SVM Classification for Pulsar Stars

John Patrick Wisnewski - ppwxmc@umsystem.edu
University of Missouri

***Abstract*—Pulsars are very interesting and important celestial bodies. They can provide researchers with an extraordinary amount of valuable data. However, they are very difficult to find and verify. The signals that are emitted from pulsars are very weak and get overshadowed by cosmic background radiation. In order to find them it takes a lot of time and effort to collect enough data to verify them. This paper will look at how limiting the amount of features used to identify pulsars could lead to quicker and more efficient results with the same degree of accuracy. This process will be done by implementing a supervised SVM machine learning algorithm in Python to classify the samples, and a radial basis function to optimize hyperparameters. Then a random forest classifier will be used for feature selection to find the two most important features. Once these hyperparameters are selected the results from the original dataset will be compared with the two feature dataset.**

I. System Information

This program will be run on Jupyter Notebook and primarily use the sklearn library. The system specifications are a Dell Latitude e5250 laptop. This machine uses an Intel i5-5300u CPU at 2.3ghz with two cores and four logical processors. 8gb of ram is installed on this machine.

II. Introduction

Pulsars are neutron stars that emit beams of light in opposite directions. Neutron stars are incredibly dense and the only thing more dense are black holes. So this makes pulsars one of the most dense celestial bodies we can study. Pulsars are rapidly spinning at no less than one spin per second and emitting beams of light energy in opposite directions. Due to their incredible mass and fast spin rate they generate extremely large and powerful magnetic fields. Pulsars also distort the fabric of spacetime around themselves due to their high density. It is for these reasons they are so sought after in the astrophysics field. Their unique qualities can give researchers insight on phenomena like gravitational waves and extreme states of matter. This can lead to advancements in the understanding of how the universe works and began. To obtain this information is no small feat. In the last fifty years roughly 2,0000 pulsars have been detected and verified [1]. Out of the hundreds of millions of stars that have been cataloged this is a very small figure. With new advancements in telescopes the amount of data that can be collected will continue to increase. There is no doubt more pulsars are yet to be discovered but finding them will take a considerable amount of time and effort. In order to efficiently go through this data it could prove beneficial to exclude features that are not important, while accuracy is maintained. This paper will look to implement a supervised support vector machine with soft margins to classify data points, the radial basis function to improve hyperparameters c and gamma to optimize results, and a random forest classifier to identify the most important features in the dataset. Then the results from using all features in the data set opposed to only using the two most important features will be compared.

III. Problem definition

The main problems this paper is going to address is how to improve efficiency by decreasing time spent classifying and optimizing, decrease the amount of data used, and maintain or improve accuracy. This paper is not attempting to redefine the support vector machine method or make any substantial alterations to it. The dataset that is being used in this paper is from the University of California Irvine and contains

data collected over the last fifty years. Without optimization a supervised SVM method can effectively identify pulsars but it is time and consuming and uses over 140,000 data points. The proposed solution is to classify the data with all features considered, then identify the two most important features, rerun the classification and optimization and compare the results.By doing this it could improve efficiency and reduce time spent. By only using the two most important features this will drastically reduce the sample size from 143,176 data points to 35,796. If the accuracy remains consistent in both results then that would mean the pulsars were identified using less data, less time, and with the same accuracy.

IV. Methods

In order to classify the data this paper will be using a supervised support vector machine method for classification. Supervised SVM will classify the data by finding an optimal hyperplane or decision boundary to separate the data.

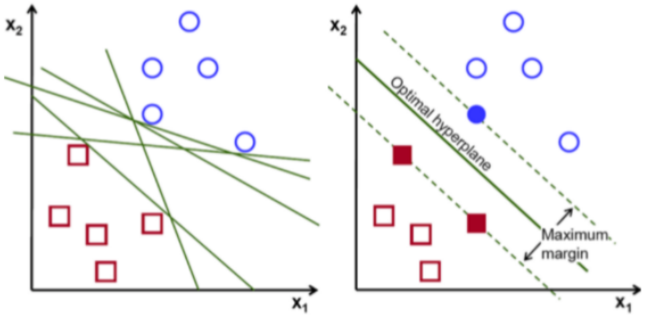


Figure 1. Example of SVM Classification [2]

In this experiment a soft margin SVM will be implemented. Soft margin means that some points are allowed to be misclassified. A soft margin is being used because this data is not linearly separable and there are a large amount of samples. If a hard margin were to be used this could lead to overfitting issues [3]. Once the data is classified a radial basis function will be implemented to find the optimal c and gamma values. These are our hyperparameters that are optimized to increase the accuracy of our classification. The value c represents how accurate we want our test data to be. Gamma represents the weight of each individual sample on the overall dataset. These parameters are found by going through the classification and changing the values of c and gamma to 1, 0.1, 0.001, and 0.0001. This means there are five different values for each parameter, totaling 125 different combinations of these values. Then whichever combination produces the highest results is selected. This is done with the kernel trick using the radial basis function. The kernel trick is defined as “a function that takes as its inputs vectors in the original vector space and returns the dot product of the vectors in the feature space” [4]. The radial basis function is a variation of a kernel and is used to classify points that are extremely close to one another. After the optimal c and gamma values are found and the RBF kernel is complete the supervised SVM method will be rerun using those new values for the hyperparameters. The results will then be compared to the results without optimization. When classification is complete for the original data set a random forest classifier will be used to identify the two most important data points. The random forest classifier or RFC in the sklearn library in python works by “selecting a random sample from the training set, creating a decision tree for it and gets a prediction; it repeats this operation for the assigned number of the trees, performs a vote for each prediction, and takes the result with the majority of votes”[5] Once these features are identified the two most important ones will be selected for the other half of the comparison. The same process will be repeated and then the results

