PULSAR PREDICTION

Naga Raju Vodnala

May 8, 2023

Contents

1 Introduction

The High Time Resolution Universe Survey (HTRU) is a project aimed at discovering new pulsars and studying their properties. Pulsars are astronomical objects that emit regular pulses of radio waves, and they are of great interest to astronomers because they provide valuable information about the properties of stars and the universe as a whole.

The HTRU project uses radio telescopes to observe the sky and collect data on pulsar candidates. The data is then analyzed using computer algorithms to identify which candidates are most likely to be pulsars. The HTRU project has discovered hundreds of new pulsars and has provided valuable insights into their properties, such as their spin rate, magnetic field, and distance from Earth.

1.1 Overview

1.1.1 What is a pulsar?

When you gaze at the enchanting night sky, you might notice a seemingly stationary, twinkling object, and there's a slim possibility that you're observing a pulsar. Although pulsars often resemble shimmering stars and seem to twinkle with a regular rhythm, their light doesn't truly flicker or pulsate, and they aren't stars. Pulsars are compact, spherical objects roughly the size of a large city but possess more mass than our sun or other typical stars.

1.1.2 What makes them so captivating?

Scientists study pulsars to investigate extreme states of matter, search for planets beyond our solar system, and measure cosmic distances. Additionally, pulsars could potentially help researchers detect gravitational waves, which could lead to the discovery of high-energy cosmic events like supermassive black hole collisions. Pulsars are enigmatic and relatively new to the scientific community, and their behavior is strikingly different from anything we experience on Earth, making them the epitome of extreme science.

1.1.3 Motivation:

Artificial intelligence could be the ideal tool for exploring the universe, as identifying specific celestial objects can be a slow and laborious process. With this in mind, the project's motivation is to develop supervised learning algorithms for predicting pulsar classes using binary classification. Binary classification is a machine learning technique that determines whether an element is true or false, or 1 or 0.
The HTRU2 data set contains information on pulsar candidates, which are astronomical objects that emit regular pulses of radio waves. Pulsars are of great interest to astronomers because they provide valuable information about the properties of stars and the universe as a whole.

The data set was created by a team of researchers from the University of Manchester, UK, who analyzed data from the High Time Resolution Universe Survey (HTRU). The HTRU is a project aimed at discovering new pulsars and studying their properties.

The data set contains statistical measures of the pulsar candidates' profile and DMSNR curve. The profile is a plot of the intensity of the radio signal from the pulsar as a function of time, while the DMSNR curve is a plot of the dispersion measure versus the signal-to-noise ratio. Dispersion measure is a measure of how much the pulsar's radio waves are dispersed as they pass through the interstellar medium, while signal-to-noise ratio is a measure of the strength of the radio signal relative to the background noise.

The statistical measures included in the data set provide information on the shape, width, and intensity of the pulsar candidates' profile and DMSNR curve. Machine learning algorithms can be trained on this data set to predict whether a candidate is a pulsar or not, based on its statistical features. The HTRU2 data set is often used as a benchmark for machine learning algorithms, particularly for classification tasks. It has been used to evaluate the performance of various machine learning models, such as decision trees, support vector machines, neural networks, and ensemble methods. The data set is freely available for download and can be used for research and educational purposes.

2 Dataset

Link for the dataset: https://archive.ics.uci.edu/ml/datasets/HTRU2

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 DM-SNR 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.

The goal of a machine learning model trained on the HTRU2 data set is to accurately predict the classification label for each pulsar candidate, based on its statistical features. This is a binary classification task, where the model must learn to distinguish between pulsars and non-pulsars. The accuracy of the model is measured by comparing its predictions to the true labels in the test set.

Accurate classification of pulsar candidates is important for the study of pulsars and their properties. The HTRU2 data set provides a valuable resource for researchers and machine learning practitioners to develop and test new algorithms for pulsar classification.

2.1 Visualization of the distribution of each input features

The below histograms shows the each info highlights of how they distributed before normalization.

