Pulsar star prediction using machine learning

Mario Jafet Aviles Blanco, *Mechatronic*, Javier Gerardo Urrecha Zambada, *Mechatronic*, and Raul Alejandro Gonzalez Gallo, *Mechatronic*.

Abstract—A pulsar star is a highly magnetized, rotating neutron star that emits beams of electromagnetic radiation out of its magnetic poles that can be observed from Earth when a beam of emission is pointing towards it. In the modern area of Big Data, research in astronomy and astrophysics provides enormous volumes of data about pulsar stars that can be analyzed to study physical phenomena such as the interstellar medium, black holes, antimatter and possible tests of a new theory of general relativity. The problem arises with the amount of computational cost and time it takes to analyze these massive amounts of data to correctly predict which observations actually come from a pulsar star. To contribute with the rigorous research on pulsar stars and help reduce the computational costs and time, we intend to find best supervised machine learning binary classification technique to classify a pulsar candidate as a real pulsar star or as radio frequency interference using the HTRU2 data set which describes a sample of pulsar candidates collected during the High Time Resolution Universe Survey. For this, we created and compared the results of five of the most famous classification algorithms: Logistic Regression, Support Vector Machine, K Nearest Neighbors, Neural Networks and Random Forest. Unlike other works that rely their results on the accuracy of their models, which doesn't represent the general performance of the model, we based our results on a combination of the specificity and sensitivity, which can both be represented using the Receiver Operating Characteristic Curve and the area under this curve. After seeking and obtaining the best hyperparameters that worked for each algorithm, we found that the model that gave the best overall performance results was Logistic Regression with a specificity of 0.9649 a sensitivity of 0.9355 and an area under the curve of 0.9794.

1 Introduction

Pulsars are rapidly rotating, highly magnetized, neutron stars that periodically emit a beam of electromagnetic radiation detectable from Earth. A simplified schematic of a pulsar is shown in figure 1. Since the discovery of the first pulsar star [1], research on these exotic cosmic phenomena has been of considerable scientific interest since they serve as probes of space-time, help study the inter-stellar medium as well as the states of matter and contribute to a wide range of other physics and astrophysics applications [2].

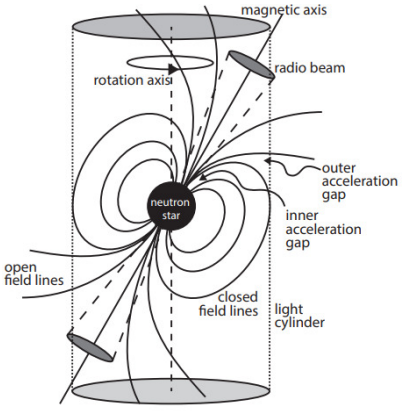

Fig. 1: Simplified schematic model of a pulsar

Searching for a pulsar star involves looking for a periodic radio signal, known as a pulse, with large radio telescopes, however, since the strength of an individual pulse is usually very weak, astronomers must combine many pulses together with respect to its rotational period, in a process known as folding, in order to build up a detectable signal, known as an integrated pulse profile (see [3] for a complete explanation on integrated pulse profile). Even though the individual pulses vary slightly with each rotation, the integrated pulse profile is usually very stable for any observation at the same radio frequency, thus it can be thought of as a unique "fingerprint" showing a cross-sectional cut through each neutron star's emission beam. Figure 2 shows the integrated pulse profiles for a sample of nine pulsars. (For more on pulsar stars see the following [2])

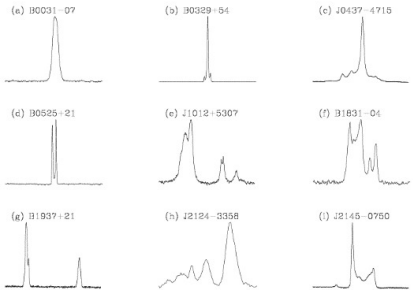

Fig. 2: Integrated pulse profiles for a sample of nine pulsars

1.1 Stating the problem

The problem arises when we try to find an efficient method to identify a signal as a pulsar. As studies have shown [4] pulsars are hard to find due to their rare nature and to the amount of data obtained in an observation, which not only carries information about possible pulsar star, but

also a lot of radio frequency interference (RFI). The search for pulsars was undertaken using a combination of large radio telescopes, signal processing algorithms, and human ingenuity; however, results have shown that this approach was both time and effort consuming. Nowadays, machine learning tools are being used to automatically label pulsar candidates to facilitate rapid analysis [5], [6], [7]. Each candidate is a possible detection of a pulsar signal, which exhibits specific characteristics of interest. Classification systems are being widely adopted, which treat the candidate data sets as binary classification problems. The problem we are trying to solve is determining if a signal detected from a candidate is an actual pulsar star or just RFI, based on machine learning techniques using the characteristics obtained.