will be compared. The desired results are to use less time with two features opposed to all features, and to maintain accuracy across both implementations. In order to compare the results the amount of time it takes to optimize the hyperparameters will be compared along with the precision, recall, and F-score's of the final classification reports. This whole process will be done in Jupyter Notebook and will primarily use the sklearn library.

V. Experimental Results

The data split for both the eight feature and two feature implementation training, test, and validation datasets is shown below.

```
X_train, X_test, y_train, y_test = train_test_split(data_noclass, data_class, test_size=0.25, random_state= 3)
```

```
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=.25, random_state= 3)
```

In order to remain consistent the same test, train, and validation splits were used along with the same random state. The random state is a value that determines how the data is chosen for the data sets. By using the same random state we ensure the samples are consistent and results can be compared fairly. The initial data set has eight features with 17,898 samples. The optimization for the training and validation data is shown below.

```
Wall time: 14min 34s
```

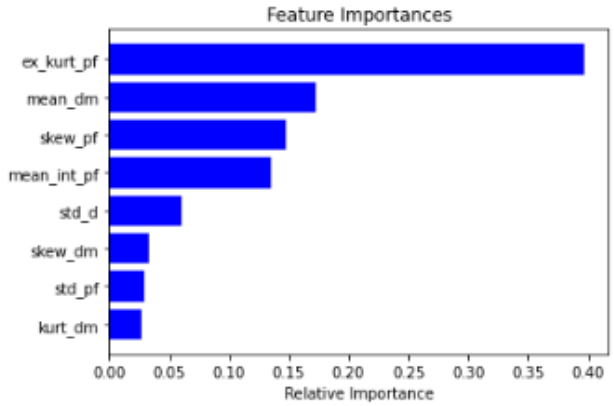
```
Wall time: 1min 25s
```

It took 14 minutes and 34 seconds to optimize the training dataset's c and gamma values and 1 minute and 25 seconds to optimize the validation dataset's c and gamma values . The final precision, recall and F-scores for the training data is shown below.

	precision	recall	f1-score	support
0	0.98	1.00	0.99	4070
1	0.95	0.80	0.87	405
accuracy			0.98	4475
macro avg	0.97	0.90	0.93	4475
weighted avg	0.98	0.98	0.98	4475

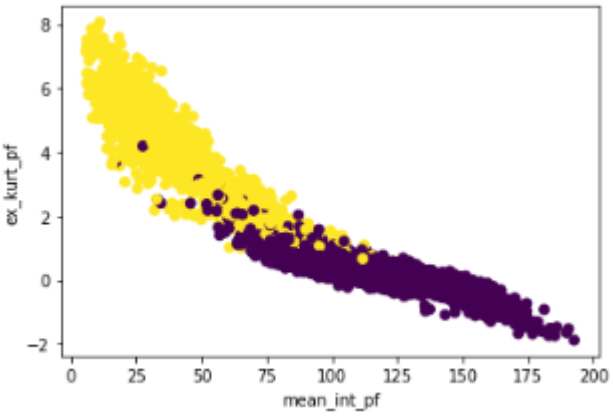
	precision	recall	f1-score	support
0	0.99	1.00	0.99	3038
1	0.96	0.87	0.91	318
accuracy			0.98	3356
macro avg	0.97	0.93	0.95	3356
weighted avg	0.98	0.98	0.98	3356

As you can see the results are very good, and there is a high F-score for both classes and remains this way for the validation data. The F-score increases from the training set to the validation set after optimizing as well. The figure below shows the importance of features from the training data set.



The top two features 'ex_kurt_pf' and 'mean_dm' have the most importance in this dataset. These two features are the ones selected for the other implementation. The figure below shows the initial

separation of the classes.