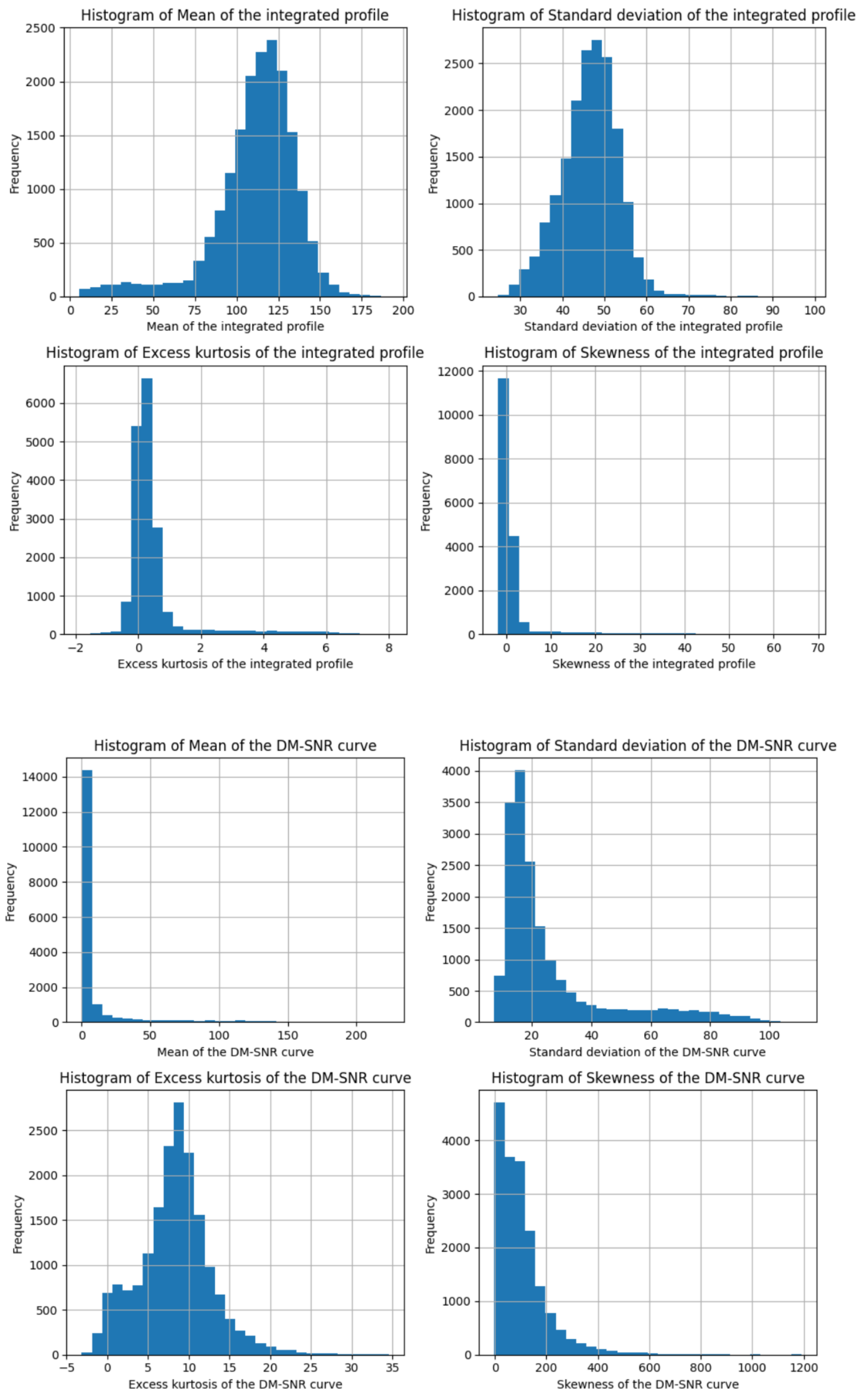


Figure 1: Input Data Distribution Histograms - Before Normalization

2.2 Distribution of the output labels

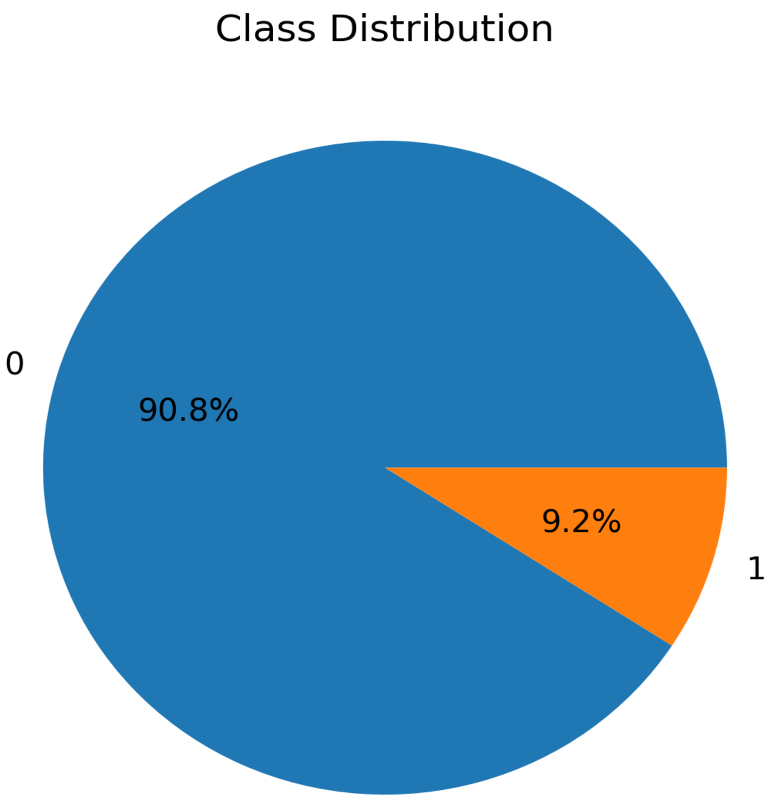


Figure 2: Output Data Distribution - Before Normalization

2.3 Data Normalization

NOTE: Data normalization is done before shuffling and splitting the data
Data pre-processing is essential before data mining to address the non-uniform distribution of data. To achieve this, normalization techniques are used to make the optimization problem more numerically stable and improve training. Normalization helps to ensure all values lie between 0 and 1 and outliers are visible within the normalized data. There are two normalization techniques available, each with its own consequences, but either technique can be used for now.

Mean Normalization Formula

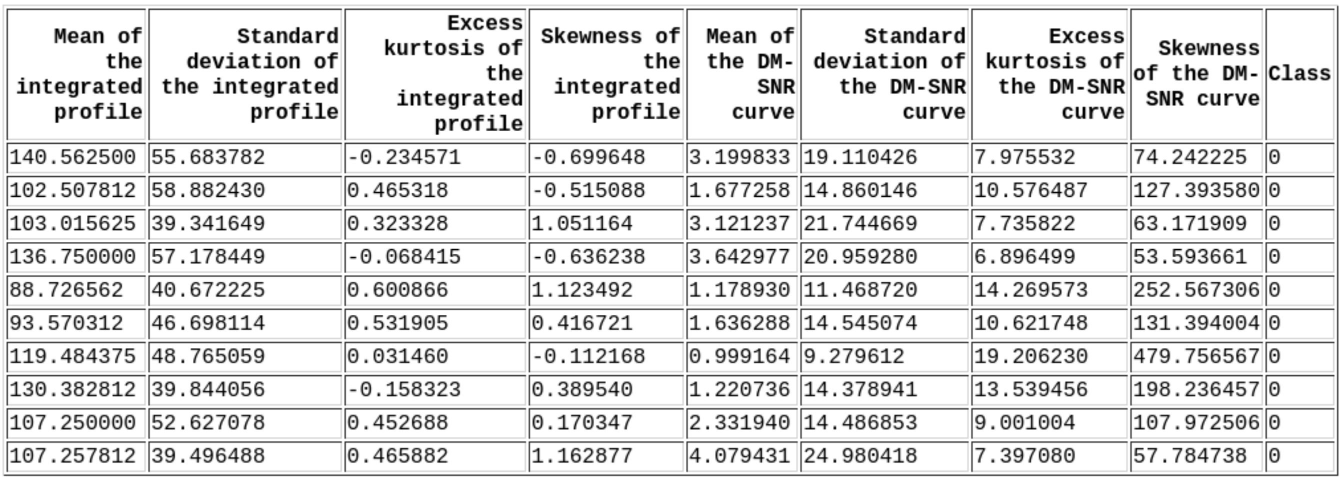
$$X_{normalized} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Z-Score Normalization

$$X_{normalized} = \frac{X - X_{mean}}{X_{standard_deviation}}$$

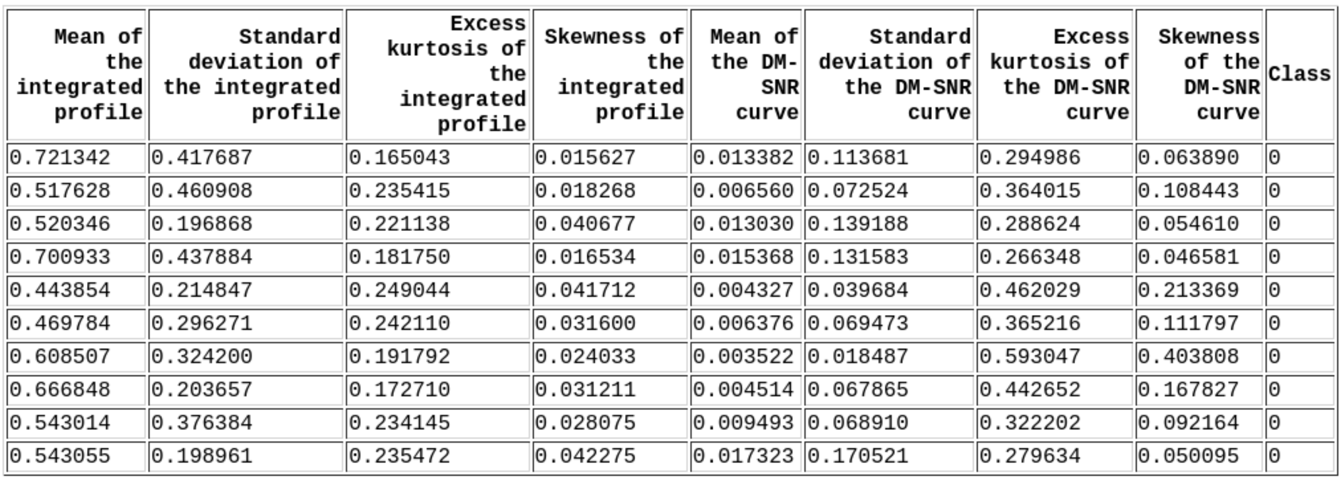
Before splitting" refers to the fact that the data-set has not yet been divided into separate training and validation sets. In machine learning, it is common to split the data-set into separate training and validation sets, where the training set is used to train the model and the validation set is used to evaluate the performance of the model.