1.2 Previous studies

Many studies have been made to try to predict pulsar stars, some even using the HTRU2 data set. All studies decided to use binary classification algorithms.
In May 2016, a thesis about Discovering Pulsars with Machine Learning by Vincent Morello was presented at the Faculty of Science, Engineering and Technology at Swinburne University in fulfillment of the requirements of the degree of Master of Science.
In this thesis [8], he presented a pulsar candidate classifier named SPINN that uses an artificial neural network, a type of Machine Learning (ML) algorithm. He argued that the accuracy reached by previous implementations has been insufficient, either missing discoveries or still leaving an excessive number of spurious candidates to be rejected by a human operator.
He emphasized on how to properly represent pulsar candidates as vectors of real-valued inputs (or features) to the algorithm. A detailed understanding of the entire pulsar searching process is helpful, if not necessary to design optimal features. He also went one step back and entirely rewrote the software responsible for producing the candidate plots, as their quality significantly impacts the accuracy of the classifier that rates them.
We will not be using neural networks as the main algorithm, but this thesis is helpful to understand the importance of knowing most of the supervised learning algorithms that were used by different people to solve the problem, and later decide which one we will be applying.
We also analyzed the research done by John. M Ford, which is a document product of extensive research conducted at the Nova Southeastern University College of Engineering and Computing about Pulsar Search Using Supervised Machine Learning.
In this paper [9] he explained that searching for pulsars is a very labor-intensive process, currently requiring skilled people to examine and interpret plots of data output by analysis programs. An automated system for screening the plots would speed up the search for pulsars by a very large factor.
In his research he concluded that while a false positive rate of 1a false positive rate of less than 2the imbalanced training and test data were explored and found to be highly effective in enhancing classification accuracy.
One of our extracting data problems was that we did not have enough test data to validate our models, so we needed to balance the train and the test data using oversampling and under sampling methods by generating new synthetic data. John's method to mitigate the imbalance data would be beneficial to our work.
In the article [10] "Separation of pulsar signals from noise using supervised machine learning algorithms", published by arXvi in 2018, Suryarao Bethapudi evaluate the performance of four different machine learning algorithms: an Artificial Neural Network Multi-Layer Perceptron, Adaboost, Gradient Boosting Classifier, and XGBoost, for the separation of pulsars from radio frequency interference and other sources of noise. He used the Synthetic Minority Over-sampling Technique to deal with high-class imbalance in the dataset, which is almost the same solution method we are using, by first dealing with the imbalanced data and then validating the supervised learning algorithms so we can select the best model.

It was difficult to find papers using logistic regression, k nearest neighbors and support vector machine as their main algorithms, but one of the principal kernels presented by Efe Ergun [11] in Kaggle used all of them, with methods just barely different from ours, but not solving the problem of the imbalance data.

Other reference written by Xinwei Wang, Yue Yu and Zihan Zheel [12] that better fits.

In the work done by Yuki Fujimoto [13] , the neural network used ReLu and tanh in almost all layers. We preferred using the sigmoid function as it gave us better results.

One last paper where they used convolutional neural networks lead by W. Zhu. [14] In our solution was not included the image patter recognition, but it is interesting to discuss that instead of analyzing data from the signals, It could be analyzed directly of the image of the signal.