This shows that there is a lot of overlap and an imbalance in the data. This imbalance is expected and is going to be found in most data sets pertaining to pulsar stars. This type of imbalance is always why we compute the F-score with a macro average. The macro average computes each class equally where the micro computes each sample equally. Usually for data that is not balanced it is common to use the micro F-score, but for this dataset, the pulsars are rare but very valuable and the classification for positive pulsar stars should be treated as equal. For the two feature implementations the optimization for c and gamma for both the training and validation sets are below.

```
Wall time: 6min 43s
```

```
Wall time: 25.2 s
```

The time to find optimization values of c and gamma for the two feature implementations are considerably lower, but this is expected due to using over 100,000 less data points. So these results are consistent with the hypothesis. Now we will look at the accuracy for the training and validation data sets of the two feature implementation.

	precision	recall	f1-score	support
0	0.98	0.99	0.99	4070
1	0.92	0.79	0.85	405
accuracy			0.97	4475
macro avg	0.95	0.89	0.92	4475
weighted avg	0.97	0.97	0.97	4475

	precision	recall	f1-score	support
0	0.98	1.00	0.99	3038
1	0.96	0.80	0.87	318
accuracy			0.98	3356
macro avg	0.97	0.90	0.93	3356
weighted avg	0.98	0.98	0.98	3356

The F-score for the two feature validation is slightly lower than that of the eight feature validation set. The figure below is a side by side comparison with the eight feature validation data on the right and the two feature validation on the left.

	precision	recall	f1-score	precision	recall	f1-score	support
0	0.99	1.00	0.99	0.98	1.00	0.99	3038
1	0.96	0.87	0.91	0.96	0.80	0.87	318
accuracy			0.98			0.98	3356
macro avg	0.97	0.93	0.95	0.97	0.90	0.93	3356
weighted avg	0.98	0.98	0.98	0.98	0.98	0.98	3356

The results are very similar and it is hard to discern who truly has the better results.
The F-score is calculated by the equation below.
F1 = 2 * (precision * recall) / (precision + recall) [6].
Precision and recall are defined as “precision is a measure of result relevancy, while recall is a measure of how many truly relevant results are returned”[7].

VI. Conclusion

In conclusion there seems to be no substantial difference in the accuracy of the eight feature and two feature implementation. There is a slight decrease in accuracy when going from eight features to two features that should be noted. The time to optimize is drastically lower in the two feature

implementation opposed to eight features which goes with what is expected due to the amount of data in each implementation. It is hard to say definitively which one is optimal due to the massive data and time decrease from eight features to two with only a slight decrease in accuracy. However in this case it would be more beneficial to identify more pulsars then there actually are because there are so few. Then they can be reviewed again and determined if they are true positives or not. It would not be beneficial to misclassify true pulsar stars because then they would be lost in the massive size of the true negatives. This is similar to a security system having more false negatives than false positives. It would be a better system if it kept more people out rather than let more people in. In conclusion, the results show promising attributes to being more optimal, but it is too close to definitively say which implementation is better. It is clear that the two feature implementation takes significantly less time, uses significantly less data, and has only slightly less accurate results.

VII. Future Works

To build upon these results there are a few options that could be explored. For example instead of a two feature implementation another feature could be added. This will increase time and data but it could also increase the accuracy to make it more comparable to the eight feature accuracy. The time and data for a three feature would still be less than eight features, but the accuracy could be higher than two features. Another change that could be made is to examine results using different kernel functions. For example instead of using a radial basis function a linear function could be used. It is unclear how this would impact the results without testing on it. By implementing some changes, there might be a way to more definitively show that you can classify this data set with less data, less time, and equal accuracy.

REFERENCES

1. C. Cofield, “What are pulsars?,” *Space.com*, 22-Apr-2016. [Online]. Available: https://www.space.com/32661-pulsars.html. [Accessed: 11-May-2022].
2. P. P. Ippolito, “SVM: Feature selection and Kernels,” *Medium*, 04-Sep-2021. [Online]. Available: https://towardsdatascience.com/svm-feature-selection-and-kernels-840781cc1a6c. [Accessed: 11-May-2022].
3. “SVM: What is SVM: Support Vector Machine: SVM in python,” *Analytics Vidhya*, 26-Apr-2021. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/04/insight-into-svm-support-vector-machine-along-with-code/#:~:text=Hard%20margin%20SVM%20does%20not,SVM%20comes%20to%20the%20rescue. [Accessed: 11-May-2022].
4. D. Wilimitis, “The kernel trick,” *Medium*, 21-Feb-2019. [Online]. Available: https://towardsdatascience.com/the-kernel-trick-c98cdbcaeb3f. [Accessed: 11-May-2022].
5. “Sklearn Random Forest classifiers in python tutorial,” *DataCamp*. [Online]. Available: https://www.datacamp.com/tutorial/random-forests-classifier-python. [Accessed: 11-May-2022].
6. “Sklearn.metrics.f1_score,” *scikit*. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html. [Accessed: 11-May-2022].
7. “Precision-recall,” *scikit*. [Online]. Available: https://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html#:~:text=Precision%2DRecall%20is%20a%20u

seful,truly%20relevant%20results%20are%20returned. [Accessed: 11-May-2022].
8. “Core.ac.uk.” [Online]. Available: https://core.ac.uk/download/pdf/53141335.pdf. [Accessed: 12-May-2022].
9. “SVM Classifier,” *Pulsarclassifier*. [Online]. Available: https://as595.github.io/tutorials/PulsarClassifier.html. [Accessed: 11-May-2022].