"Before normalization" refers to the fact that the data-set has not yet been scaled to a common range. In machine learning, it is common to normalize or scale the data-set to ensure that all variables are on a similar scale. This is done to prevent variables with larger values from dominating the training process and to ensure that the model is sensitive to all variables equally.



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	Class
140.562500	55.683782	-0.234571	-0.699648	3.199833	19.110426	7.975532	74.242225	0
102.507812	58.882430	0.465318	-0.515088	1.677258	14.860146	10.576487	127.393580	0
103.015625	39.341649	0.323328	1.051164	3.121237	21.744669	7.735822	63.171909	0
136.750000	57.178449	-0.068415	-0.636238	3.642977	20.959280	6.896499	53.593661	0
88.726562	40.672225	0.600866	1.123492	1.178930	11.468720	14.269573	252.567306	0
93.570312	46.698114	0.531905	0.416721	1.636288	14.545074	10.621748	131.394004	0
119.484375	48.765059	0.031460	-0.112168	0.999164	9.279612	19.206230	479.756567	0
130.382812	39.844056	-0.158323	0.389540	1.220736	14.378941	13.539456	198.236457	0
107.250000	52.627078	0.452688	0.170347	2.331940	14.486853	9.001004	107.972506	0
107.257812	39.496488	0.465882	1.162877	4.079431	24.980418	7.397080	57.784738	0

Figure 3: Data-set - Before Splitting and Before Normalization



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	Class
0.721342	0.417687	0.165043	0.015627	0.013382	0.113681	0.294986	0.063890	0
0.517628	0.460908	0.235415	0.018268	0.006560	0.072524	0.364015	0.108443	0
0.520346	0.196868	0.221138	0.040677	0.013030	0.139188	0.288624	0.054610	0
0.700933	0.437884	0.181750	0.016534	0.015368	0.131583	0.266348	0.046581	0
0.443854	0.214847	0.249044	0.041712	0.004327	0.039684	0.462029	0.213369	0
0.469784	0.296271	0.242110	0.031600	0.006376	0.069473	0.365216	0.111797	0
0.608507	0.324200	0.191792	0.024033	0.003522	0.018487	0.593047	0.403808	0
0.666848	0.203657	0.172710	0.031211	0.004514	0.067865	0.442652	0.167827	0
0.543014	0.376384	0.234145	0.028075	0.009493	0.068910	0.322202	0.092164	0
0.543055	0.198961	0.235472	0.042275	0.017323	0.170521	0.279634	0.050095	0

Figure 4: Data-set - Before Splitting and After Normalization

Below, after normalization, the values of each feature are re-scaled to have a mean of 0 and a standard deviation of 1. This process ensures that each feature has equal importance in the model and that the model is not biased towards any particular feature

.

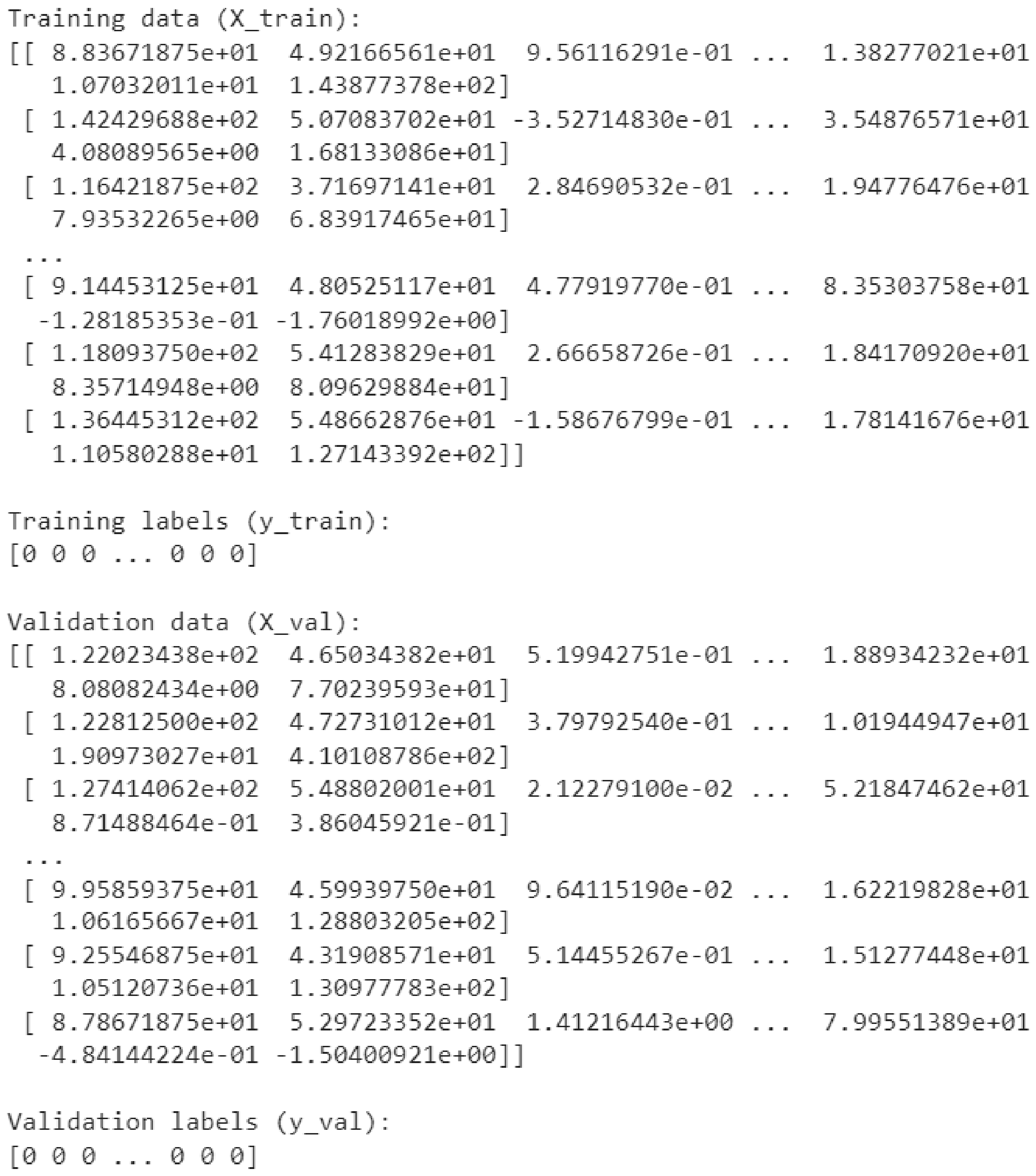
3 Data Processing

3.1 Data Shuffling and Splitting

In this section, we use shuffled and split data for building the model, but the data is not normalized. We construct the model using the raw data without applying normalization.