2 Methods

First, we extracted the 17,898 samples from the CSV file uploaded on [4] and selected the first 8 columns for the independent variables or features and used the 9 th column as the target class using the pandas library from Python. Then we randomly divided the dataset into seventy percent for training data and the remaining thirty percent for test data. Next, we needed to verify that out training set was balanced so that our algorithms could both learning to identify non- pulsar stars and correctly identify pulsar stars. As shown in figure 3, our training set was unbalanced since most of our data are non-pulsar candidates. We decided to balance our training set so that our algorithms could learn correctly. We considered three sampling techniques to balance out our data [15]:
Undersampling: it is going to reduce the class 0 randomly until we have the same number of class 0 and class 1. It works, but we lose information.
Oversampling: it is going to do the opposite of

undersampling, it is going to duplicate randomly observations of class 1 until we have the same number of class 0 and class 1. It works and we don't lose information, but we can easily overfit here.
Synthetic Data Generation: it is kind of oversampling data, but it does not replicate existing data. While undersampling and oversampling work on the observations, synthetic data generation works on the data of the observation. It creates artificial data for the class 1 using bootstrap and KNN.

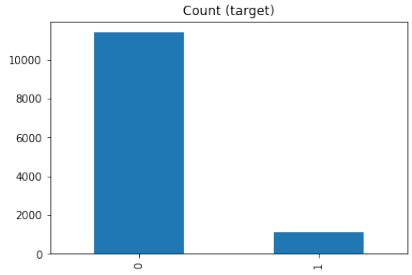

Fig. 3: Unbalanced data

For our study, we made a combination of oversampling and undersampling techniques using the combine imbalanced learning function SMOTEENN from the library imblearn.combine to balance our data. Figure 4 shows the balanced training data.

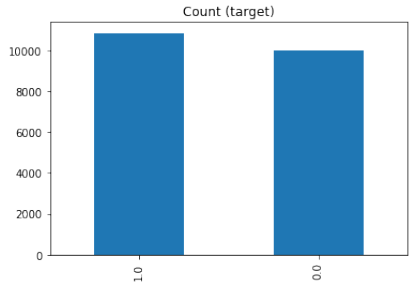

Fig. 4: Balanced data

Finally, once our training dataset was balanced, we could normalize each feature of our training and testing datasets to facilitate the generation of the algorithms.

In this study we decided to measure five evaluation metrics: accuracy, precision f1 score, sensitivity and specificity, as well as the area under the ROC curve and the logarithmic loss. To choose the best algorithm, we focused on the sensitivity and specificity, which can be described using the ROC Curve and the AUC, since these metrics can show the general performance of the model better than the accuracy and precision. The confusion matrix, logarithmic loss, area under the curve and ROC curve were obtained for each algorithm using the sklearn.metrics library. Finally, to choose the best threshold for each algorithm, we obtained the true positive rates and the false positive rates and the thresholds from the metrics library, subtracted the false positive rate from the true positive rate and found the position of the highest result of the subtraction, which corresponds to the position of the best threshold.

Algorithm	Regularization
Logistic Regression	L1 Lasso Regularization

Algorithm	Kernel
Support Vector Machine	Linear Kernel

Algorithm	# of neighbors
K Nearest Neighbors	42

Algorithm	# of neurons per layer
Neural Networks	1st layer: 8 2nd layer: 9 3rd layer: 5 4th layer: 2 5th layer: 1

Algorithm	Criterion	# of trees
Random Forest	Gini Impurity	123

TABLE 1: Data

2.1 Logistic regression

To implement logistic regression, the LogisticRegression function was imported from the `sklearn.linear_model` library. The logistic regression model uses a logistic function, or sigmoid function, to predict the output based on weights attributed for each feature of the dataset. First, the algorithm was implemented using the lasso regularization. The ROC Curve and AUC obtained were:

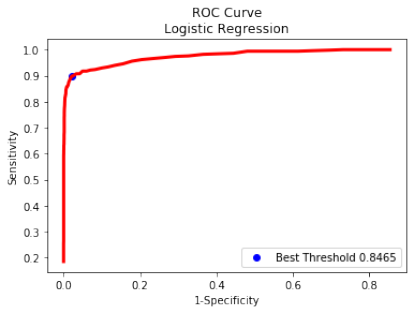

Fig. 5: ROC curve for linear regression

In the next algorithm, we change the regularization to the ridge regularization and obtained the following results:

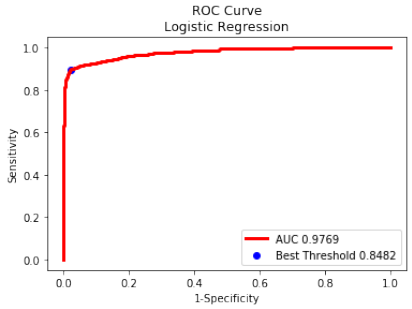

Fig. 6: ROC curve for linear regression with librarys

As we can see in the graphs above, the model with the lasso regularization gave a higher AUC, resulting in the

best hyperparameter to implement this algorithm.

