Pulsar Stars Predictions

Prathyusha Velupula

November 30, 2023

Contents

List of Figures

1 Abstract

HTRU2 is a data set which describes a sample of pulsar candidates collected during the High Time Resolution Universe Survey (South). Pulsars are a rare type of Neutron star that produce radio emission detectable here on Earth. They are of considerable scientific interest as probes of space-time, the inter-stellar medium, and states of matter . As pulsars rotate, their emission beam sweeps across the sky, and when this crosses our line of sight, produces a detectable pattern of broadband radio emission. As pulsars rotate rapidly, this pattern repeats periodically. Thus pulsar search involves looking for periodic radio signals with large radio telescopes.
Each pulsar produces a slightly different emission pattern, which varies slightly with each rotation . Thus a potential signal detection known as a 'candidate', is averaged over many rotations of the pulsar, as determined by the length of an observation. In the absence of additional info, each candidate could potentially describe a real pulsar. However in practice almost all detections are caused by radio frequency interference (RFI) and noise, making legitimate signals hard to find.
Machine learning tools are now being used to automatically label pulsar candidates to facilitate rapid analysis. Classification systems in particular are being widely adopted, which treat the candidate data sets as binary classification problems.
Here the legitimate pulsar examples are a minority positive class, and spurious examples the majority negative class. At present multi-class labels are unavailable, given the costs associated with data annotation.

2 Motivation

Given how difficult and time-consuming it can be to identify individual celestial objects, artificial intelligence may be the perfect instrument for exploring the universe. In light of this, the goal of the project is to create supervised learning algorithms that use binary classification to forecast pulsar classifications. A machine learning method called binary classification establishes whether an input is true or false, or 1 or 0.

3 Dataset

Link for the dataset https://archive.ics.uci.edu/dataset/372/htru2
A group of researchers from the University of Manchester, UK, examined information from the High Time Resolution Universe Survey (HTRU) to create the data set. The goal of the HTRU project is to find new pulsars and investigate their characteristics.
The data set includes statistical measures of the DMSNR curve and profile of the pulsar candidates. The DMSNR curve plots the dispersion measure against the signal-to-noise ratio, whereas the profile plots the pulsar's radio signal intensity as a function of time. The signal-to-noise ratio measures the strength of the radio signal in relation to the background noise, whereas the dispersion measure measures how much the pulsar's radio waves are dispersed as they travel through the interstellar medium.
The HTRU2 data set contains a total of eight input features, which are described below:

1. **Mean of the integrated profile:** This feature is the mean value of the integrated profile, which is a plot of the intensity of the radio signal from the pulsar as a function of time. It represents the overall shape of the pulsar candidate's signal.

2. **Standard deviation of the integrated profile:** This feature is the standard deviation of the integrated profile, which measures the spread or variability of the signal shape.

3. **Excess kurtosis of the integrated profile:** This feature is a measure of the "peakedness" of the integrated profile, compared to a normal distribution. Positive values indicate a more peaked profile, while negative values indicate a more flat-topped profile.

4. **Skewness of the integrated profile:** This feature is a measure of the symmetry of the integrated profile. A value of 0 indicates a symmetric profile, while positive and negative values indicate a skewed profile.

5. **Mean of the DM-SNR curve:** This feature is the mean value of the DM-SNR curve, which is a plot of the dispersion measure versus the signal-to-noise ratio. It represents the overall shape of the pulsar candidate's DM-SNR curve.

6. **Standard deviation of the DM-SNR curve:** This feature is the standard deviation of the DM-SNR curve, which measures the spread or variability of the DM-SNR curve.

7. **Excess kurtosis of the DM-SNR curve:** This feature is a measure of the "peakedness" of the DMSNR curve, compared to a normal distribution. Positive values indicate a more peaked curve, while negative values indicate a more flat-topped curve.

8. **Skewness of the DM-SNR curve:** This feature is a measure of the symmetry of the DM-SNR curve. A value of 0 indicates a symmetric curve, while positive and negative values indicate a skewed curve.

Output feature/ classification label: The output in the HTRU2 data set is the classification label,which is the target variable for the binary classification task. The label is a binary variable with two possible values:
1. Class 0: This label indicates that the candidate pulsar is not a pulsar. It means that the candidate is a noise signal or a signal from a source other than a pulsar.
2. Class 1: This label indicates that the candidate pulsar is a pulsar. It means that the candidate

is a signal from an astronomical object that emits regular pulses of radio waves.

