Pulsar Star Detection: A Comparative Analysis of Classification Algorithms using SMOTE

Apratim Sadhu
Department of Computer Science Engineering
Chandigarh University
Mohali, India
Email: apratimsadhu01 [AT] gmail.com

Abstract— **A Pulsar is a highly magnetized rotating compact star whose magnetic poles emit beams of radiation. The application of pulsar stars has a great application in the field of astronomical study. Applications like the existence of gravitational radiation can be indirectly confirmed from the observation of pulsars in a binary neutron star system. Therefore, the identification of pulsars is necessary for the study of gravitational waves and general relativity. Detection of pulsars in the universe can help research in the field of astrophysics. At present, there are millions of pulsar candidates present to be searched. Machine learning techniques can help detect pulsars from such a large number of candidates. The paper discusses nine common classification algorithms for the prediction of pulsar stars and then compares their performances using various classification metrics such as classification accuracy, precision and recall value, ROC score and f-score on both balanced and unbalanced data. SMOTE-technique is used to balance the data for better results. Among the nine algorithms, XGBoosting algorithm achieved the best results. The paper is concluded with prospects of machine learning for pulsar detection in the field of astronomy.**

Keywords:* *Accuracy, Algorithms, Classification, Machine Learning, Pulsars, SMOTE

I. Introduction

Pulsars are rotating neutron stars that have a very strong magnetic field and pulses of radiation at very regular intervals that typically range from milliseconds to seconds[1]. This accelerated light produces very powerful beams of light. They have huge astronomical applications. Due to the variety of important applications in the field of astrophysics, it is very important to successfully detect them. Various astronomical procedures have been employed for the task of pulsar detection. Astronomical telescopes play an important role in capturing radiations from the pulsars.

Machine learning systems can be a very useful alternative for the detection of pulsars. There are millions of pulsars in the universe and precise detection can be performed using classification algorithms. This paper discusses several standard classification algorithms such as k-nearest neighbours, logistic regression, support vector machines, decision trees and ensemble of decision trees. These algorithms have been used on a pulsar dataset which contains around 18,000 samples of pulsar candidates out of which about 1,600 candidates are confirmed pulsars. The implementation detail of the algorithms is presented in the paper. The performance of the algorithms discussed here has been compared in terms of accuracy, f-score, specificity, sensitivity and ROC value. The result of the comparisons has been presented in the paper. Implementation of an artificial neural network on the dataset is presented and its performance is compared with the other classification algorithms. Since this is an end-to-end machine learning project, details about the deployment of the algorithms are also presented.

The paper is concluded with a discussion of the performance of the best classifier. Future scope and applications for machine learning algorithms for the detection of pulsar candidates is discussed at the end of the paper.

II. Literature Review

Various authors and researchers have earlier tried to predict pulsars in space. Different studies including those including machine learning methods to detect these pulsars are going on for a while now.

N. Obody [2] in his paper discussed the performance of artificial neural networks and support vector machines(SVM) and their different kernel functions to predict pulsars from the same dataset used in this paper. He presented a detailed study of the two methods. He concluded that both neural networks SVM with linear kernel achieved an accuracy of 98%. Although he concluded that none of the methods stood out to be a better alternative.

Zhen Hong Shang et al.[3] in their paper presented three classification algorithms and discussed their performance for the pulsar detection task. The discussed neural networks, support vector machines and decision tree-based classification techniques.

P.Mounika et al. [4] in their paper discussed the performance of four machine learning models for pulsar classification. They discussed decision tree classifier, support vector machines, random forests and k-nearest neighbour(KNN). They have concluded that the KNN has outperformed the other models.

Amitesh Singh et al. [5] in their work, used the EPN pulsar dataset and used machine learning regression algorithms like decision tree regressor, k-nearest neighbour regressor, support

vector regressor and other algorithms to predict pulsars. They have compared the performance of the regressors to find the best algorithm for the task. They have used FFA techniques for reducing the number of periods.
Many other machine learning algorithms can be used for the above-mentioned task. This project brings all machine learning classification algorithm under one hood and implements them to compare their performance in order find which algorithm will be most suited for the classification of pulsars.