2.2 K nearest neighbors

For this method, the KNeighborsClassifier function from the sklearn.neighbors library was used. First, the model was trained with k=5, just to prove the model worked and predicted correctly. After showing that the metrics and the confusion matrix indicate its reliability, the model was trained with one hundred different values of k, iterated between 1 and 100, to choose the best number of neighbors. In this case, we chose the number of neighbors that gave the highest F1 Score. A graph of the F1 Score in function of k is shown in figure 7, were the blue dot represents the value of k that yields the highest F1 Score.

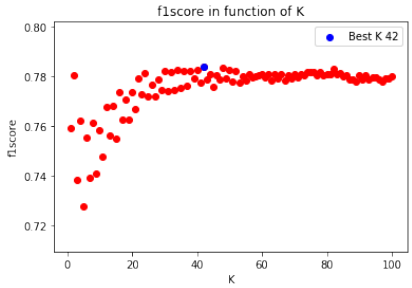

Fig. 7: F1 Score in function of K

The number of neighbors that gave the highest F1 Score was 42. We trained the model and obtained the following ROC Curve and AUC:

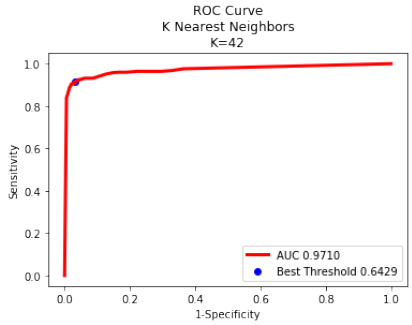

Fig. 8: ROC curve of K nearest neighbors with K=42

2.3 Support vector machine

For the implementation of the Support Vector Machine algorithm, we used the library SVM from sklearn.svm. The objective of this algorithm is to find the hyperplane in an N-dimensional space, that has the maximum margin, i.e. the maximum distance between data points of both classes, and distinctly classifies the data points. For this algorithm, we only changed the type of kernel it used. First, we used a linear kernel to see if the data was linearly separable and obtained the following ROC Curve and AUC:

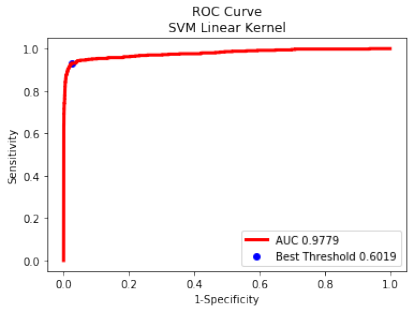

Fig. 9: ROC curve of a linear kernel

Then, we changed the kernel to radial basis functions to see if the algorithm improved if we used infinity dimensions.

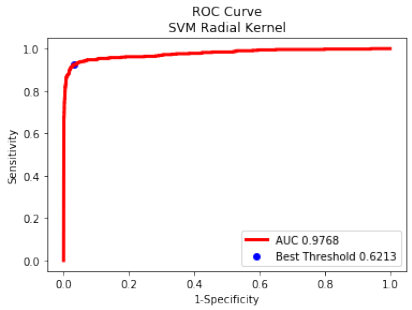

Fig. 10: ROC curve of a radial kernel

2.4 Random forrest

For the Random Forest algorithm, we used the library RandomForestClassifier from sklearn.ensemble. First, we made the algorithm with the information gain/entropy criterion and 100 decision trees to test the functionality of the algorithm. Then, we iterated the number of trees from 64 to 128 since, as previous woks have shown[n], this range of trees results in the best combination of performance and computational cost. For each number of trees, we used both criterion: the Gini Impurity and the information gain/entropy and chose the combination of hyperparameters that yield the highest AUC. In this case, the best combination was 123 trees using the Gini Impurity criterion. The results were as follow:

2.5 Neural networks

The neural network was created using the sequential model from Keras.
This classification problem is quite easy, so a neural network with a simple architecture was created. It was defined with 5 layers, were the neurons of the first three were iterated during the optimization, the first and second layers were tested between 8 and 10 neurons, while the third layer was tested between 4 and 6 neurons. We obtained the f1 score of each combination of neurons and chose the combination that gave the highest f1 score.