Accurately predicting each pulsar candidate's classification label based on statistical features is the aim of a machine learning model trained on the HTRU2 data set. In order to complete this binary classification task, the model needs to be able to discriminate between pulsars and non-pulsars. By contrasting the model's predictions with the actual labels in the test set, the model's accuracy is determined.

3.1 Distribution of input features

Below figure shows how input features are distributed before normalization (see **Figure 1**).

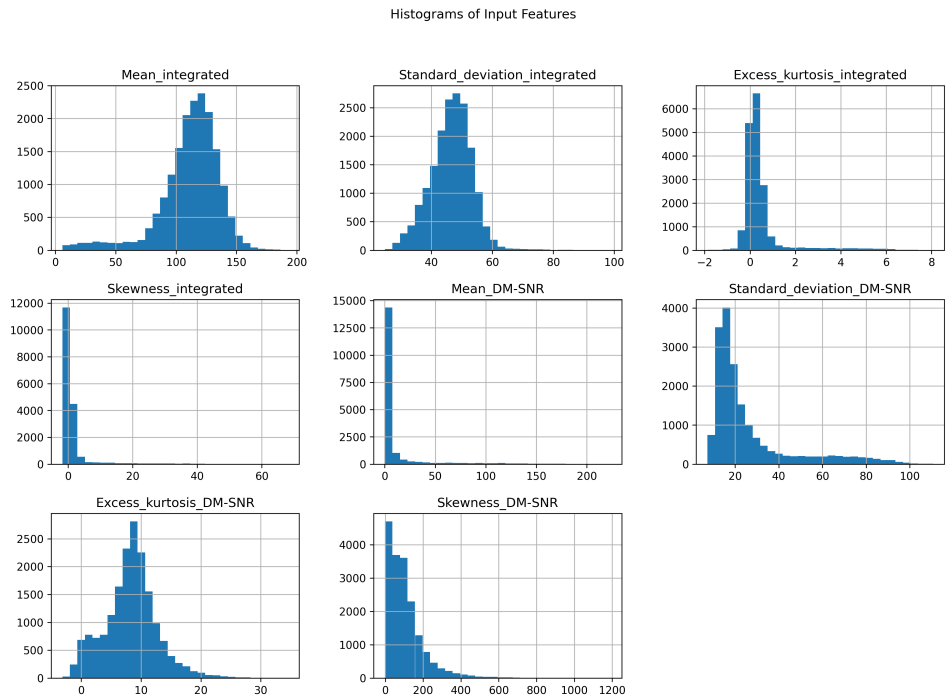

Figure 1: Histograms of Distribution of input features - before normalization

3.2 Distribution of Output label

Below figure shows how class feature is distributed before normalization (see **Figure 2**).

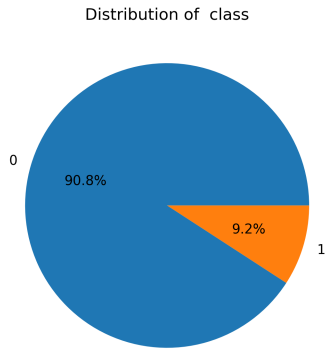

Figure 2: Pie diagram of Distribution of class label

4 Data Processing

The data in this section is split and shuffled, but it hasn't been normalized in any way. Without using normalization, we build the model using the raw data. To make sure there is no bias or innate order in the data, which could cause the model to perform poorly, we first shuffle the data set. In order to train the model on one set of data and assess its performance on an additional, unseen set, we next divided the data into training and validation sets.
After a random shuffling of the data, the data set was divided into training and validation, with 80 percent of the data set going toward training and 20 percent toward validation.

5 Modelling

5.1 Single layer Neural Network

The first model is a baseline model Create a sequential model object named single-layer-model. Add a dense layer with a single neuron and sigmoid activation function to the model. The input dimension is set to 8, as there are 8 input features. Compile the model with the Adam optimizer, binary cross-entropy loss function, and accuracy as the evaluation metric. An early stopping technique was used during the training.
We get: Single Layer Model Training Accuracy: 97.78
Single Layer Model Validation Accuracy: 97.85
Recall: 84.85
Precision: 91.21
F1 Score: 87.91

