University of Caen

Machine learning and data analysis

GENARD Tom, DAUMAS-TSCHOPP Sacha

February 20, 2020

Abstract

To study pulsars, astronomers often has to resort to radio astronomy to differentiate them from regular stars. The major downside is that radio telescopes detects a lot of signals unrelated to pulsars, which makes the task difficult. The use of machine learning allows a way to differentiate with a decent certainty between the two, using data collected by the telescopes. This project is the result of a two week master class on computer science applied to physics problems. It will use the various techniques that have been presented in the lectures and performed during the practise sessions. This report will be focus on few points:

- The purpose of the project
- The different programs to handle it
- The result we obtained
- The discussion of the results
- The conclusion

In this second project, we will study this data using two techniques : logistic regression and neural networks.

Key words: Machine learning, Neural network, Logistic Regression, Data Analysis, Pulsar

Contents

1 Acknowledgements **4**

2 Introduction **5**
2.1 Goal of the project 5

3 Programs **6**
3.1 Analysis using neural networks 6
3.2 Analysis using a logistic regression 6
3.3 Analysis using a decision tree 6
3.4 Analysis using a random forest 7

4 Results **8**
4.1 Results of the neural network analysis 8
4.2 Results of the logistic regression 9
4.3 Results of the decision tree 9
4.4 Results of the random forest 9

5 Analysis and discussion **10**
5.1 Discussions about the algorithms 10
5.2 Displaying the results 10
5.3 Interpretation of the results 12

6 Summary **13**

1 Acknowledgements

We would like to firstly thank Christopher Jacquiot from the Deep Learning and Computer Science Master at the University of Caen for his help in understanding some concepts, and helping us for what was a very long debugging process for the neural network.

We also want to thank Paul Guibourg and Tristan Le Cornu with which we discussed ideas and concepts, and with which we could discuss and compare our methodologies.

Finally, we would like to thank Morten Hjorth-Jensen for this opportunity to learn machine learning, and also Stian Bilek, Lucas Charpentier and Hanna Svennevik for their help during the laboratory sessions.

2 Introduction

2.1 Goal of the project

In this second project, we will study data provided by the HTRU2 (High Time Resolution Universe Survey) study. It was uploaded on the Kaggle website by Pavan Raj in 2018. This data is labelled, and contains 17,898 entries manually checked by human annotators. For each pulsar candidate, we have at our disposal 8 variables, and one boolean label indicating us if the candidate is a false positive (0) or a pulsar (1). These 8 variables can be separated into two batch of 4 variables. The first one relates to the integrated profile of the detected pulse. The second part comes from the DM-SNR curve (Dispersion Measure - Signal-to-Noise Ratio). For each part, we have the mean value, the standard deviation, the excess kurtosis and the skewness. Overall, in this dataset we have 16,259 false positives, for 1,639 pulsars.

To fit this problem, we will first use a neural network to classify our data. Then, we will try a logistical regression technique. Our results will be compared to results found in a paper published by Bates et al.[1].

3 Programs

3.1 Analysis using neural networks

In this first analysis, we wanted to use neural networks. They are quite efficient, and provides decent results with relative ease. In our code, we decided to use the TensorFlow library[2] provided by Google^{TM}. Specifically, we used Keras[3] which provides a good interface with which we can communicate with TensorFlow. Our neural network will use the Multilayer Perceptron method.

For our neural network, we are going to use three hidden layers, each having 1024 neurons. This gave us a good balance between accuracy and running time. To better pinpoint the ideal pair,used a grid search on the lambda values. This would have required even more time to setup and properly test, which is the reason why we did not do it. The input vector contains the 8 parameters of each candidate, and the output during our training will be a 2 values parameters giving us the probability of a candidate of being either a false positive, or a *bona fide* pulsar. For this, we separated our data into two parts, like in the previous project. The first part is our training data and contains 80% of all candidates (14318 candidates chosen randomly, 10738 in our neural network because we use 25% of it for validation). Our test data will contain the remnant 20%, and will allow us to score our neural network.

We have monitored the results of this neural network for both training and test data using two metrics : the MSE and the accuracy. The most important metric is of course accuracy, as it gives us the ability of our neural network to predict a good result from new data, but the MSE also gave us a glimpse at the minimization process. This was important, as we used a grid search to test multiple values of the hyperparameter lambda. We tested multiple values separated evenly in the log space, and between 10^{-5} and 1.

