Predicting if a Neutron Star is a Pulsar using Binary Classification

Rishikesh Kulkarni 19173

3rd Year BS-MS Physics Department

Abstract

Neutron Stars are celestial bodies mainly made up of neutrons, and they have a very high density and mass. There are different categories of neutron stars, out of which one is Pulsar. A neutron star may or may not be Pulsar depending on the conditions. Therefore, our data will have a binary output: Yes(1) or No(0). In such cases, Binary Classification is used to solve the problem. Binary Classification is a powerful method in Machine Learning to predict the result with high accuracy. It uses the attributes given and indicates the result in binary form.

Objective

The data I have used contains eight attributes, and the ninth one is the target class. Our objective is to make a model which extracts features from the data, makes algorithms, and predicts the output. I have used the classification algorithms due to the discrete nature of the target class to create a which will return us the supposed result.

Binary Classification

The classification model is suitable for our problem. Our target class has two kinds of results: 1 and 0. Due to this discrete binary nature of the target class, I use Binary Classification, which is dominantly used for these kinds of problems.

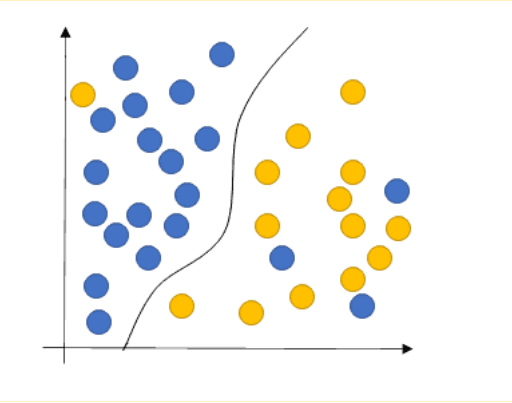

Figure 1: Binary Classification

Code Structure

The code begins with loading the relevant libraries. `Numpy`, `Pandas` and `Matplotlib` are the usual libraries that are uploaded at the beginning of the code. `Sklearn` and its required classes are imported when needed. `Seaborn` is also used at some places to enhance the visualization. Later, we uploaded our data which was taken from the Kaggle website. The data is a `csv` file, which is presented in Data Frame.

	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	Skewness of the DM-SNR curve	target_class
0	140.562500	55.683782	-0.234571	-0.699648	3.199833	19.110426	7.975532	74.242225	0
1	102.507812	58.882430	0.465318	-0.515088	1.677258	14.860146	10.576487	127.393580	0
2	103.015625	39.341649	0.323328	1.051164	3.121237	21.744669	7.735822	63.171909	0
3	136.750000	57.178449	-0.068415	-0.636238	3.642977	20.959280	6.896499	53.593661	0
4	88.726562	40.672225	0.600866	1.123492	1.178930	11.468720	14.269573	252.567306	0

Figure 2: First few entries of the data

Then, the statistical summary of the data is displayed to get a general idea of the data. The number of zeroes and ones are counted, and the data type is evaluated. The data contains `float64` type of numbers in the attributes and `int64` for the target class. As seen in the output of the code, there are no null values in the complete data. The count, mean, standard deviation, and the other statistical summary are also shown.
Then we proceed to find the correlation of data. The correlation of attributes with themselves is studied using the *Pearson's method*.

Data Visualization

We then move further on the visualization of the data. The count of zeroes and ones is displayed using `Matplotlib` library. This visualization helps us understand the data trends more visually using the different plots.

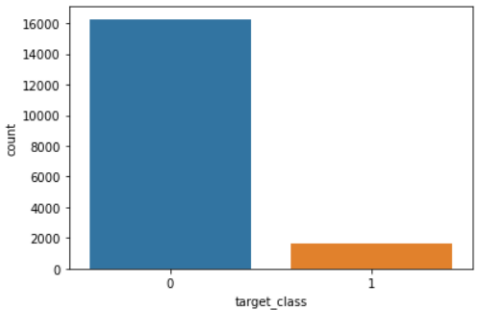

Figure 3: Number of zeroes and ones

The histogram, density plots, and box plots are displayed. The correlation we evaluated is visually presented using a color bar plot whose different color shade represents the amount of correlation one attribute has on another.