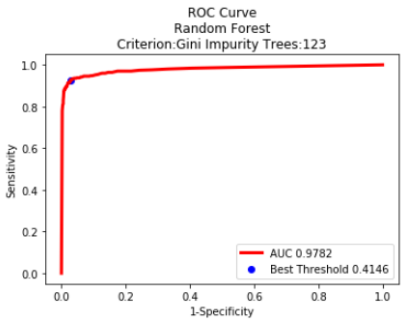

Fig. 11: ROC curve of random forest

All the layers used a sigmoid activation, loss was obtained with binary cross entropy, and adam optimizer with beta 1 = 0.9, beta 2 = 0.999 and alpha = 0.002 was used.
Based on the f1 scores of the neural networks, the best combination of neurons was 8 for the first layer, 8 for the second layer and 5 for the third layer. The fourth layer had 2 neurons and the fifth layer just 1 neuron.
A graph with the ROC curve and AUC of the model is presented as follows:

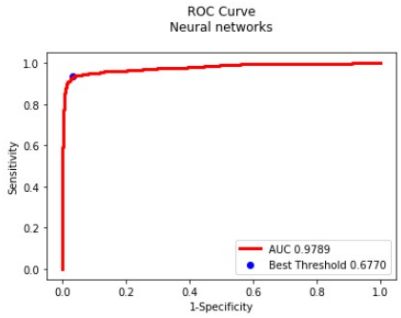

Fig. 12: ROC curve for Neural Network

3 Results

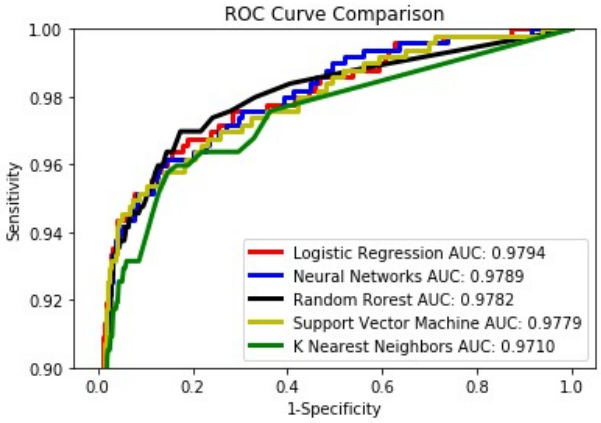

Fig. 13: Comparison of all ROC curves

Metrics comparison with work done by wang 3

Algorithm	Accuracy	Precision	Sensitivity	Specificity	F1 Score	Log Loss
Logistic Regression	0.9622	0.7307	0.9355	0.9649	0.8205	1.3057
SVM	0.9706	0.7894	0.9294	0.9748	0.8537	1.0162
KNN	0.9529	0.6800	0.9254	0.9557	0.7839	1.6273
Neural Networks	0.9454	0.6369	0.9415	0.9458	0.7612	1.8846
Random Forest	0.9710	0.8004	0.9133	0.9768	0.8531	1.0034

TABLE 2: Results

	Accuracy	Negative Precision	Positive Precision	Negative Recall	Positive Recall
LR	0.972975	0.9656244	0.9806157	0.9810514	0.9648466
RF	0.988444	0.9847004	0.9922729	0.9923834	0.9844801
CNN	0.983692	0.9751824	0.9925728	0.9927549	0.9745699
MLP	0.985742	0.9768417	0.9950447	0.9951699	0.9762528