III. Theoritical Background

A. *Pulsars*

After a star of extremely large mass collapses into its own gravity and superheated plasma is expelled at extremely high speeds (also known as Supernova) leading to the formation of an enormously dense object that are made of neutrinos. Pulsars are neutron stars that rotates and emits pulses of radiation at very regular intervals that generally range from milliseconds to seconds. They have very strong magnetic fields that channels jets of particles out of the two magnetic poles. This accelerated light produces very powerful beams of light. When this beam of light crosses our line of sight, we notice a pulse[6]. This is why they seem to flicker or blink. This can be detected in the form of detectable broadband radio emission. These patterns repeat periodically as they rotate rapidly.
Pulsars belong to a family of objects known as neutron stars. Neutron stars are formed when the core massive stars run out of fuel and collapse in an explosion. This stellar phenomenon is the collapse is called a supernova. The neutron stars are a dense chunk of materials left over after the supernova. The only entity denser than a neutron star is a black hole. The magnetic field of a pulsar ranges from 100 million times to 1 quadrillion times that of Earth's.
Pulsars can radiate electromagnetic waves of multiple wavelengths, from simple radio waves to highly energetic gamma rays. According to various researches, it is clear that pulsar radiation is due to its rapid rotation and very strong magnetic field. A spinning magnetic field produces an electric field which results in the acceleration of charged particles. This exact phenomenon takes place in the magnetosphere of pulsars. These accelerated charged particles emit electromagnetic waves. Each pulsar consists of a slightly different emission pattern that varies moderately with each rotation. These signals are detected using a radio telescope and a gamma-ray telescope. A potential positive signal as determined by the length of the observation and averaged over many rotations is known as a 'candidate'. In the absence of additional information, there is potential for each candidate to describe a real pulsar. However, radio frequency interference (RFI) and associated noise originate almost all detections of potential pulsar candidate, making it hard to detect legitimate signals. Studies are going on for a better understanding of the process. The majority of pulsars are detected using the Parkens telescope in Australia. Various other telescopes around the world detect pulsars. So far, over 2,000 pulsars have been detected in total.
The detection of pulsars is very pivotal in the field of astronomy and astrophysics. They have huge potential applications in astrophysical experiments. These include probes of ISM, probes of space-time and gravitational wave detection. Pulsars are also used as precise astronomical clocks in all these experiments. [7]. The variety of applications of pulsars makes it even more important to detect and study the properties of these highly dense entities.

B. *Machine Learning Algorithms*

Machine learning tools are being used now to label pulsar candidate for rapid analysis facilitation on an automatic level. Classification algorithms are used extensively for the binary classification of pulsar candidates. Various classification algorithms that can be used are:

- Logistic Regression
- K-Nearest Neighbor
- Support Vector Machines
- Decision Tree
- Random Forests
- Meta Bagging
- Adaptive Boosting (AdaBoost)
- Gradient Boosting
- Extreme Gradient Boosting (XGBoosting)

Along with these standard classification algorithms, an artificial neural network is implemented as well for binary classification of the pulsar candidates.

C. *Logistic Regression*

Logistic regression is a statistical model used for classification problems in machine learning. It is also called the sigmoid function. There are mainly three types of logistic regression namely binary, multinomial and ordinal logistic regression. The logistic regression uses a sigmoid function based on a given hypothesis to predict the outcome.
Given a hypothesis $y = wx + b$ for a problem, the values are predicted using the function given below.

$$f(x) = \frac{1}{1 + e^{-x}} \qquad (1)$$

The output probability lies in the range of 0 and 1. This equation forms an S-shaped curve which provides non-linearity to the decision boundary. The maximum likelihood method is used to fit the function. Since this is a classification algorithm, a threshold is used to classify the estimated probability into classes. Logistic regression a simple and efficient algorithm that provides a probability score for the observations. This model cannot handle a large number of categorical variables efficiently.

D. K-Nearest Neighbour

K-Nearest Neighbour algorithm is a very simple machine learning algorithm. This supervised learning algorithm classifies data points using the closest observation in the dataset hence the name "nearest neighbours". The value of k signifies the number of neighbours considered to classify the training data. Considering more neighbours leads to a smoother classification boundary which corresponds to a simpler model. Having few neighbours results in a complex model.

$$Euclidean = (\sum_{i=1}^{k} |x_i - y_i|^p)^{\frac{1}{p}} \qquad (2)$$

The primary use of the equation stated above is to calculate the distance between two data points x_i and y_i where k is the number of dimensions which is determined based on the dataset. We set p=2, Euclidean Distance formula.

There are two important parameters of the nearest neighbour classifier: the number of neighbours and how the distance between the data points are measured. This model is generally very fast but can slow down while training a large number of data. The main drawback of this algorithm is its inability to handle many features.

E. Support Vector Machines

Support Vector Machines(SVM) was first introduced by Vapnik[8]. Support vector classifier is also known as soft margin classifier. The purpose of a support vector classifier is to find a separating hyperplane. The smallest distance of training observations from the hyperplane is called a margin. The maximal margin hyperplane is the separating hyperplane is the hyperplane for which the margin is the largest. This hyperparameter is chosen to classify the observations into respective classes. A subset of the observation that lies on the margin or the wrong side of the margin for their respective classes is called support vectors. This support-vector classifier has a tuning parameter C that controls the number of misclassification the classifier will tolerate. C controls the bias-variance trade-off. A small C signifies that narrows the margin and tends to decrease violations of the margin. When C is large, the margin is wide and allows more observations to violate the margin[9]. SVM functions by selecting critical data points from all classes. These are known as support vectors. It then separates the classes by generating a function which divides the samples using these support vectors. SVM is an extension of the support vector classifier from extending the data samples into n-dimensional feature space in a specific form by the use of kernels. A kernel is a function that quantifies the similarities of two observations using the inner products of the feature points. The kernel trick mainly incorporates non-linear boundary between the classes by expanding the feature representation without actually computing the expansions. There are mainly four kernel functions in SVM. They are:

- Linear kernel: $K_{Linear}(x_i, x_j) = \sum_{k=1}^{p} x_{ik} x_{jk}$ (3)
- Polynomial kernel: $K_{Polynomial}(x_i, x_j) = (1 + \sum_{k=1}^{p} x_{ik} x_{jk})^2)$ (4)
- Radial Basis Function: $K_{RBF}(x_i, x_j) = \exp(-\gamma \sum_{k=1}^{p} (x_{ik} - x_{jk})^2)$ (5)
- Sigmoid kernel: $K_{Sigmoid}(x_i, x_j) = \tanh(r + \gamma \sum_{k=1}^{p} x_{ik} x_{jk})$ (6)

The linear kernel uses Pearson correlation to essentially quantify similarities between a pair of observations. The value of d in the polynomial kernel controls the flexibility of the algorithm. The γ in RBF kernel is a constant.

SVM required the pre-processing of the data before training and careful hyperparameter tuning. These models work well with higher dimensional data because they are affected by the data points near the margin. They have longer computational costs as compared to other algorithms and takes a longer time to train the data.

F. Decision Tree Classifier

Decision trees a very powerful supervised algorithms used for both regression and classification. The decision tree classifier is a tree form a tree of questions and classifies the datapoints into respective classes based on the answers of the questions. Each node of the tree forms the most relevant question about the data. The terminal node represents the answer. A recursive process of this task yields a decision tree. The recursive partitioning of the data is repeated until each node of the partition contains the target value. A leaf of the tree that contains datapoints that belong to the same target class is called a pure node. The priority in nodes in the decision tree is set using Gini Index or cross-entropy, it is a score given to the best classifier among the set of attributes.[9]

The Gini index is represented by (7) represents the node purity.

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}) \qquad (7)$$

Where $\hat{p}_{mk}$ represents the fraction of training data in the mth region that belongs to the kth target class. A small value of Gini index represents a node that contains observations from a single class.

The cross-entropy is represented by (8).

$$D = -\sum_{k=1}^{K} \hat{p}_{mk} log \hat{p}_{mk} \qquad (8)$$

The cross-entropy will have a small value If the mth nod is pure.

The complexity of a decision tree is controlled by pruning the tree. Pruning of a tree involves controlling the depth of the tree, number of nodes, etc.

An advantage of tree-based model is that it can be easily visualized and understood. Decision trees are independent of the feature scaling of the training data. A disadvantage is that the decision tree tends to overfit the training data.

G. Random Forests

Random forests are an ensemble of decision trees. Random forests are essentially a bagging method. Random forests is essentially a collection of a number of decision tree where each tree differs from each other. The basic idea behind the decision tree is that each tree will classify data well and have a fair share of overfitting. Their overfitting can be reduced by aggregating and averaging their combined performance. Random forests reduce the correlation of the trees. It combines both the concepts of bagging and random feature subspace selection which makes these robust models.

H. Boosting

The main intuition behind boosting is to iteratively fit the model on the data in order to reduce the bias. In boosting, each model in the sequence is fitted giving more importance to the training observation that was not handled well by the previous models. The basic idea is behind boosting is to train a number of weak learners where each tries to correct its predecessor in order to form a strong learner There are primarily three boosting algorithms, They are, Adaptive Boosting, Gradient Boosting and XGBoosting algorithm.

1) Adaptive Boosting:

Adaptive boosting also known as Adaboost basically combines multiple weak learners to form a strong learner. The mentioned weak learners are essentially decision trees with a single split called decision stumps. In the first decision stump of Adaboost, all the observations weighted equally. More weights are carried by the observation to correct the incorrect classification in the previous error. In this way, the model will update the weights until the most accurate predictor is obtained.

2) Gradient Boosting:

In Gradient Boosting, instead of changing the weights of observation for incorrect classification in the previous predictors, it aims to fit the new predictor to the residual errors of the previous predictor. It uses the gradient descent method to identify the shortcomings of the previous predictors. By accounting for the errors of weak learners of all the previous layers, the final predictive model is able to reduce the error over time.

3) XGBoosting:

XGBoost is an ensemble of decision tree algorithm. It stands for eXtreme Gradient Boosting. It is an implementation of a gradient boosting framework but with improved performance. Gradient Boosting models are slower because of sequential implementation and are not scalable. XGBoost improves the computation speed and improves the model performance. The main features of XGBoost are parallelization, distributed computing, out-of-core computing and cache optimization.