3.2 Analysis using a logistic regression

This classification problem can be answered by a binary answer, 0 or 1. That is why we thought that we could also use a logistic regression, using our 8 parameters as input and having a boolean output signifying one of the two options. As said in the previous project, *scikit-learn* provides functions that computes this regression for us. For this project, we used *LogisticRegressionCV*, which uses cross-validation as a way to further improve our regression and our analysis.

Once again, the data set is separated into training and test data, in the same proportions. In this program though, we only track the accuracy of our prediction.

3.3 Analysis using a decision tree

Finally, we studied our data set using a decision tree. This will give us a third way classifying information. For this method, we tested multiple depths to find the best accuracy for our test data. Once again, *scikit-learn* provides an handy way of setting up a decision tree, using only our decision matrix and our labels vector.

3.4 Analysis using a random forest

With these three analysis, we could have a classification that is able to correctly identify most false positives and pulsars. But having three different methods of classification opens the path for another method, random forests. A random forest uses multiple decision trees to refine the classification process, allowing it to reach even higher accuracy. It also reduces the problems encountered due to overfitting, which is useful for large depths of tree.

We can also use results from each method and compare them between each other to determine if a candidate is a pulsar or not. This final analysis is similar to the random forest in principle, and we hope would increase the accuracy for the test data even more.

4 Results

4.1 Results of the neural network analysis

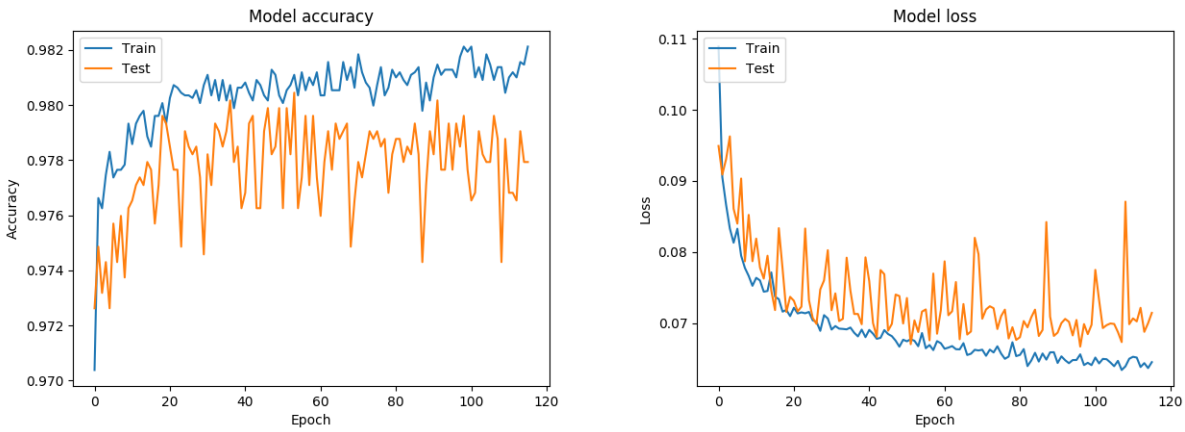
First, we used a grid search to find the ideal value of lambda for our analysis. This is done with a small sample on a few epochs, as to quickly pinpoint the best value. We end up with the following table :

Lambda	Accuracy (Training)	MAE (Training)	Accuracy (Test)	MAE (Test)
10^{-8}	0.98193332	0.06094808	0.97513966	0.02974309
10^{-7}	0.98249208	0.05870206	0.97458101	0.03793148
10^{-6}	0.98193332	0.05937176	0.97653631	0.03639188
10^{-5}	0.98081579	0.06296403	0.98156425	0.05029546
10^{-4}	0.9791395	0.06797373	0.98100559	0.02906611
10^{-3}	0.97783572	0.07668478	0.98044693	0.05411367
10^{-2}	0.97494878	0.09954117	0.97206704	0.0396847
10^{-1}	0.90510337	0.30646771	0.9122905	0.17455631
1	0.90510337	0.3090253	0.9122905	0.15341742

Table 4.1: Array gathering the results of the training and test data for different values of lambda

For our analysis, we decided to use a lambda of 10^{-4} because it was both the ideal condition for our accuracy, coupled with a lower MAE (Mean Absolute Error). It ran with 1000 epochs, with a batch size of 32. To prevent our neural network from reaching the point where it would overfit our data, we put in place an early stopping measure. This also makes the program faster, as we cut the training when it does not improve enough anymore. Our program usually stops after 120 epochs.