We first shuffle the data set to ensure that there is no inherent order or bias in the data, which could lead to poor model performance. Then, we split the data into training and validation sets, allowing us to train the model on one set of data and evaluate its performance on another unseen set.

The data was randomly shuffled and then the data-set was split into training and validation, where 80% of the data-set was allocated for training and 20% was allocated for validation.



```
Training data (X_train):
[[ 8.83671875e+01  4.92166561e+01  9.56116291e-01 ...  1.38277021e+01
   1.07032011e+01  1.43877378e+02]
 [ 1.42429688e+02  5.07083702e+01 -3.52714830e-01 ...  3.54876571e+01
   4.08089565e+00  1.68133086e+01]
 [ 1.16421875e+02  3.71697141e+01  2.84690532e-01 ...  1.94776476e+01
   7.93532265e+00  6.83917465e+01]
 ...
 [ 9.14453125e+01  4.80525117e+01  4.77919770e-01 ...  8.35303758e+01
  -1.28185353e-01 -1.76018992e+00]
 [ 1.18093750e+02  5.41283829e+01  2.66658726e-01 ...  1.84170920e+01
   8.35714948e+00  8.09629884e+01]
 [ 1.36445312e+02  5.48662876e+01 -1.58676799e-01 ...  1.78141676e+01
   1.10580288e+01  1.27143392e+02]]

Training labels (y_train):
[0 0 0 ... 0 0 0]

Validation data (X_val):
[[ 1.22023438e+02  4.65034382e+01  5.19942751e-01 ...  1.88934232e+01
   8.08082434e+00  7.70239593e+01]
 [ 1.22812500e+02  4.72731012e+01  3.79792540e-01 ...  1.01944947e+01
   1.90973027e+01  4.10108786e+02]
 [ 1.27414062e+02  5.48802001e+01  2.12279100e-02 ...  5.21847462e+01
   8.71488464e-01  3.86045921e-01]
 ...
 [ 9.95859375e+01  4.59939750e+01  9.64115190e-02 ...  1.62219828e+01
   1.06165667e+01  1.28803205e+02]
 [ 9.25546875e+01  4.31908571e+01  5.14455267e-01 ...  1.51277448e+01
   1.05120736e+01  1.30977783e+02]
 [ 8.78671875e+01  5.29723352e+01  1.41216443e+00 ...  7.99551389e+01
  -4.84144224e-01 -1.50400921e+00]]

Validation labels (y_val):
[0 0 0 ... 0 0 0]
```

Figure 5: Data showing Model accuracy and model loss

4 Modelling

A feed forward artificial neural network architectures was used to create the model.

The model is built using non-normalized data, we can analyze how the model performs without the benefits of normalization.

NOTE: Data is shuffle. Thus, the result will vary every time. All models were compiled and fit on May 08, 2023.

4.1 Selected Neural Network Architecture

4.1.1 Single_Layer_Model

The first model is a baseline model (act as a control) that has a basic architecture of one input and one output layer. An early stopping technique was used during the training.

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.

We get:
Single Layer Model Training Accuracy: 96.47
Single Layer Model Validation Accuracy: 96.28

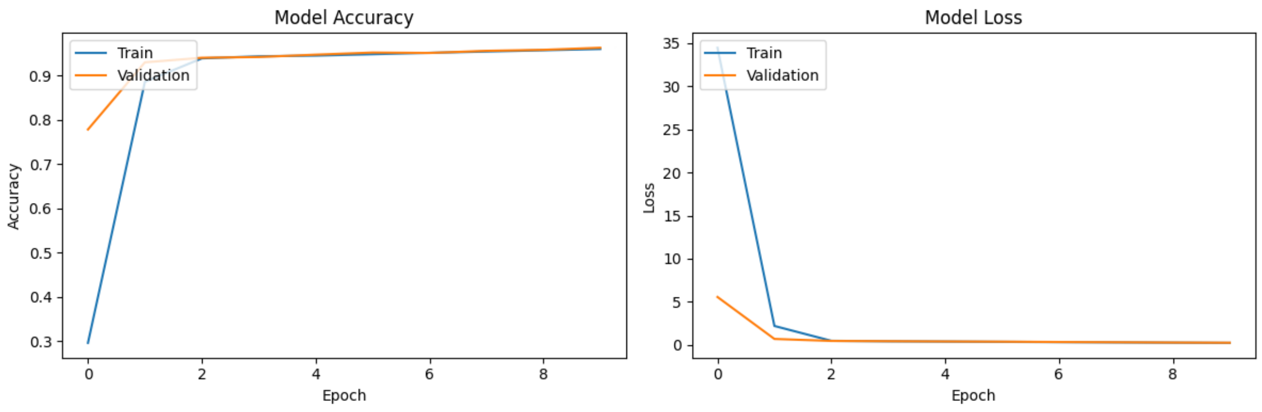


Figure 6: Data showing Model accuracy and model loss

Recall: 69.59
Precision: 89.14
F1 Score: 78.16

4.1.2 Multi_Layer_Model

Define a sequential neural network model (multi_layer_modelSS) 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