IV. METHODOLOGY

A. The Dataset

The dataset used for the detection of pulsars contains 17,898 pulsar candidates[10]. The dataset is available at[11]. Out of these 17,898 candidates, 16,259 are spurious samples caused by noise or radio frequency interference(RFI) and the rest 1,639 are confirmed pulsar samples. Each sample in the dataset is described by 8 continuous features. The first four feature are statistics procured from the integrated profile of pulse. This is an sequence of variable which is continuous in nature that describe a longitude-resolved version of the signal that has been averaged in both time and frequency. The remaining four feature variables are similarly obtained from the DM-SNR(Dispersion Measure- Signal to weight Ratio) curve[12]. Signal-to-Noise Ratio (SNR), is a quantity of signal strength proportional to the background noise. The 8 features are summarized below:

- 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.

The dataset is presented in CSV format with each column representing the 8 feature variables and the 9th column specifies the class label where 0 and 1 represent negative and positive samples respectively. The data present in the dataset is feature data extracted from candidate files using the PulsarFeatureLab tool[13].

In the dataset, about 91% of the observed samples are non-pulsar stars and about 9% are pulsar stars. It evident that this dataset is a very unbalanced dataset with a large number of observations belonging to a single class.

The density distribution of the eight features mentioned above is visualized in figure 1. It is visible from the figure that the mean, standard deviation, excess kurtosis and skewness of both the integrated profile DM-SNR curves of the pulsar candidates are not distributed normally. The mean of integrated profile feature is slightly negatively skewed. The mean, standard deviation, excess kurtosis and skewness of DM-SNR curve is positively skewed. The excess kurtosis and skewness of integrated profile is also positively skewed. The standard deviation of integrated profile is slightly negatively skewed.

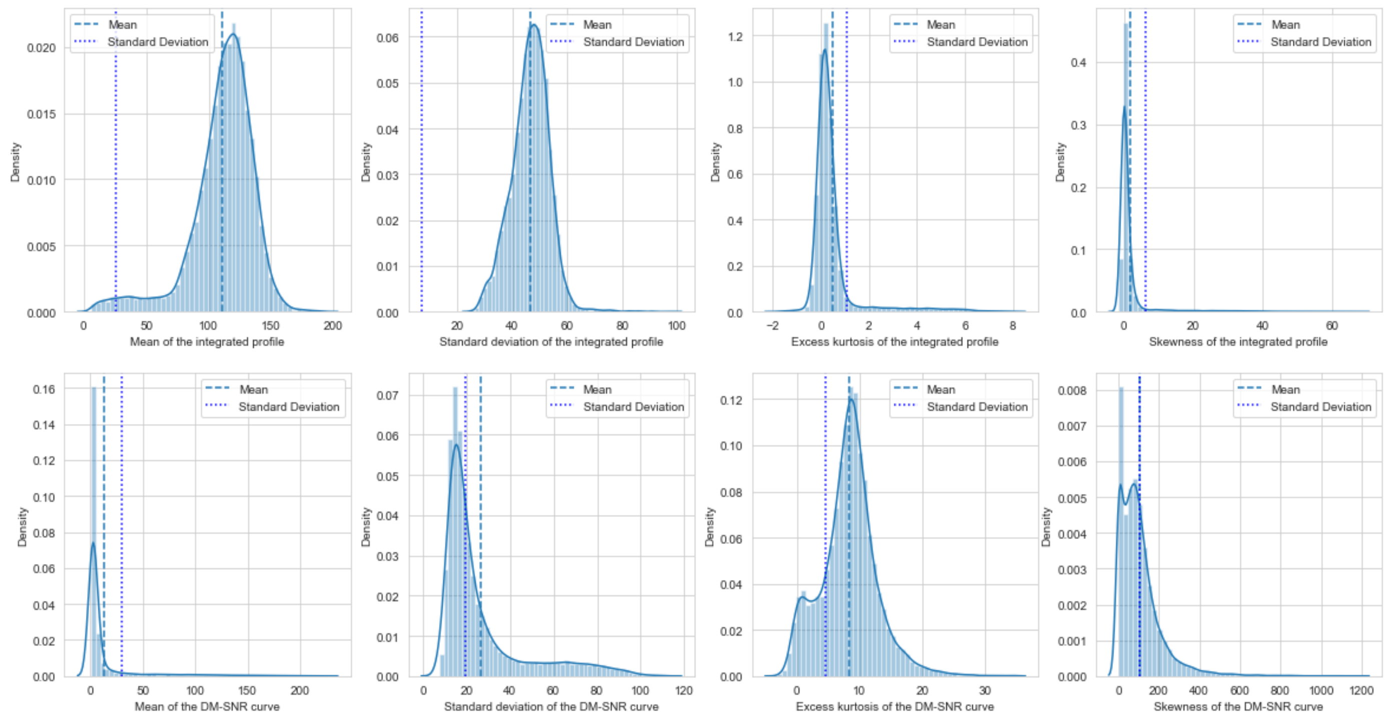

Figure 1. Density Distribution of the eight features in the dataset.