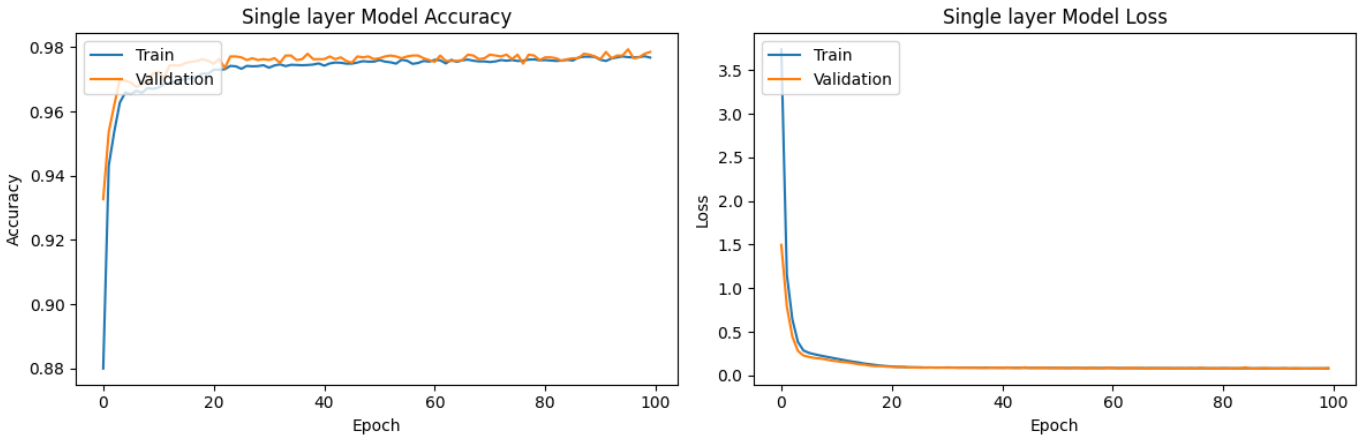

Figure 3: Single layer model accuracy and loss

5.2 Multi layer Neural Network

Define a sequential neural network model (multi layer model) with L1 and L2 regularization.The model has an input dimension of 8 and consists of three layers:The first layer has 16 neurons with ReLU activation and L1 L2 regularization.The second layer has 8 neurons with ReLU activation and L1 L2 regularization.The third layer has 1 neuron with sigmoid activation for binary classification.The model is compiled using the 'adam' optimizer, binary cross-entropy loss, and accuracy as the metric.

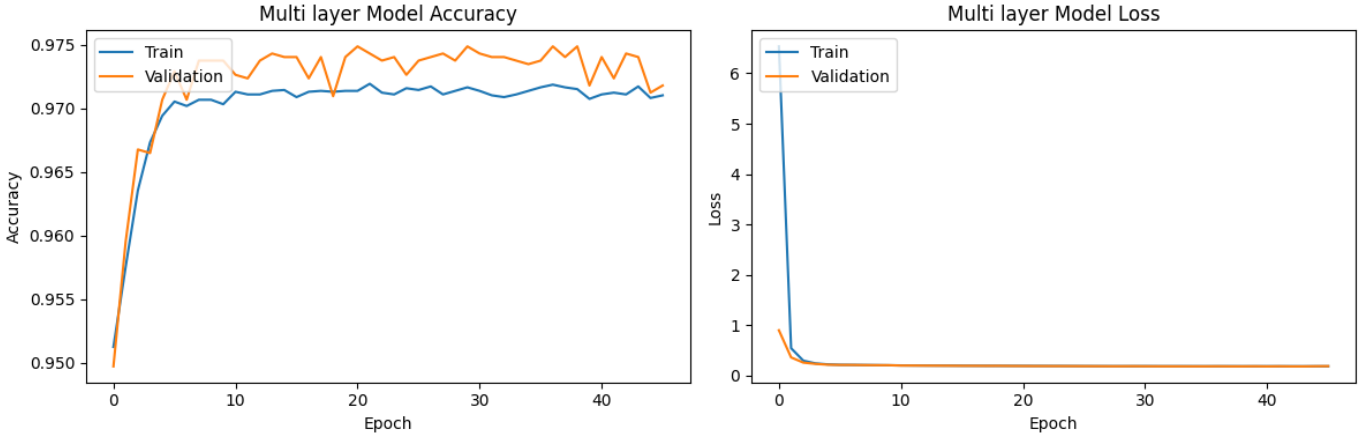

Figure 4: Multi layer model accuracy and loss

Multi layer model Training Accuracy: 96.95
Multi layer model Validation Accuracy: 97.07
Recall: 0.72
Precision: 0.94
F1 Score: 0.82