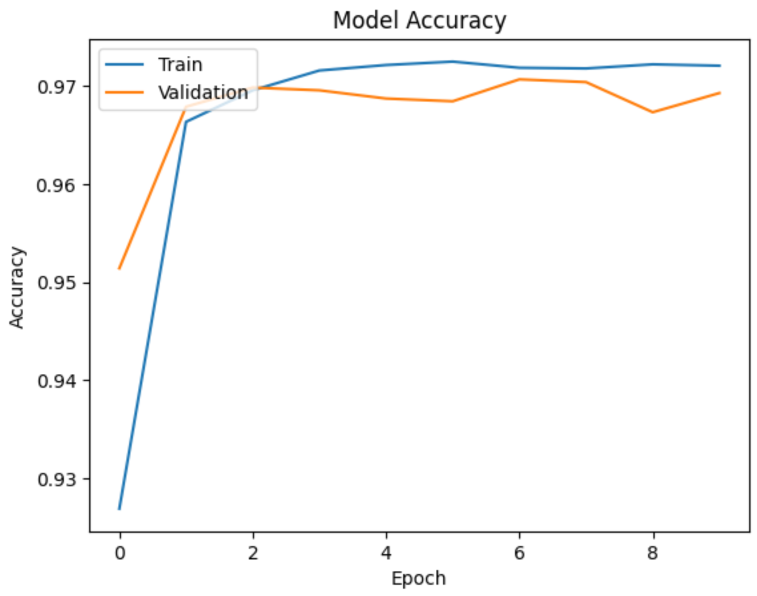


Figure 7: Data showing Model accuracy and model loss

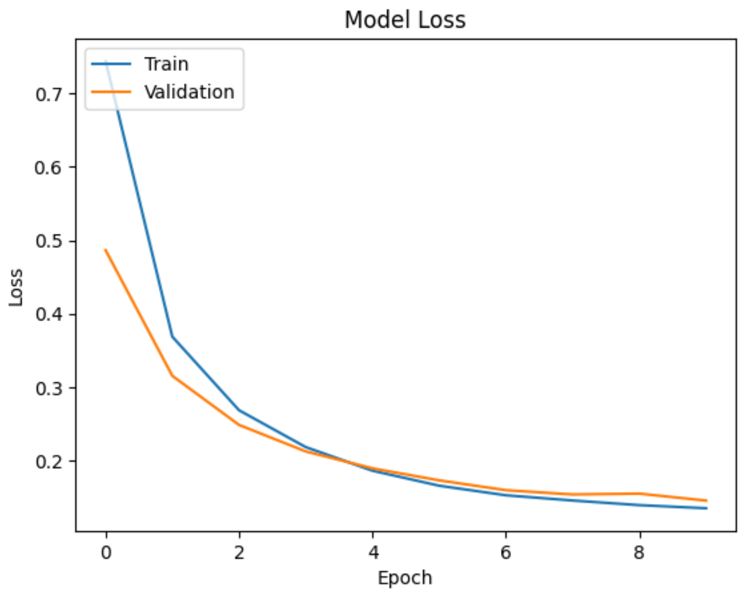


Figure 8: Data showing Model accuracy and model loss

Training Accuracy: 97.21 Validation Accuracy: 96.93 Recall: 0.71 Precision: 0.96 F1 Score: 0.82

4.1.3 Logistic Regression

Building a model using 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).

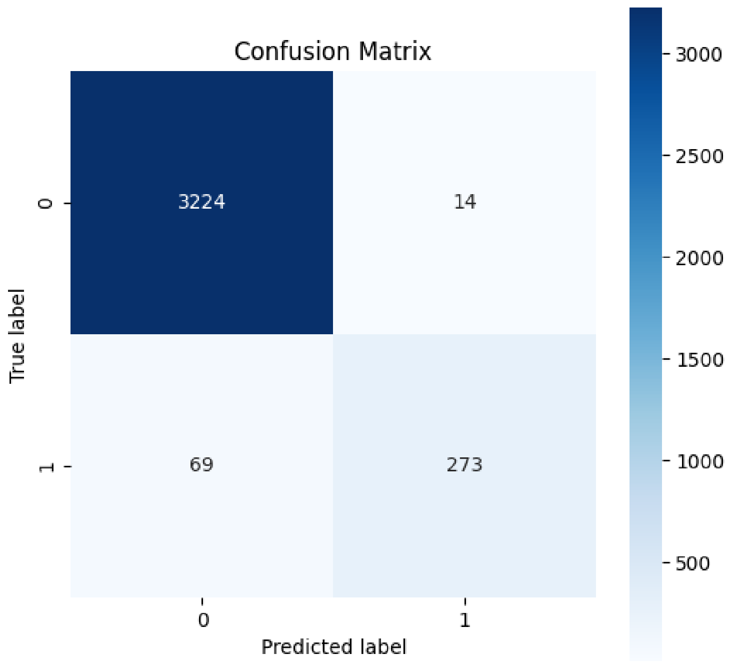


Figure 9: Confusion Matrix

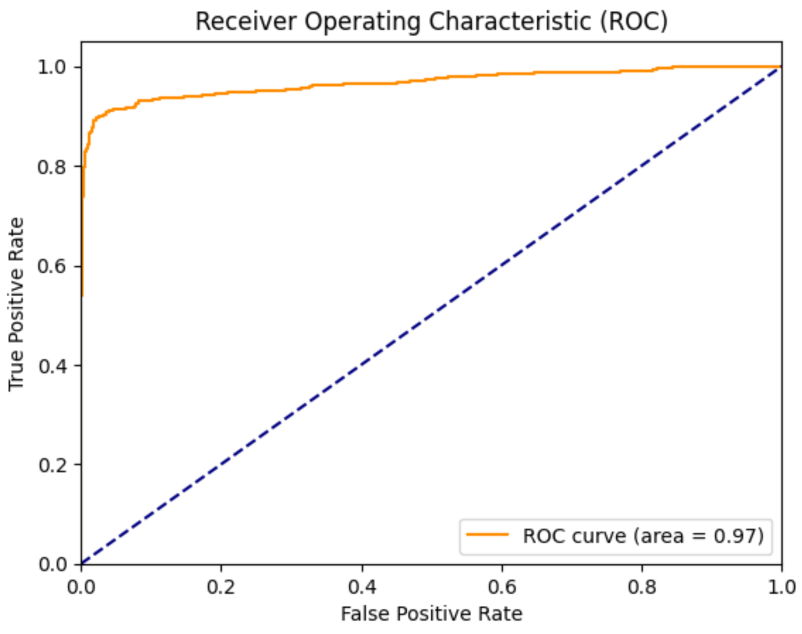


Figure 10: ROC CURVE

Logistic Regression Model Training Accuracy: 97.95 Logistic Regression Model Validation Accuracy: 97.68 Recall: 0.80 Precision: 0.95 F1 Score: 0.87

4.1.4 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. It serves as a basic point of comparison for more complex models, allowing you to assess whether your actual model is performing better than random chance. In the context of the provided code, a stratified random baseline classifier is used, which generates predictions by respecting the training set's class distribution.

Here, Import the necessary libraries and functions.Create and train a stratified random baseline classifier using the training data.Then make predictions on the validation set using the trained classifier. Calculate and print the accuracy of the random baseline classifier on the validation set.

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

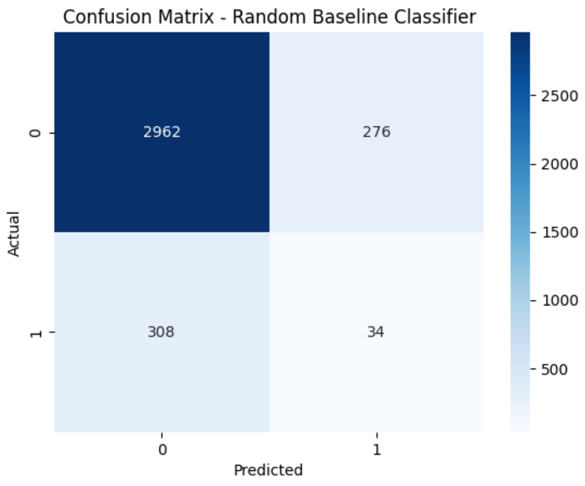


Figure 11:

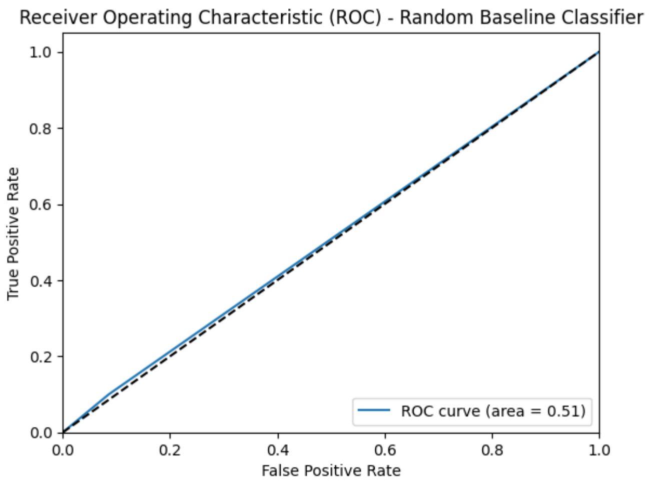


Figure 12:

Random Baseline Classifier Validation Accuracy: 83.69 Training Accuracy: 83.62 Validation Accuracy: 83.69 Recall: 0.10 Precision: 0.11 F1 Score: 0.10

4.1.5 Custom_Prediction_Function

This model code is for evaluating the performance of a binary classification model. It uses a custom prediction function, my_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.'