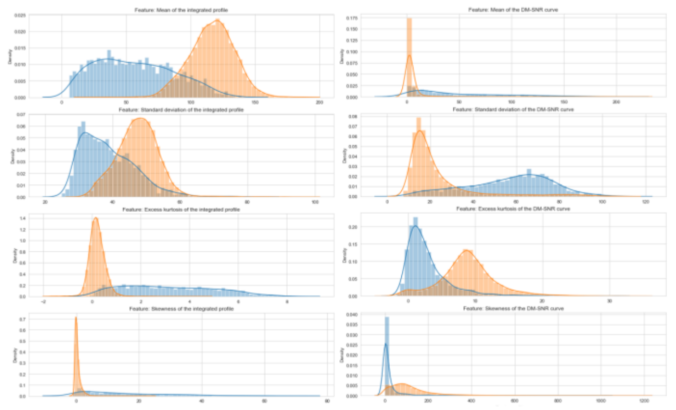

Figure 2. Density Distribution each class of the eight features in the dataset.

B. *Data Pre-Processing*

The training set for the implementation of machine learning models contains 90% of the total samples, i.e. 16,108 observations. Rest 10% samples are included in the test set. The training set is a class imbalanced dataset with a majority of the observation belonging to the non-pulsar class. Since the data are normally distributed, no additional pre-processing such as normalization and standardization is required. The models built are compared using the k-fold cross-validation method. In this paper, k=10 is considered for the cross-validation method.

C. *Balancing the dataset using SMOTE*

The fact that the dataset used is unbalanced, any classification algorithm will perform poorly on the minority class while learning to create a decision boundary. This problem can be solved by oversampling the data in the minority class of the dataset. This is done by duplicating the data of the minority class prior to training and fitting of the model. This is a kind of data augmentation. The new synthesized data does not provide any additional information to the model. The commonly used oversampling technique is Synthetic Minority Oversampling TEchnique also known as SMOTE[20]. SMOTE works by selecting data examples that are close to the feature space. The effectiveness of the model is due to the fact that it is plausible and the data points are close to the feature space.

D. *Model Implementation*

The machine learning models for the prediction of pulsars are implemented using the Scikit package in the Python programming language[14]. The algorithms mentioned above are implemented and compared using a 10-fold cross-validation method. Optimal hyperparameters of the models are tuned for the best performance of the models.

K-nearest neighbour algorithm is implemented using "n_neighbour"=7. This implies that 7 neighbours are considered for the classification task. The distance between the observation is calculated using equation (2) where p=2, i.e. Euclidean distance. Support vector classifier is implemented with C=1.0. The linear kernel is used to train the model as it showed better results upon comparing with other kernels. The decision tree classifier is implemented with a maximum depth of 6 and an entropy criterion.

In the random forest algorithm, the "max_feature" is set to 100 which defined the number of trees used to construct the random forest. Also "max_depth"=6 is set for the training of the model which defines the depth of the tree. Using "max_depth"=6, overfitting of the model is avoided. The adaptive boosting algorithm is implemented using "n_estimate"=100, with a learning rate of 1.0 and the decision tree as a base estimator. The gradient descent boosting classifier is implemented using "n_estimate"=100, A "max_depth" of 1.0. a "learning _rate" of 1.0 is used while training the model. XG boosting algorithm is implemented with "mlogloss".

All these algorithms are implemented using the Scikit learn package[13]. Performance of all the presented algorithms are compared using the 10-fold cross-validation method and the comparison of performance is presented in the following section.

V. Result And Observation

The classification algorithms mentioned above are implemented on the unbalanced training set and their performance is compared using 10-fold cross-validation. Their performance is compared using five classification metrics. The classification accuracy, f-score, Precision, recall and ROC value of the classification models are compared. Table I summarized the performance comparison of the algorithms.

The hyperparameters or the models are optimized to their best values. Since the dataset is class-imbalanced data, calculating only the classification accuracy of the models is not enough for efficient comparison of the algorithms. Other classification metrics are taken into account for efficient comparison.

TABLE I. PERFORMANCE COMPARISON OF NINE ALGORITHMS ON UNBALANCED TRAINING SET

Algorithms	**Accuracy**	**F-score**	**Precision**	**Recall**	**ROC**
Logistic Regression	0.9783	0.8715	0.9381	0.8151	0.9735
K-Nearest Neighbour	0.9729	0.8382	0.9128	0.7755	0.9360
Support Vector Machine	0.9788	0.8743	0.9417	0.8168	0.9737
Decision Tree	0.9787	0.8742	0.9206	0.8352	0.9630
Random Forest	0.9787	0.8744	0.9375	0.8191	0.9725
Bagging	0.9787	0.8809	0.9291	0.8351	0.9529
XGBoost	0.9797	0.8825	0.9261	0.8437	0.9745
Adaptive Boosting	0.9692	0.8331	0.8252	0.9372	0.9116
Gradient Boosting	0.9768	0.8635	0.9259	0.8100	0.9208

From table I, it can be observed that the classification accuracy of the XGBoost classifier is comparatively higher than the rest of the algorithms. Support vector machine, decision tree, random forests and bagging algorithms show very similar accuracy and not much less than XGBoost. Among all the classifiers that are compared using 10-fold cross-validation, the classification accuracy of Adaptive boosting is the least. The almost similar accuracies of the models attribute to the fact that the dataset is normally distributed and the models are optimized using appropriate hyperparameters.