5.3 Logistic Regression

Create and train a logistic regression model: A" Logistic-Regression model is created with a higher maximum number of iterations (max iter=1000) than the default value (which is 100) to allow the model to converge. The model is then trained using the fit method with the training data (X train, y train).
Logistic Regression Model Training Accuracy: 97.91
Logistic Regression Model Validation Accuracy: 97.96
Recall: 0.84
Precision: 0.93
F1 Score: 0.88

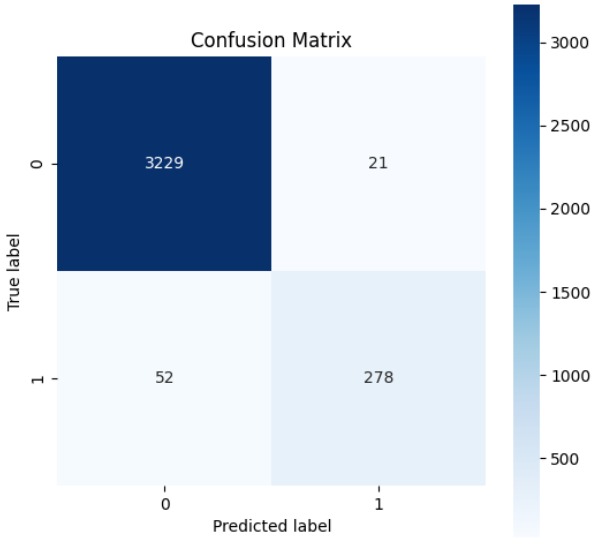

Figure 5: Confusion matrix of logistic Regression

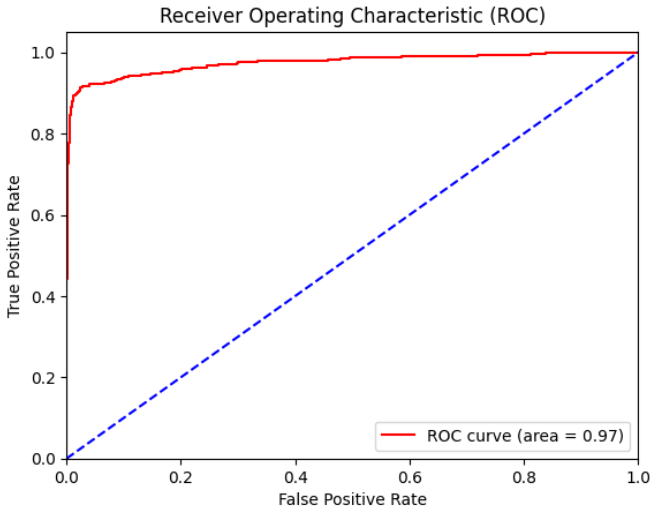

Figure 6: ROC of Logistic regression

5.3.1 Random Baseline Classifier

A random baseline classifier is a simple model that makes predictions by randomly guessing the class labels, without considering the input features.
By comparing the performance of this random baseline classifier with other models, you can evaluate if your actual models are learning useful patterns in the data or just performing at the level of random chance.
Random Baseline Model Training Accuracy: 83.29
Random Baseline Model Validation Accuracy: 83.21
Recall: 0.06
Precision: 0.07
F1 Score: 0.07