TABLE 3: Results

3.1 Logistic regression

see figure 14

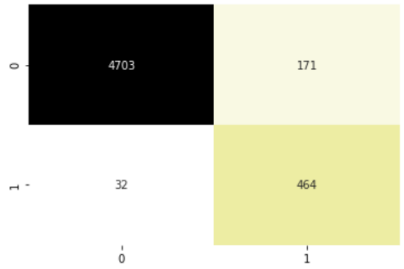

Fig. 14: Logistic regression confusion matrix

3.2 K nearest neighbors

see figure 15

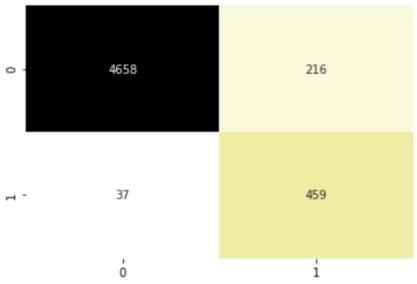

Fig. 15: K nearest neighbors confusion matrix

3.3 Support vector machine

see figure 16

3.4 Random forrest

see figure 17

3.5 Neural networks

see figure 28

These were the training and validation metrics obtained from the Neural Network: **??**

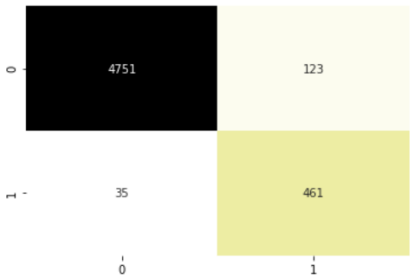

Fig. 16: Support vector machine confusion matrix

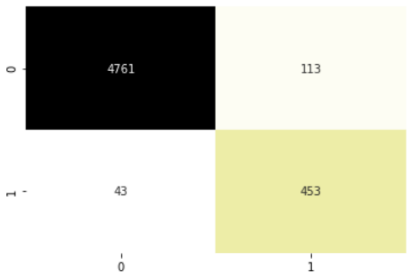

Fig. 17: confusion matrix random forest

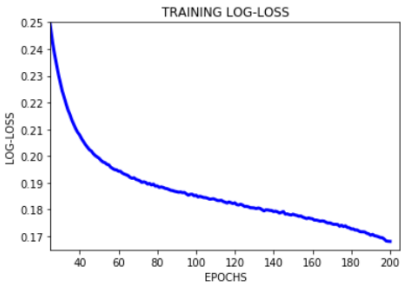

Fig. 18: Training LOG-LOSS

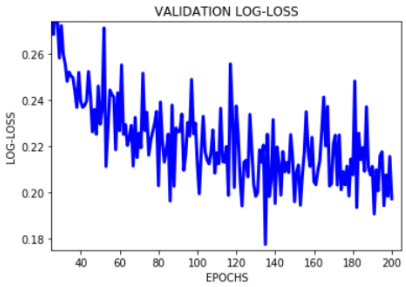

Fig. 19: Validation LOG-LOSS

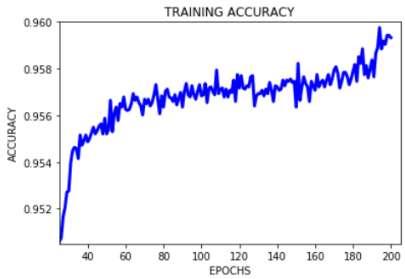

Fig. 20: Training Accuracy

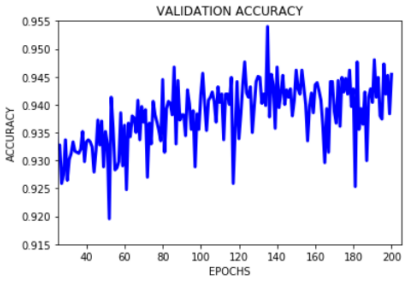

Fig. 21: Validation Accuracy

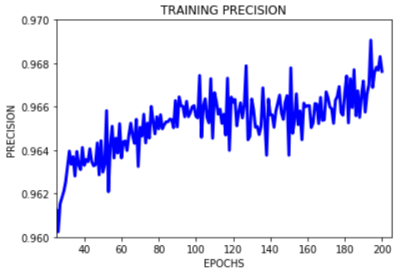

Fig. 22: Training Precision

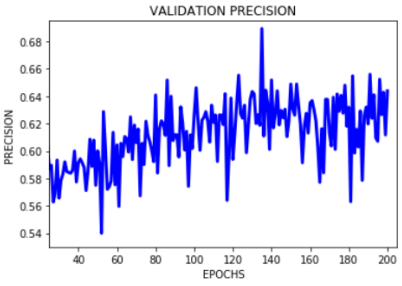

Fig. 23: Validation Precision

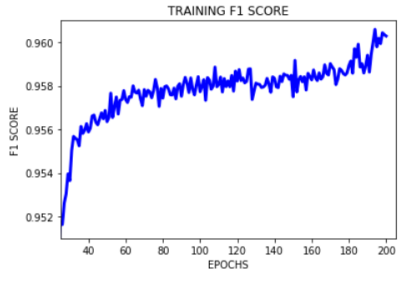

Fig. 24: Training F1 score

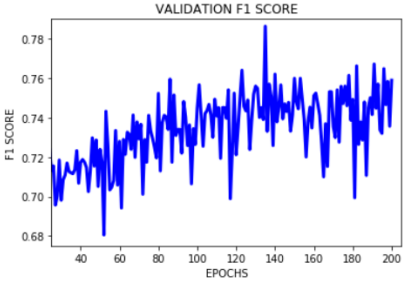

Fig. 25: Validation F1 score

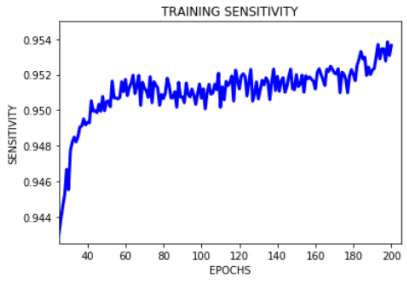

Fig. 26: Training Sensitivity

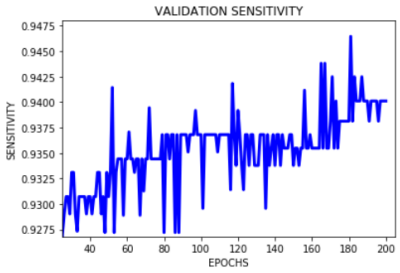

Fig. 27: Validation Sensitivity

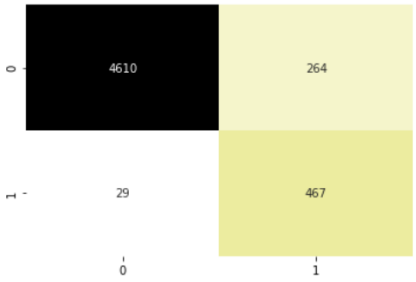

Fig. 28: Neural networks confusion matrix

4 Discussion