XGBoosting algorithm outperforms others in comparison of f-score with a score of 88.25. The f-score of the Adaptive boosting model is the least. Precision and recall values are maximum for SVM and XGBoosting respectively. The ROC score for the XGBoosting model is the greatest among the models with a score of 97.45. The ROC score of Logistic regression, SVM and random forests is also almost the same as the XGBoost model.

A. SMOTE-balanced training set

The classification algorithms mentioned are also implemented on the SMOTE-balanced training set and their performance is compared using 10-fold cross-validation. Their performance is compared using five classification metrics. The classification accuracy, f-score, Precision, recall and ROC value of the classification models are compared. Table II summarized the performance comparison of the algorithms.

From table II, it can be observed that the classification accuracy of the XGBoost classifier is comparatively higher than the rest of the algorithms. Support vector machine, decision tree, random forests and bagging algorithms show very similar accuracy and not much less than XGBoost. Among all the classifiers that are compared using 10-fold cross-validation, the classification accuracy of logistic regression is the least.

TABLE II. PERFORMANCE COMPARISON OF NINE ALGORITHMS ON SMOTE-BALANCED TRAINING SET

Algorithms	**Accuracy**	**F-score**	**Precision**	**Recall**	**ROC**
Logistic Regression	0.9425	0.9408	0.96.81	0.9150	0.9781
K-Nearest Neighbour	0.9548	0.9556	0.9385	0.9734	0.9866
Support Vector Machine	0.9448	0.9428	0.9766	0.9113	0.9786
Decision Tree	0.9468	0.9457	0.9670	0.9251	0.9835
Random Forest	0.9573	0.9559	0.9792	0.9334	0.9929
Bagging	0.9721	0.9721	0.9791	0.9672	0.9922
XGBoost	0.9732	0.9732	0.9746	0.9719	0.9962
Adaptive Boosting	0.9603	0.9612	0.9566	0.9668	0.9600
Gradient Boosting	0.9481	0.9471	0.9654	0.9295	0.9867

XGBoosting algorithm outperforms others in comparison of f-score with a score of 97.32. The f-score of the logistic regression model is the least. Precision value of random and bagging is highest with a value of 0.979 and recall values is maximum for KNN with a value of 97.19. The ROC score for the XGBoosting model is the greatest among the models with a score of 99.62. The ROC score of random forests and bagging is also almost the same as the XGBoost model.
After balancing the dataset using SMOTE technique, the accuracy, f-score, precision, recall and ROC have shown an increase in performance. Comparison of classification accuracy, f-score, Precision, recall and ROC value is visualized in figure 3.

From table I, table II and figure 3, it can be observed that the performance of the classification models is better in SMOTE-balanced training set compared to unbalanced training set. The 10-fold CV classification accuracy of the models on unbalanced data is better than accuracy on the SMOTE-dataset. The f-score, precision recall and ROC value of the models is better than on the SMOTE-balanced data than on the unbalanced data. The fact that that it is the classification algorithm performs better on the SMOTE-balanced data makes the results more desirable than the results on the unbalanced data. . Among all the classification models in all both the training data, XGBoosting performs the best with the highest accuracy, f-score, precision, recall and ROC value.

From the above discussion, it can be observed that the performance of the XGBoost algorithm is the best among nine others on the considered dataset. XGBoost has the largest accuracy, f-score, recall and Roc score with almost the greatest precision value. The average runtime of the algorithm is fairly low as compared to others.

B. A Neural Network Approach

A different approach was considered for the classification of pulsars and non-pulsars from the mentioned dataset using artificial neural networks. Neural networks are robust classifiers and can provide better performance relative to the classical machine learning classifiers.

A neural network model is developed for this classification problem using Keras Sequential API[15]. The model consists of 9 layers with a total of 1,265 parameters. Dropout layers are used to avoid overfitting of the model Since it is a binary classification problem, a sigmoid activation function is used on the output layer. 90% of the observations are used for the training of the model and the rest 10% is test set. 10% of the train set is used as a validation set for the model validation.
Adaptive Moment Estimation(Adam) optimizer[16][17] with its default learning rate is used for the model compilation. Since this is a binary classification problem, binary cross-entropy loss [17] is used for monitoring the performance. The model is trained up to 100 epochs using a batch size of 32 . The model achieved a training accuracy of 97.72% and the loss value is 0.0799. The model also achieved a validation accuracy of 98.28%.