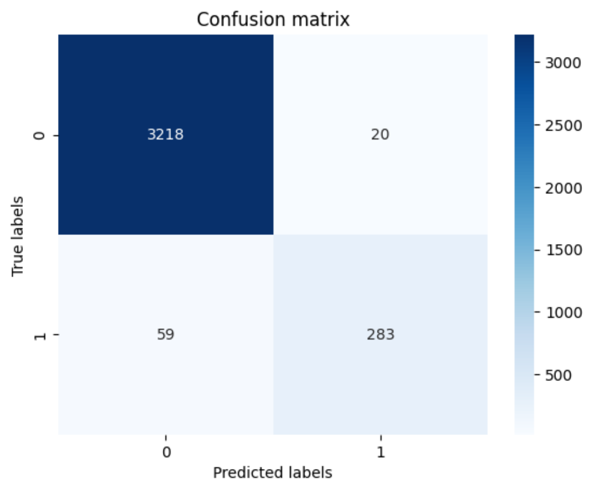


Figure 13:

Training Accuracy: 99.86Validation Accuracy: 97.79Recall: 82.75Precision: 93.40F1 Score: 87.75

4.1.6 Model Performance

Accuracy comparison for all the built models:

MODEL NAME	TRAINING ACCURACY	VALIDATION ACCURACY
SINGLE LAYER MODEL	96.47	96.28
MULTI LAYER MODEL	97.21	96.93
LOGICAL REGRESSION MODEL	97.95	97.68
RANDOM BASELINE CLASSIFIER	83.62	83.69
CUSTOM MODEL	99.86	97.79

The table shows the performance of different machine learning models on a task where they have to classify data. The models were trained on a specific set of data, and their accuracy was measured to see how well they could classify new data.

The accuracy of a model is a measure of how well it performs on the task. The training accuracy shows how well the model did on the data it was trained on, while the validation accuracy shows how well it can classify new data that it hasn't seen before.

Looking at the table, we can see that the Custom Model had the highest training accuracy of 99.86%, which means it did very well on the data it was trained on. However, its validation accuracy of 97.79% was lower than the Logical Regression Model, which had the highest validation accuracy of 97.68%.

The Random Baseline Classifier performed the worst, with a training accuracy of 83.62% and a validation accuracy of 83.69%. This suggests that it wasn't able to learn much from the training data.

Overall, the accuracy measures give us an idea of how well each model performed on the classification task and how well they can generalize to new data. It's important to choose a model with a good validation accuracy, as this tells us how well it will perform on new data.

5 Model Evaluation

Three essential classification model metrics to evaluate. 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.

Model	Recall	Precision	F1-Score
SINGLE LAYER MODEL	69.59	89.14	78.16
MULTI LAYER MODEL	71.00	96.00	82.00
LOGICAL REGRESSION MODEL	80.00	91.00	87.00
RANDOM BASELINE CLASSIFIER	10.00	11.00	10.00
CUSTOM MODEL	82.75	93.40	87.75

A useful tool when predicting the probability of a binary outcome is the Receiver Operating Characteristic curve or ROC curve. The area covered by the curve is the area between the red line and the axis. This area covered is AUC. The bigger the area covered, the better the machine learning models are at distinguishing the given classes. In other words, the AUC can be used as a summary of the model skill. The ideal value for AUC is 1.

The table shows how well different machine learning models perform on a task where they have to classify data. In addition to accuracy, we use three other metrics: recall, precision, and F1-Score to measure their performance.

Recall is a measure of how well the model can identify actual positive cases. Precision, on the other hand, is a measure of how well the model's positive predictions match the actual positive cases. F1-Score is a measure that combines both recall and precision.

When we look at the table, we see that the Logical Regression Model had the highest recall of 80%, which means it could identify most of the actual positive cases. The Custom Model had the highest precision of 93.4%, which means it had fewer false positive predictions than the other models. Finally, the Multi Layer Model had the highest F1-Score of 82%, which is a good overall measure of its performance.

However, the Random Baseline Classifier performed poorly on all three metrics, meaning it struggled to identify positive cases and made many false positive predictions.

Overall, these metrics provide us with more information on how well the models perform on the classification task. We should consider all the metrics together to get a better understanding of the models' effectiveness.

6 Feature Importance Analysis

As of now, 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.

6.1 Significance of individual features

6.1.1 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

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

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

Below are the accuracies of each feature when built individually:

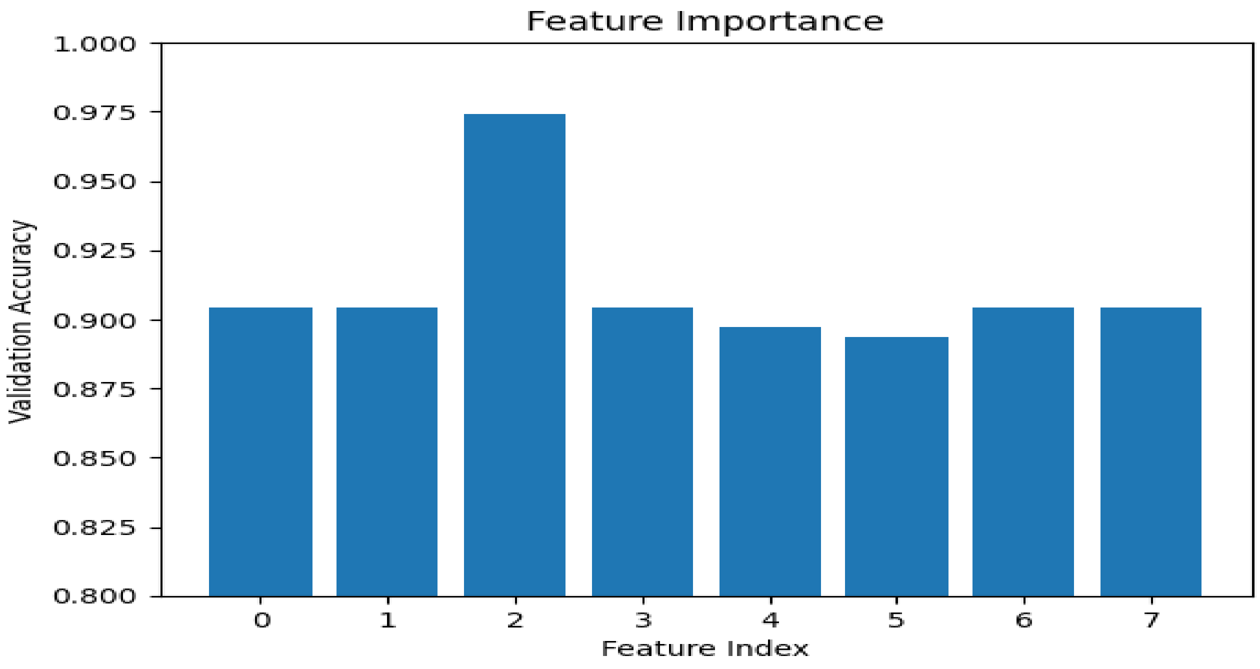


Figure 14: Significance of individual features

Performance after removing less important features one by one:

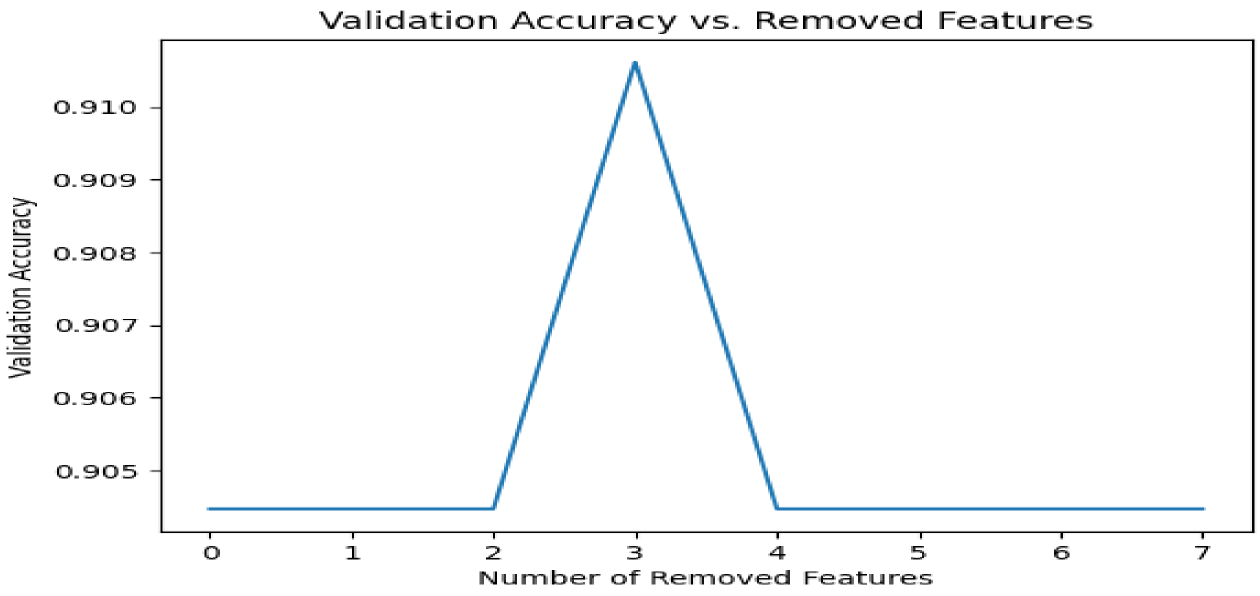


Figure 15: Graph between validation accuracy and removed features

6.1.2 Reduction using LIME and SHAP:

LIME (Local Interpretable Model-agnostic Explanations) and SHAP (SHapley Additive exPlanations) are two popular techniques used for explaining the predictions of machine learning models. LIME is a model-agnostic method that generates local explanations for individual predictions, while SHAP is based on the concept of game theory and provides global explanations by attributing the contribution of each feature to the final prediction. Both techniques provide valuable insights into how a model is making its predictions and which features are the most important.

The code implements feature importance analysis using two techniques: SHAP and individual feature importance. For individual feature importance, the code trains multiple single-feature models and evaluates the validation accuracy of each. For SHAP, the code trains an XGBoost model and uses SHAP values to rank the features. Then, the code compares the validation accuracy of progressively reducing the number of top features in the model. Finally, plots the validation accuracy for each approach.

Next, remove unimportant features and compare the validation accuracies of the reduced feature models. Sorted the features by their SHAP importance scores and iteratively removed features in descending order of their importance. For each reduced feature set, train a single-layer neural network and evaluated its accuracy on the validation set.

Finally, we plotted the validation accuracies for each reduced feature set to determine the optimal number of top features to use in the model.

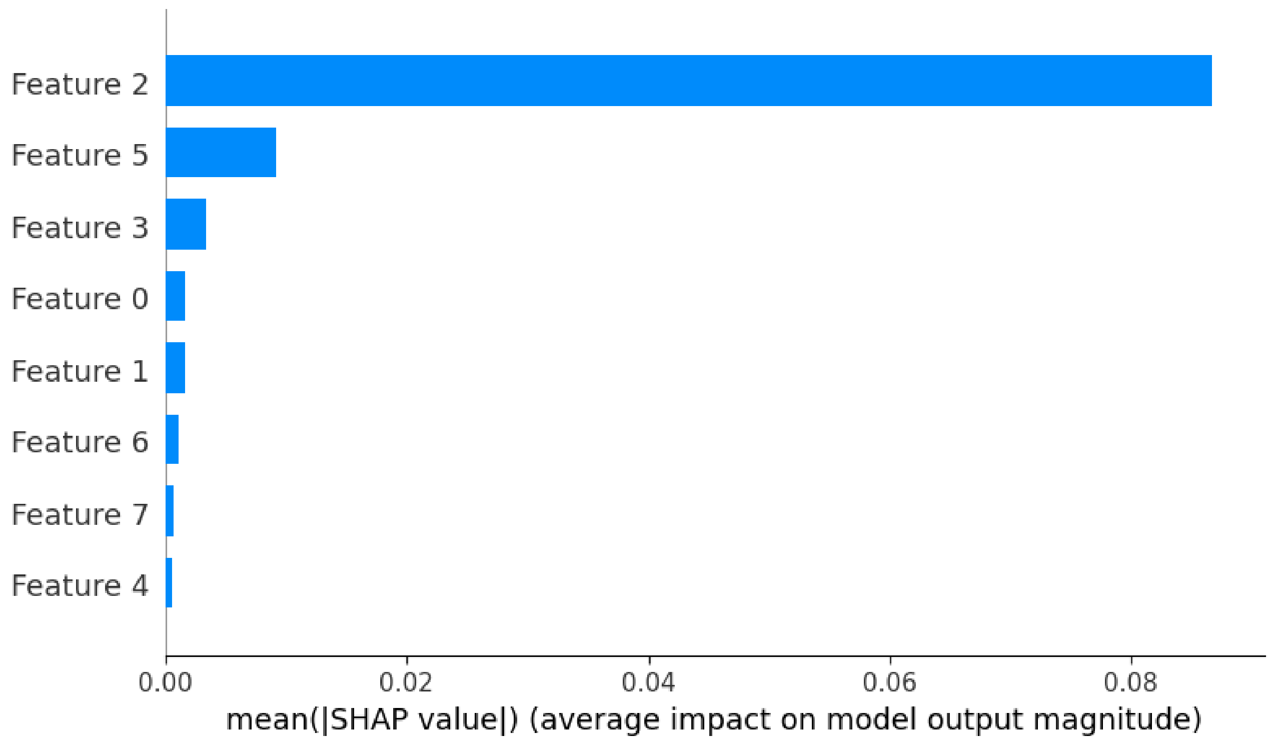


Figure 16: The below figure shows the "Excess kurtosis of the integrated profile" is the most important input feature when compared with other input feature.