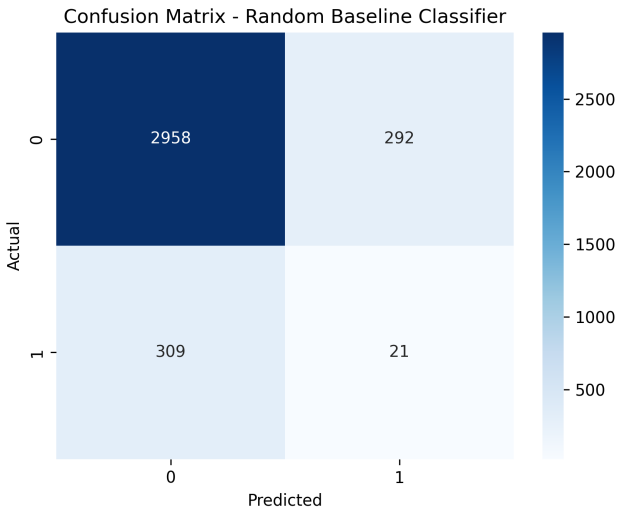

Figure 7: Confusion matrix of Random baseline model

5.4 Custom Prediction Function Model

This model code is for evaluating the performance of a binary classification model. It uses a custom prediction function to make predictions on the training and validation data.
If the model is a tree-based model (e.g., XGBoost), the prediction function uses the predict proba method to get the predicted probabilities. If the model is a linear model (e.g., logistic regression), the prediction function uses the model's weights and bias to calculate the predicted probabilities using the sigmoid activation function.
Custom prediction model Training Accuracy: 98.26 percent
Custom prediction model Validation Accuracy: 98.07 percent
Recall: 86.97 percent
Precision: 91.69 percent
F1 Score: 89.27 percent

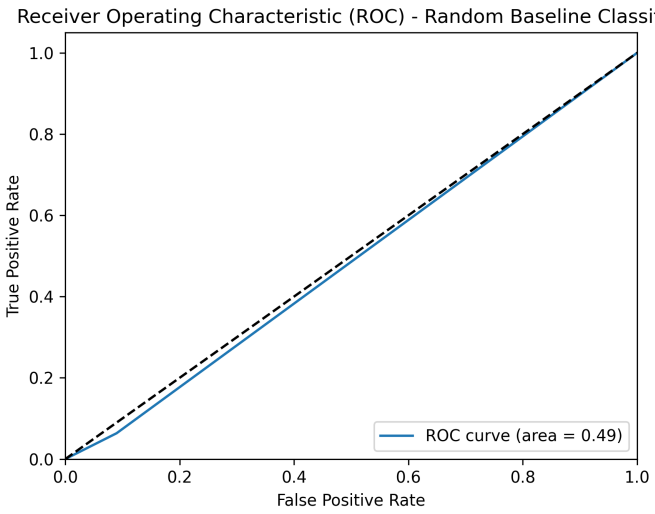

Figure 8: ROC of Random baseline model

6 Model Performances

Accuracy comparisons of all the built models are below

Model Name	Training Accuracy	Validation Accuracy
Single layer Model	97.78	97.85
Multi layer model	96.95	97.07
Logistic Regression model	97.91	97.96
Random Baseline model	83.29	83.21
Custom Prediction Model	98.26	98.07

The table shows the performance of different machine learning models on a task where they have to classify data. Looking at the table, we can see that the Custom Model had the highest training accuracy of 98.26% and highest validation accuracy of 98.07% . The Random Baseline Classifier performed the worst, with a training accuracy of 83.29% and a validation accuracy of 83.21%. This suggests that it wasn't able to learn much from the training data.

7 Model Evaluation

The table given below shows the precision, recall and f1 score for the neural network model.
1. Precision: what proportion of positive identifications was actually correct?
2. Recall: what proportion of actual positives was identified correctly?
3. F1-Score: evaluation metric for classification algorithms, where the best value is at 1 and the worst is at 0.

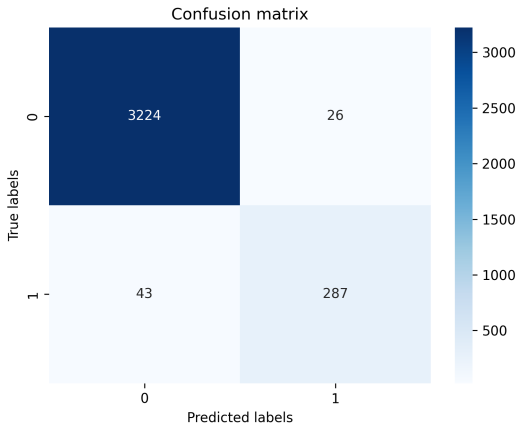

Figure 9: Confusion matrix of custom prediction model

Model Name	Recall	Precision	F-1 Score
Single layer Model	84.85%	91.21%	87.91%
Multi layer model	72%	94%	82%
Logistic Regression model	84%	93%	88%
Random Baseline model	60%	70%	70%
Custom Prediction Model	86.97%	91.69%	89.27%

When we look at the table, we see that the Custom prediction method has highest Recall , Precison and F-1 score . However, the Random Baseline Classifier performed poorly on all three metrics, meaning it struggled to identify positive cases and made many false positive predictions.