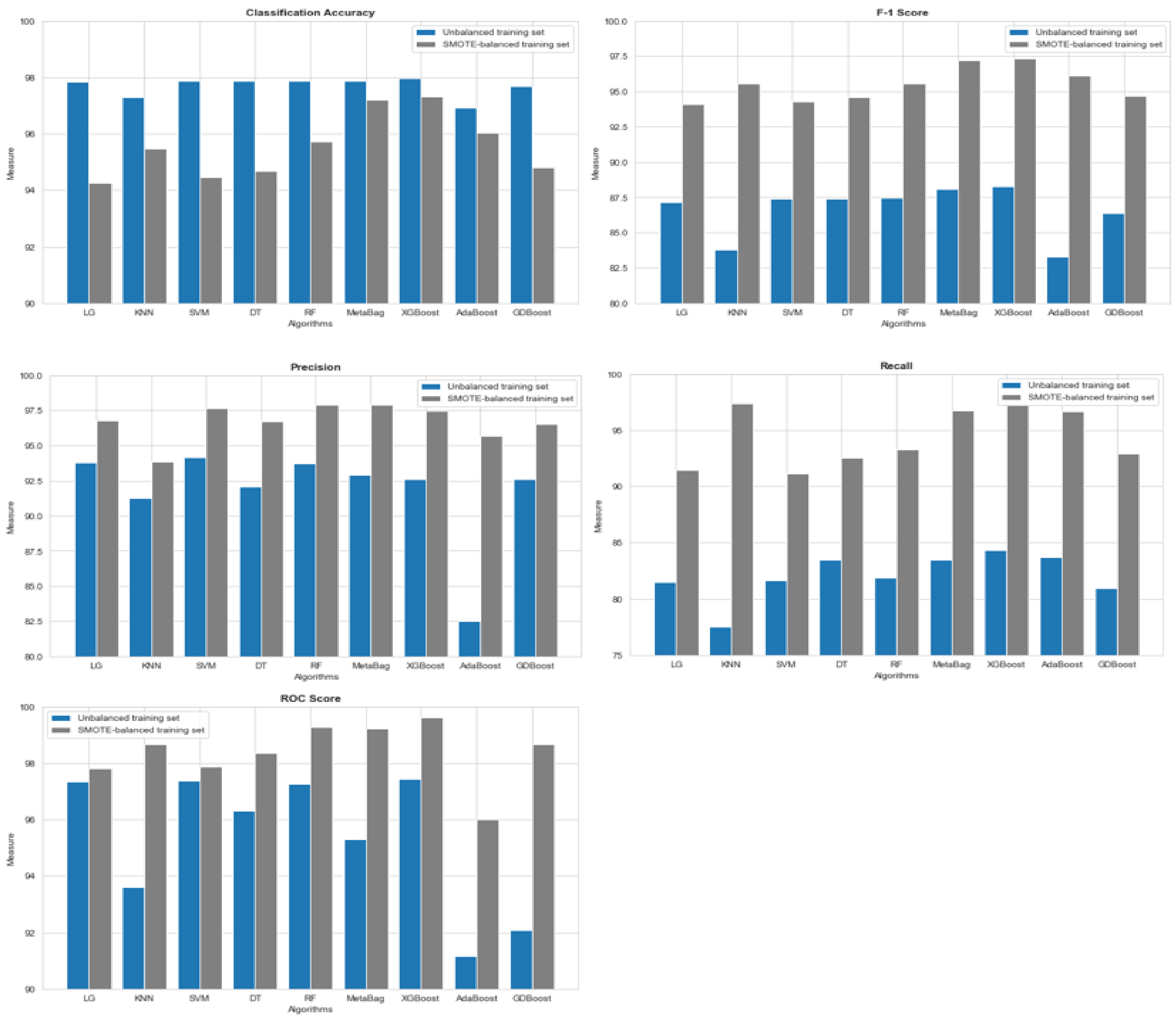

Figure 3. Accuracy, f-score, Precision, Recall, ROC score and average computation time of the nine algorithms.

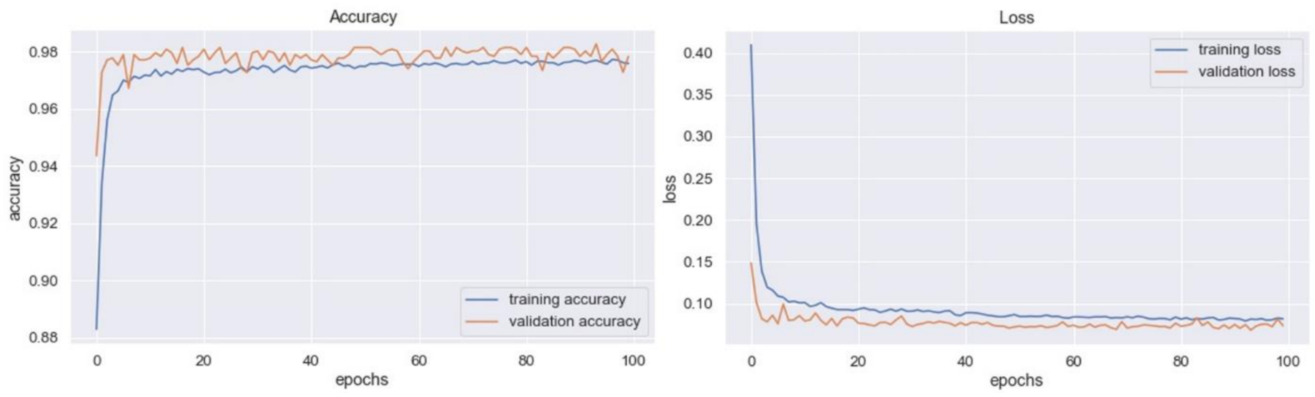

Figure 4. (Left) Training and Validation Accuracy, (Right) Training and Validation Loss score