Performance after removing less important features: We can see the graph below for the performance.

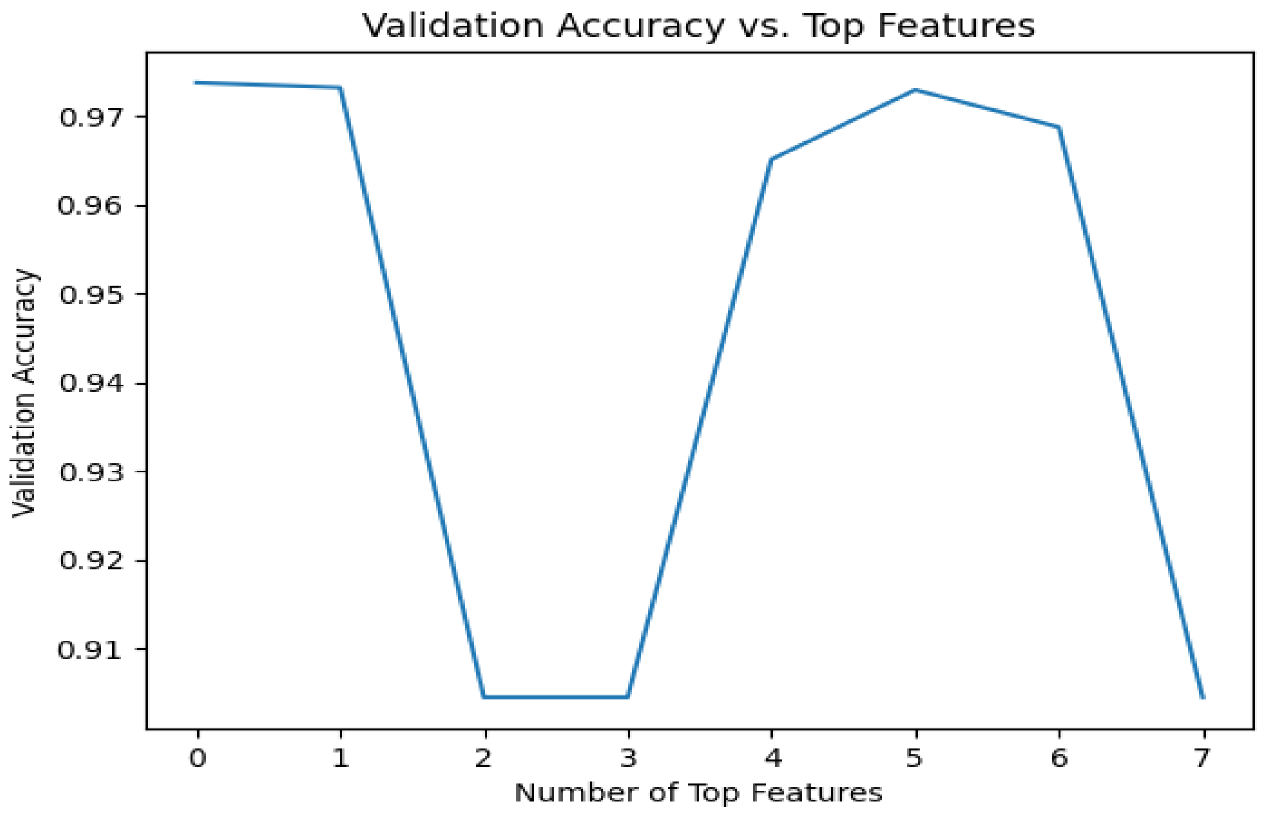


Figure 17: Performance after removing less important features

7 Challenges Faced

I faced challenges while building multi-layered models as my data turned out to be more over-fitting, which reduced the accuracy. However, I used the early stopping technique and L1_L2 regularization to overcome this issue. I also faced some problems while coding the checkpoint function as my developed model didn't save the best accuracy at first, but upon multiple trials, the best accuracy started saving into the .h5 file. Plotting ROC and confusion matrix were also major tasks that required attention. '

8 Conclusion

The aim of this project is to develop a neural network model to predict whether an element belongs to one of two categories, either true or false (1 or 0). Specifically, this model will be used to classify pulsars. Models like Muilti_Layer_Model, Logical Regression and Custom_Model gave similar accuracies, but the accuracy from Random Baseline Classifier is really low when compared with others. **Note:** The Random Baseline Classifier is a model that predicts the target variable randomly. It doesn't use any information from the input features to make predictions, which makes it a very simple and basic model. Therefore, the accuracy of this model is expected to be lower compared to other models that use the input features to make predictions.

In both the reduction techniques, SHAP feature reduction and Forward selection, the feature 'Excess kurtosis of the integrated profile' was found to be the most important feature for predicting the class of pulsars.

To summarize, the HTRU_2 machine learning model is used to predict whether a pulsar is present or not. It is a binary classification problem where the input data consists of various features such as mean and standard deviation of the integrated profile and DMSNR curve, and the output is either 1 (pulsar present) or 0 (pulsar not present). The goal is to develop a model that can accurately predict whether a pulsar is present or not based on these input features.

9 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. P. Eatough et al., 'Selection of radio pulsar candidates using artificial neural networks', Monthly Notices of the Royal Astronomical Society, vol. 407, no. 4, pp. 2443-2450, 2010.

[6] S. D. Bates et al., 'The high time resolution universe pulsar survey vi. an artificial neural network and

timing of 75 pulsars', Monthly Notices of the Royal Astronomical Society, vol. 427, no. 2, pp. 1052-1065, 2012.

[7] D. Thornton, 'The High Time Resolution Radio Sky', PhD thesis, University of Manchester, Jodrell Bank Centre for Astrophysics School of Physics and Astronomy, 2013.

[8] K. J. Lee et al., 'PEACE: pulsar evaluation algorithm for candidate extraction a software package for post-analysis processing of pulsar survey candidates', Monthly Notices of the Royal Astronomical Society, vol. 433, no. 1, pp. 688-694, 2013.

[9] V. Morello et al., 'SPINN: a straightforward machine learning solution to the pulsar candidate selection problem', Monthly Notices of the Royal Astronomical Society, vol. 443, no. 2, pp. 1651-1662, 2014.

[10] R. J. Lyon, 'PulsarFeatureLab', 2015, [Web Link].