Data Transformation

The data I am working on has positive, negative, and minimal numbers. So, it becomes necessary to transform our data into a different class that does not have any complications. Therefore, the data is converted into data that lies in the range 0 to 1. The library used `sklearn.preprocessing` and from it `MinMaxScaler` which fits the data in the range.

Feature Selection

The next step is feature selection. For this, `SelectKBest` and `chi2` is imported from `sklearn.feature_selection`. The SelectKBest class scores the features using a function and then removes all the k highest scoring features. The chi-squared distribution is used in the common chi-squared tests for the goodness of fit of an observed distribution to a theoretical one. The top features are *Excess kurtosis of the integrated profile*, *Skewness of the integrated profile*, *Mean of the DM-SNR curve* and *Standard deviation of the DM-SNR curve*.

Evaluating Algorithms and Predictions

This is the most important part of the problem. The different algorithms used here are *Logistic Regression Model*, *K-Nearest Neighbor*, *Decision Trees*, *Random Forest Classifier*, *Support Vector Machines* and *Naive Bayes*.
First, featured data is split into *test* and *train* set. The test size I have used is 0.25. And the random state is 0. The class `KFold` and `cross_val_score` from `sklearn.model_selection` which will be used to estimate the skill of the model I have created.
The accuracy score using the cross-validation score is then measured. A loop intakes the models and gives the corresponding accuracy scores. The highest accuracy score is for the K Neighbors classifier model and Random Forest Classifier.

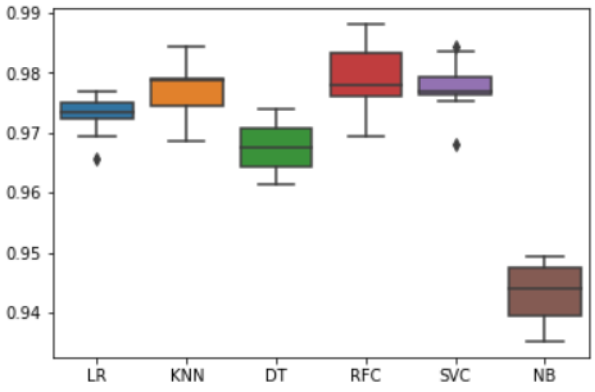

Figure 4: Accuracy scores for various algorithms

Accuracy score and Confusion matrix

The last part of this problem is our model's predictions and accuracy. The accuracy score is computed for the K Neighbors classifier model and Random Forest Classifier. The library used is `sklearn.metrics` which is exclusively used for the predictions. The *accuracy_score* measures the subset accuracy: the set of labels predicted for a sample must exactly match the corresponding set of labels in y_test. Then the *classification_report* measures the quality of predictions from the two classification algorithms. It contains *precision, recall, f1-score* and *support*. Finally, I create the *Confusion matrix* which is used to compute the accuracy of the machine learning algorithm in classifying the data into its corresponding labels.

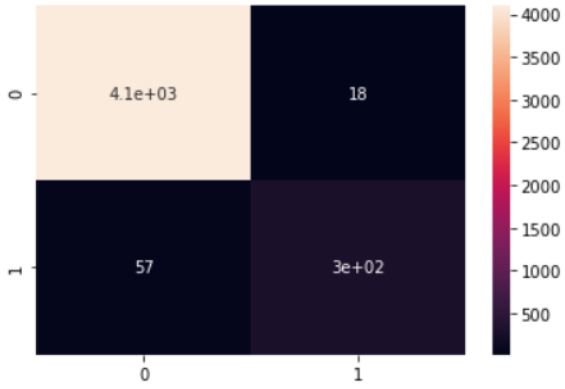

Figure 5: Confusion Matrix

Conclusion

The objective of this problem is fulfilled, and we have successfully created a model with the help of Binary Classification to predict the outputs of a dataset that contains binary outcomes. Different algorithms give different efficiencies, and the best model can be thus chosen.

References

[1] Data Set - https://www.kaggle.com/spacemod/pulsar-dataset
[2] Binary Classification - https://en.wikipedia.org/wiki/Binary_classification
[3] Paper - https://arxiv.org/abs/2002.08519
[4] Another paper - https://ieeexplore.ieee.org/document/9001517
[5] Informative article - https://deepchecks.com/glossary/binary-classification/