The performance of the neural network model on both the train and validation set is plotted in figure 4. It can be observed from the figure that the difference between the training accuracy validation accuracy is very small near the 100th epoch, which proved that the model does not overfit. The training and validation loss values seem to converge at the 100th epoch which is a good result for the model.

VI. CONCLUSION

Pulsars are the most studied natural phenomena in the field of radio astronomy. The results obtained in this project proves that machine learning algorithms have a huge potential for the detection of pulsar stars. It can be attributed to its high accuracy and negligible prediction time.

In conclusion, this project implemented the nine most common classification algorithms and compared their performance on 10-fold cross-validation using five classification metrics such as classification accuracy, f-score, precision, recall and ROC value. The classification algorithms were implemented on both unbalanced and SMOTE-balanced data. The performance of the algorithms on SMOTE-balanced data is better than that on the unbalanced data. Among the nine algorithms on both the type of training data, the XGBoosting method outperformed others in almost every aspect with the highest accuracy of almost 98%, while the other models are only marginally terrible than the XGBoost model. XGBoosting also had better performance in f-score, precision, recall and ROC values. A 9 layered artificial neural network was also implemented which showed an accuracy of almost 98% with no overfitting. Thus, it can be concluded that neither of them stands out as a better alternative than the other. Implementation of any one of them will provide the best result in pulsar detection[19].

The results presented in this paper can help in further research in the field of radio astronomy for pulsar detection. Further development on the use of machine learning will speed up the process with improved accuracy. On the off chance that machine learning models are extensively applied for the task, the results presented here shall prove to be extremely helpful.

REFERENCES

[1] https://imagine.gsfc.nasa.gov/science/objects/neutron_stars1.html

[2] https://d1b10bmlvqabco.cloudfront.net/attach/jz8smbptoj35ra/i8xgc5x4yhoyo/k0fny104qt72/project.pdf

[3] Cheng Jun Zhang, Zhen Hong Shang, Wan Min Chen, Liu Xie, Xiang Hua Miao. (2020). A Review of Research on Pulsar Candidate Recognition Based on Machine Learning. Procedia Computer Science, Volume 166, 2020, Pages 534-538, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2020.02.050.

[4] https://www.irjet.net/archives/V7/i6/IRJET-V7I61195.pdf

[5] Singh, Amitesh & Pathak, Kamlesh. (2020). A machine learning-based approach towards the improvement of SNR of pulsar signals.

[6] https://imagine.gsfc.nasa.gov/science/objects/neutron_stars1.html

[7] http://www.scienceguyrob.com/wpcontent/uploads/2016/12/WhyArePulsarsHardToFind_Lyon_2016.pdf

[8] Vapnik, V. N. (1998). Statistical Learning Theory. Wiley-Interscience.

[9] Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani. An Introduction to Statistical Learning : with Applications in R. New York :Springer, 2013.

[10] R. J. Lyon, B. W. Stappers, S. Cooper, J. M. Brooke, J. D. Knowles, Fifty Years of Pulsar Candidate Selection: From simple filters to a new principled real-time classification approach MNRAS, 2016.

[11] https://archive.ics.uci.edu/ml/datasets/HTRU2

[12] R. J. Lyon, "Why Are Pulsars Hard To Find?", PhD Thesis, University of Manchester, 2015.

[13] R. J. Lyon, "PulsarFeatureLab", 2015, https://dx.doi.org/10.6084/m9.figshare.1536472.v1.

[14] https://scikit-learn.org/stable/supervised_learning.html#supervised-learning

[15] https://keras.io/api/models/sequential/

[16] Kingma, Diederik & Ba, Jimmy. (2014). Adam: A Method for Stochastic Optimization. International Co ference on Learning Representations.

[17] https://keras.io/api/optimizers/adam/

[18] https://keras.io/api/losses/probabilistic_losses/#binarycrossentropy-class

[19] https://github.com/apratimsadhu01/Pulsar-Star-Detection

[20] Chawla, Nitesh & Bowyer, Kevin & Hall, Lawrence & Kegelmeyer, W.. (2002). SMOTE: Synthetic Minority Over-sampling Technique. J. Artif. Intell. Res. (JAIR). 16. 321-357. 10.1613/jair.953.