With the metrics presented in the results, the best algorithm in terms of AUC, for predicting pulsar stars is a Logistic Regression with L1 or Lasso Regularization. The algorithm gives the most balanced evaluation metrics amongst all the algorithms; however, it doesn't score highest in any of them nor it scores lowest in the logarithmic loss. These results are inconsistent with results presented by other works that choose the Random Forest algorithm since it gives the highest accuracy amongst the other algorithms. It's worth to say that overall, the evaluation metrics presented by the Random Forest were the highest, except for the sensitivity which it scored the lowest and f1 score which it scored in second place after Support Vector Machine; also Random Forrest presented the lowest logarithmic loss while the Logistic Regression presented the third lowest. These results might be somewhat unexpected due to the balancing process made with the dataset, as well as the hyperparameter tuning, however, with the results obtained we are convinced that this model works as a good predictor for pulsar star candidates.

5 Conclusion

Using normalized and balanced data can help solve the sensitivity, but it could also affect precision. Since the balancing method creates synthetic data that can later affect the prediction, obtaining too many false positives. Not balancing data can affect sensitivity, but the models would not predict so many false positive.

Random forests were expected to be the best model, but surprisingly, logistic regression was. Neural networks also made good predictions as expected.

It is not easy to find the best hyper parameters. Neural networks have too many. Each author uses different numbers of layers and neurons, some of them use more than one hundred, but some use less than ten, obtaining less the same results. The activation function also tended to be changed in the models of different authors.

