scikit-learn's confusion matrix using the predicted values and the actual values indicates that for non-pulsars, the algorithm predicted 3248 correctly and 52 incorrectly. For pulsars, the algorithm predicted 264 correctly and 16 incorrectly.

The "method one" of preprocessing for SVM model produced an accuracy of 98.296% for the linear model, 98.464% for the Gaussian RBF model, 98.128% for the polynomial model, and 88.436% for the Sigmoid model. The linear, RBF, and polynomial models all performed similarly with RBF performing the best at 98.46% accuracy. Sigmoid performed the worst at 88.43% accuracy. The "method two" of preprocessing for SVM model produced an accuracy of 54.878% for the linear model, 99.848% for the Gaussian RBF model, 72.561% for the polynomial model, and 42.378% for the Sigmoid model. Here we have an interesting shift. The linear and sigmoid models both perform significantly worse, with the polynomial model also performing worse but to a smaller degree. However, the Gaussian RBF model actually performs better with a near perfect accuracy of 99.85%.

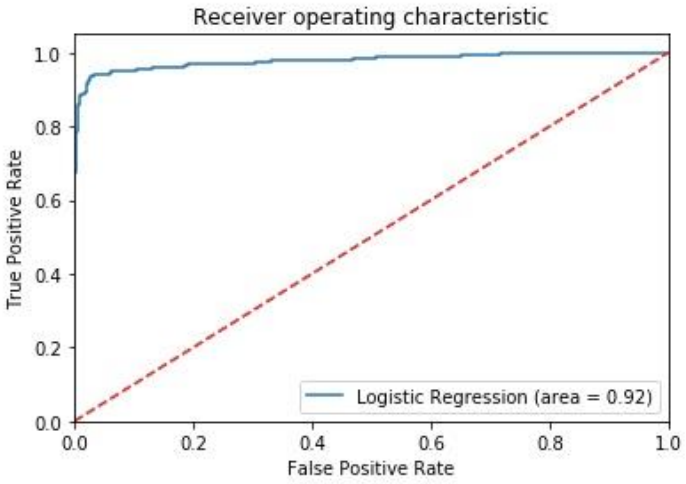

Figure 2: ROC curve for logistic regression model.

The "method one" of preprocessing for SVM model produced an accuracy of 98.296% for the linear model, 98.464% for the Gaussian RBF model, 98.128% for the polynomial model, and 88.436% for the Sigmoid model. The linear, RBF, and polynomial models all performed similarly with RBF performing the best at 98.46% accuracy. Sigmoid performed the worst at 88.43% accuracy. The "method two" of preprocessing for SVM model produced an accuracy of 54.878% for the linear model, 99.848% for the Gaussian RBF model, 72.561% for the polynomial model, and 42.378% for the Sigmoid model. Here we have an interesting shift. The linear and sigmoid models both perform significantly worse, with the polynomial model also performing worse but to a smaller degree. However, the Gaussian RBF model actually performs better with a near perfect accuracy of 99.85%.

In both preprocessing methods, the Gaussian RBF model was the highest performing algorithm. This may be because our model is not linearly separable, especially when the data set is balanced by using method 2 of preprocessing. As shown in Figure 3, if our data looks like the data in "Radial basis," then all other algorithms are likely to perform very poorly.

However, more research about the intricacies of these kernel functions and experimentation with other preprocessing methods and hyperparameters are needed in order to fully understand why the Gaussian RBF kernel model performed the best with our data set.

5. Future Work

If we decide to continue with this project in the future, there are a few things may help us better understand our results. Scikit-learn and matplotlib allow users to plot decision boundaries for SVM algorithms. However, since our dataset had more than two features, it was difficult figuring out how to do this in multiple dimensions. We hope to find a way to do this to better understand how the data points are being split.

Also, it may be interesting to see how other kernel functions of SVM perform with this dataset. We used the four most-used kernel functions, but there may be a more specialized function that suits our project better. Additionally, we may need to conduct further research to understand why the Gaussian RBF model performed the best in our experiments.

We may also want to try new training and preprocessing methods. It may be interesting to see how cross validation and feature weighting change our model.