8 Feature Importance Analysis

As we have a pretty good idea of which model to use, the number of epochs, and a reasonable validation set to use for the network architecture. The next step is to find out which input features is redundant or insignificant.

8.1 Forward Selection with Check pointing

After selecting the appropriate model, number of epochs and validation set, the next step is to determine the importance of individual input features
Feature 0 - Mean of the integrated profile,
Feature 1 - Standard deviation of the integrated profile,
Feature 2 - Excess kurtosis of the integrated profile,
Feature 3 - Skewness of the integrated profile,
Feature 4 - Mean of the DM-SNR curve,
Feature 5 - Standard deviation of the DM-SNR curve,
Feature 6 - Excess kurtosis of the DM-SNR curve Skewness of the DM-SNR curve.
Feature 7 - Skewness of the DM-SNR curve.

A graph illustrating the performance of the model with respect to these features is provided below.

Performance after removing each feature one by one

The code performs feature importance analysis and reduction. It trains single-feature models

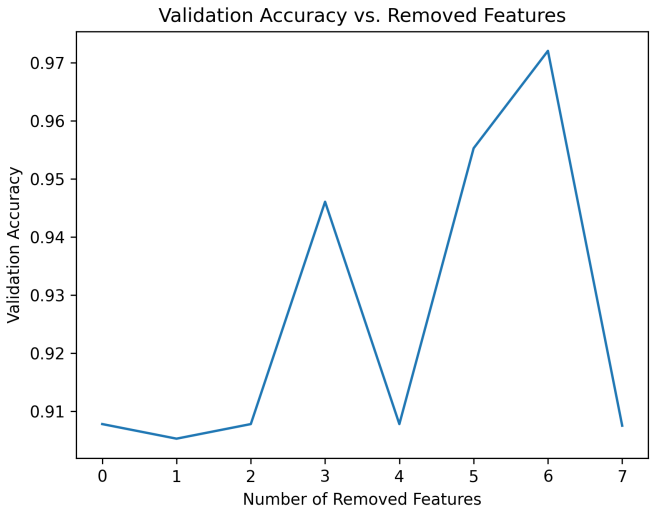

Figure 10: Graph between validation accuracy and each feature

and calculates their validation accuracies. The validation accuracies are plotted to determine the significance of each input feature. Based on the plot, five input features have a small impact on the overall accuracy of the model. Then, the unimportant features are removed and the validation accuracy of the reduced-feature models are compared.

The plot shows the validation accuracy versus the number of removed features. The feature reduction technique used is forward selection, where features are removed one by one based on their significance.

The accuracy of each feature when built individually

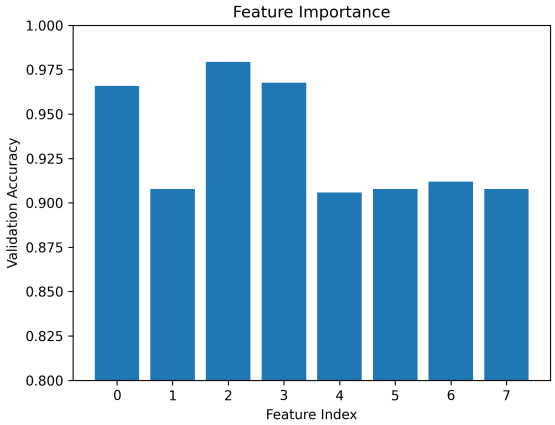

Figure 11: Significance of each feature

8.1.1 Feature Reduction using LIME and SHAP

Two well-liked methods for elucidating machine learning model predictions are SHAP (SHapley Additive exPlanations) and LIME (Local Interpretable Model-agnostic Explanations). While SHAP is based on the idea of game theory and provides global explanations by attributing the contribution of each feature to the final prediction, LIME is a model-agnostic method that produces local explanations for individual predictions. These methods offer insightful information about the features that matter most and how a model generates its predictions.

Two methods are used in the code to implement feature importance analysis: individual feature importance and SHAP. The code trains multiple single-feature models and assesses each one's validation accuracy to determine the importance of each individual feature. The code ranks the features using SHAP values after training an XGBoost model. The validation accuracy of gradually lowering the number of top features in the model is then compared by the code. Plotting the validation accuracy for each method is the last step.

After that, eliminate features that aren't necessary, and evaluate the reduced feature models' validation accuracy. Features were removed iteratively in decreasing order of significance after being sorted according to their SHAP importance scores. Train a single-layer neural network with each reduced feature set, then assess the network's accuracy using the validation set.