This work provides more information and tools about the techniques used to automatically predict pulsar stars, as well as cons and pros about using the algorithms and methodology that we presented. More research and hyperparameter tuning needs to be done in order to reduce the false positive examples and increase the precision for all the models. Obtaining more examples of real pulsar stars could help overcome the problem of needing to make an oversample and losing precision.

References

[1] N. Gledenning, "Neutron stars and pulsars," in *International Workshop on Strong Magnetic Fields and Neutron Stars. http://www. lbl. gov/Science-Articles/Archive/sb/Nov-2004/03-Neutron-Stars. pdf*, 1998.

[2] M. D.R.Lorimer, *Handbook of pulsar astronomy*, ser. Cambridge Observing Handbooks for Research Astronomers. Cambridge University Press;. [Online]. Available: http://gen.lib.rus.ec/book/index.php?md5=f2857543c19c6b46d15b9c5573557866

[3] M. Vivekanand and V. Radhakrishnan, "The structure of integrated pulse profiles," *Journal of Astrophysics and Astronomy*, vol. 1, no. 2, pp. 119–128, 1980.

[4] R. J. Lyon, "Why Are Pulsars Hard To Find?" *[Thesis]. Manchester, UK Univ. Manchester; 2016.*, 2016. [Online]. Available: https://search.proquest.com/openview/edf940dd99103c1c598c2028fa99c997/1?pq-origsite=gscholar{&}cbl=51922{&}diss=y

[5] R. J. Lyon, B. W. Stappers, S. Cooper, J. M. Brooke, and J. D. Knowles, "Fifty years of pulsar candidate selection: from simple filters to a new principled real-time classification approach," *Mon. Not. R. Astron. Soc.*, vol. 459, no. 1, pp. 1104–1123, 2016. [Online]. Available: https://doi.org/10.1093/mnras/stw656

[6] R. P. Eatough, N. Molkenthin, M. Kramer, A. Noutsos, M. Keith, B. Stappers, and A. Lyne, "Selection of radio pulsar candidates using artificial neural networks," *Monthly Notices of the Royal Astronomical Society*, vol. 407, no. 4, pp. 2443–2450, 2010.

[7] V. Morello, E. Barr, M. Bailes, C. Flynn, E. Keane, and W. van Straten, "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.

[8] V. Morello, "Discovering Pulsars with Machine Learning," Tech. Rep., 2016.

[9] J. M. Ford, "Pulsar search using supervised machine learning," 2017.

[10] S. Bethapudi and S. Desai, "Separation of pulsar signals from noise using supervised machine learning algorithms," *Astronomy and computing*, vol. 23, pp. 15–26, 2018.

[11] "Prediciting A Pulsar Star — Kaggle." [Online]. Available: https://www.kaggle.com/efeergun96/prediciting-a-pulsar-star

[12] Group, Wang, Yu, Yue, and Z. Zheng, "Identifying the Existence of Pulsar Stars," Tech. Rep.

[13] Y. Fujimoto, K. Fukushima, and K. Murase, "Methodology study of machine learning for the neutron star equation of state," nov 2017. [Online]. Available: http://arxiv.org/abs/1711.06748http://dx.doi.org/10.1103/PhysRevD.98.023019

[14] W. W. Zhu, A. Berndsen, E. C. Madsen, M. Tan, I. H. Stairs, A. Brazier, P. Lazarus, R. Lynch, P. Scholz, K. Stovall, S. M. Ransom, S. Banaszak, C. M. Biwer, S. Cohen, L. P. Dartez, J. Flanigan, G. Lunsford, J. G. Martinez, A. Mata, M. Rohr, A. Walker, B. Allen, N. D. R. Bhat, S. Bogdanov, F. Camilo, S. Chatterjee, J. M. Cordes, F. Crawford, J. S. Deneva, G. Desvignes, R. D. Ferdman, P. C. C. Freire, J. W. T. Hessels, F. A. Jenet, D. L. Kaplan, V. M. Kaspi, B. Knispel, K. J. Lee, J. van Leeuwen, A. G. Lyne, M. A. McLaughlin, X. Siemens, L. G. Spitler, and A. Verkataraman, "Searching for pulsars using image pattern recognition," sep 2013. [Online]. Available: http://arxiv.org/abs/1309.0776http://dx.doi.org/10.1088/0004-637X/781/2/117

[15] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.