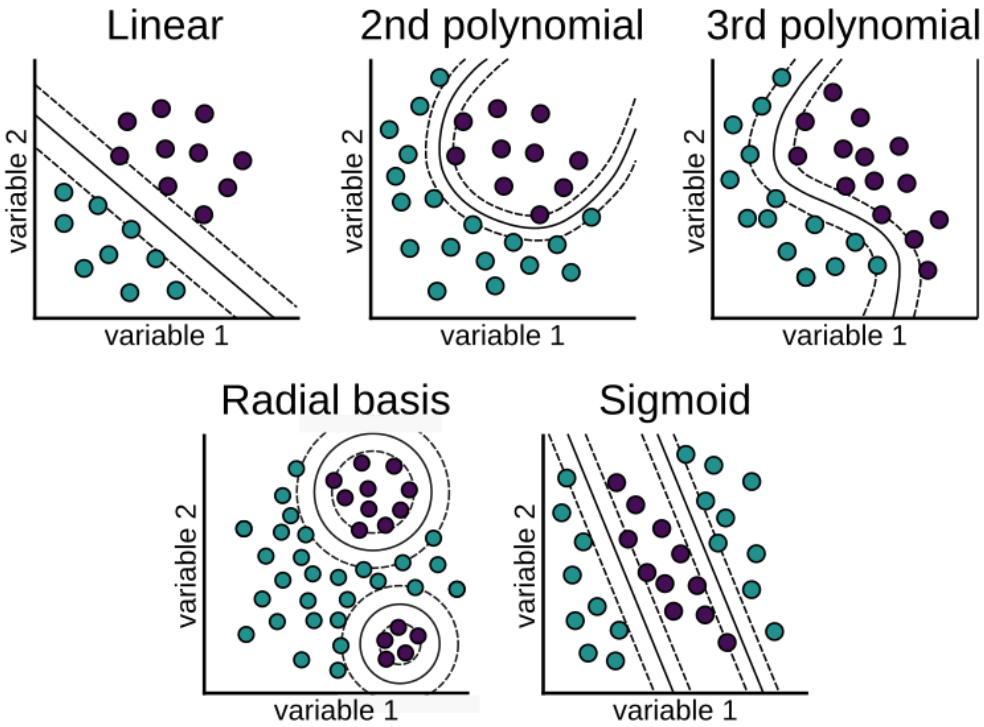

Figure 3: Comparing SVM kernel functions graphically (Rhys, 2019)

Lastly, we may need to spend some time to make sure that our model is not overfitting results. To do this, we would need to research more about the "overfitting traps" of each of our models.

6. Future Work

If we decide to continue with this project in the future, there are a few things may help us better understand our results. Scikit-learn and matplotlib allow users to plot decision boundaries for SVM algorithms. However, since our dataset had more than two features, it was difficult figuring out how to do this in multiple dimensions. We hope to find a way to do this to better understand how the data points are being split.

Also, it may be interesting to see how other kernel functions of SVM perform with this dataset. We used the four most-used kernel functions, but there may be a more specialized function that suits our project better. Additionally, we may need to conduct further research to understand why the Gaussian RBF model performed the best in our experiments.

We may also want to try new training and preprocessing methods. It may be interesting to see how cross validation and feature weighting change our model.

Lastly, we may need to spend some time to make sure that our model is not overfitting results. To do this, we would need to research more about the "overfitting traps" of each of our models.

7. Conclusion

In the end, the Gaussian RBF SVM model performed the best with our "method two" of preprocessing at 99.848% accuracy. Many other models performed similarly well at greater than 98%, which was promising to see. With our current work and potential future endeavors in this

project, we hope that our findings can help astronomers in using data and machine learning to identify and classify galactic objects, such as pulsars.

References

Lorimer, D. R., & Kramer, M. (2012). Handbook of pulsar astronomy. Cambridge: University Press.

Hille, K. (2017, August 1). NASA Continues to Study Pulsars, 50 Years After Their Chance Discovery. NASA. https://www.nasa.gov/feature/goddard/2017/nasa-continues-to-study-pulsars-50-years-after-their-chance-discovery.

Lyon, R. J., Cooper S., Brooke J. M., Knowles J. D., Stappers B. W. (2016, March 16). Fifty years of pulsar candidate selection: from simple filters to a new principled real-time classification approach, Monthly Notices of the Royal Astronomical Society, Volume 459, Issue 1, 11 June 2016, Pages 1104–1123, https://doi.org/10.1093/mnras/stw656

Lyon, R. J. (2017) HTRU2, DOI: 10.6084/m9.figshare.3080389.v1.

Rhys, H. (2019, October 10). Support Vector Machines with the mlr package. Machine learning with R, tidyverse, and mlr. https://machinelearningwithmlr.wordpress.com/2019/10/10/example-post/.