In order to identify the ideal number of top features to include in the model, we lastly plotted the validation accuracy for each reduced feature set.

Feature Importance based on SHAP values

Performance after removing less important feature

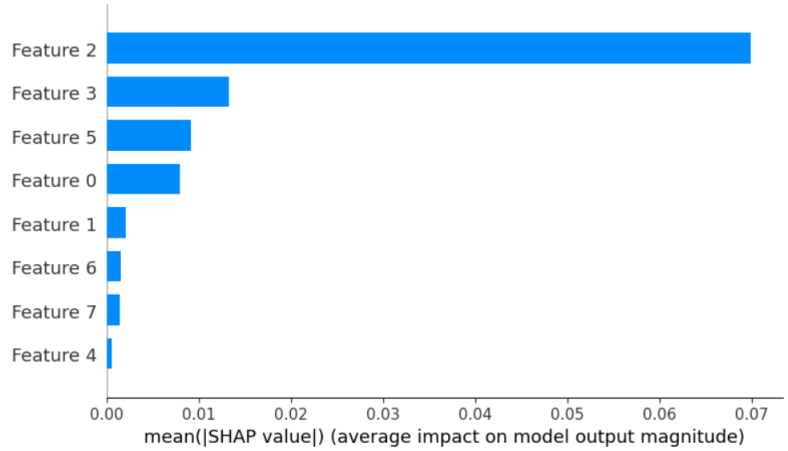

Figure 12: The figure shows the "Excess kurtosis of the integrated profile" is the most important input feature when compared with other input feature.

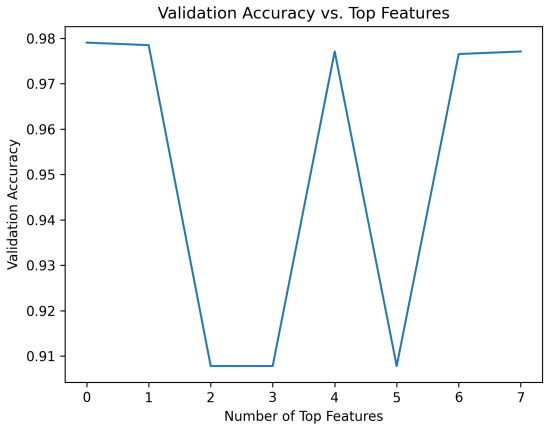

Figure 13: Performance after removing each least important feature

9 Challenges faced

As the data turned to be more overfitting than expected , I faced challenges while training multilayer neural networks in determining how many layers, neurons and epochs are sufficient for training. While coding the checkpoint function, I also ran into some issues because my developed model initially failed to save the best accuracy; however, after a few tries, the best accuracy began to save into the.h5 file. Another important task that needed to be completed was plotting the ROC and the confusion matrix.

10 Code Access

All the Phases of the Project are in this github repository

11 Conclusion

To summarize , Pulsar Stars Prediction is a Binary classification problem to predict if a pulsar star is present or not and in this project , we trained different models to find which model gives best training and validation accuracy scores. Out of all the models we trained , Multi layer neural netwrok model, Logistic regression and custom model has given best accuracies. We conclude that XGboost model with custom prediction function gives the best results . We also made analysis on input feature to determine their significance where both methods determined "Excess Kurtosis of integrated profile" is most important input feature . To conclude, there is still need for improvement by doing cross - validation , hyper parameter tuning and more concepts on this dataset for furter improvements.

12 References

1. M. J. Keith et al., 'The High Time Resolution Universe Pulsar Survey - I. System Configuration and Initial Discoveries',2010, Monthly Notices of the Royal Astronomical Society, vol. 409, pp. 619-627. DOI: 10.1111/j.1365-2966.2010.17325.x

2. D. R. Lorimer and M. Kramer, 'Handbook of Pulsar Astronomy', Cambridge University Press, 2005.

3. R. J. Lyon, 'Why Are Pulsars Hard To Find?', PhD Thesis, University of Manchester, 2016.

4. R. J. Lyon et al., 'Fifty Years of Pulsar Candidate Selection: From simple filters to a new principled real-time classification approach', Monthly Notices of the Royal Astronomical Society 459 (1), 1104-1123, DOI: 10.1093/mnras/stw656

5. R. J. Lyon, 'PulsarFeatureLab', 2015, [Web link].