From this training of our neural network, we can output both the accuracy and the loss as a function of the epoch :



As we can see, we quickly reach what can be considered a saturation value after around 50 epochs. However, we can see big fluctuation in both graphs, in the order of magnitude of half a percent for example for the accuracy.

We nonetheless reach an accuracy of 98.2% with a MSE of 0.063 on our training data and 97.9% with a MSE of 0.072 on our test data.

4.2 Results of the logistic regression

The logistic regression is set up with a maximum iteration of 1000, as to prevent our program from stopping before reaching a convergence.

This methods gives us an accuracy of 97.85% on our training data and 98.30% on our test data.

4.3 Results of the decision tree

In our analysis, we used three different decision tree depth: 3, 5 and 11. Thus, we have three different accuracies and MSE for both training and test data:

Depth	Accuracy
2	0.9770917725939376
5	0.9819108814080179
8	0.9875680961028076
11	0.99301578432742

Depth	Accuracy
2	0.9801675977653631
5	0.9810055865921787
8	0.976536312849162
11	0.975977653631285

Table 4.2: Array gathering the result of the training (left) and test (right) data

We can see that, even when the accuracy on the training data reached over 99%, the ideal depth for our system seems to be 5. This was expected, as setting an high number of layers tends to overfit our training data, which in consequence reduce our ability to predict data.

4.4 Results of the random forest

For our final algorithm, we used a Random Forest. We, again, did a grid search of the best depth, which gave these results :

Depth	Accuracy
5	0.9812823019974857
10	0.9902919402151138
15	0.9963682078502584
20	1.0

Depth	Accuracy
5	0.9779329608938547
10	0.9793296089385475
15	0.9801675977653631
20	0.9804469273743017

Table 4.3: Array gathering the result of the training (left) and test (right) data

We took large depth as a way for us to take advantage of the loss of most overfitting effects in random forests. We see that for a depth of around 15, we obtain results that we consider enough for our purposes.

5 Analysis and discussion

5.1 Discussions about the algorithms

These multiple results allows us to see how each method accurately predicts if a candidate is a pulsar or not. The data is composed of around 91% positives for 9% positives, thus we expect our results to be above 91% as hat would mean better than blindly predict every star to the false positives. Every method gives a result of above 97% for our test data, which exceeds the results found in the article published by Bates et al.[1] of 85% for a blind search. This represented our aim for our classification, and we greatly exceeded it.

As expected, logistic regressions and decision trees gives similar results with ideal conditions. These methods were quite fast, which means that for a low cost of time we have results that can be sufficient in most cases. We had here 8 parameters to fit, which is ideal for this kind of analysis. Furthermore, the quick convergence proves that for cases like this one where the two can be distinguished rather easily, we can stop at this step.

Our neural network gave expectedly even better results, going as high as 98% accuracy for test data using the ideal conditions. Neural networks are of course more expansive, which means longer training times. However, they shine in large sets of data, and in cases with large numbers of parameters. One drawback we found was the time taken to do our grid search, finding the ideal value of lambda can take more time than what is worthwhile.

Finally, the random forest allows us to hopefully have even better results. For our training, we used a total number of 1000 trees. We also took the best depth we found earlier, 15. The aim of this final analysis was for us to be able to use our multiple methods to reach a final verdict for each candidate. Each of these method will receive a weight based on how well they performed.

5.2 Displaying the results

For the case of our neural network, we used shap[4] following an advice from C. Jacquiot. This tool shows graphically the contribution of each parameter during the calculation of the final value, what is called its *explicability*. This means that we can directly observe what parameters are the most important for the calculation:

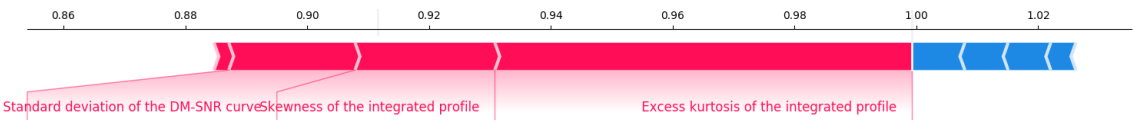


Figure 5.1: Exemple of a *shap* analysis. Here, the system has detected a pulsar.

We can also show a global representation of this in a correlation matrix. This will show not only the importance of each parameter, but also their relation between each others:

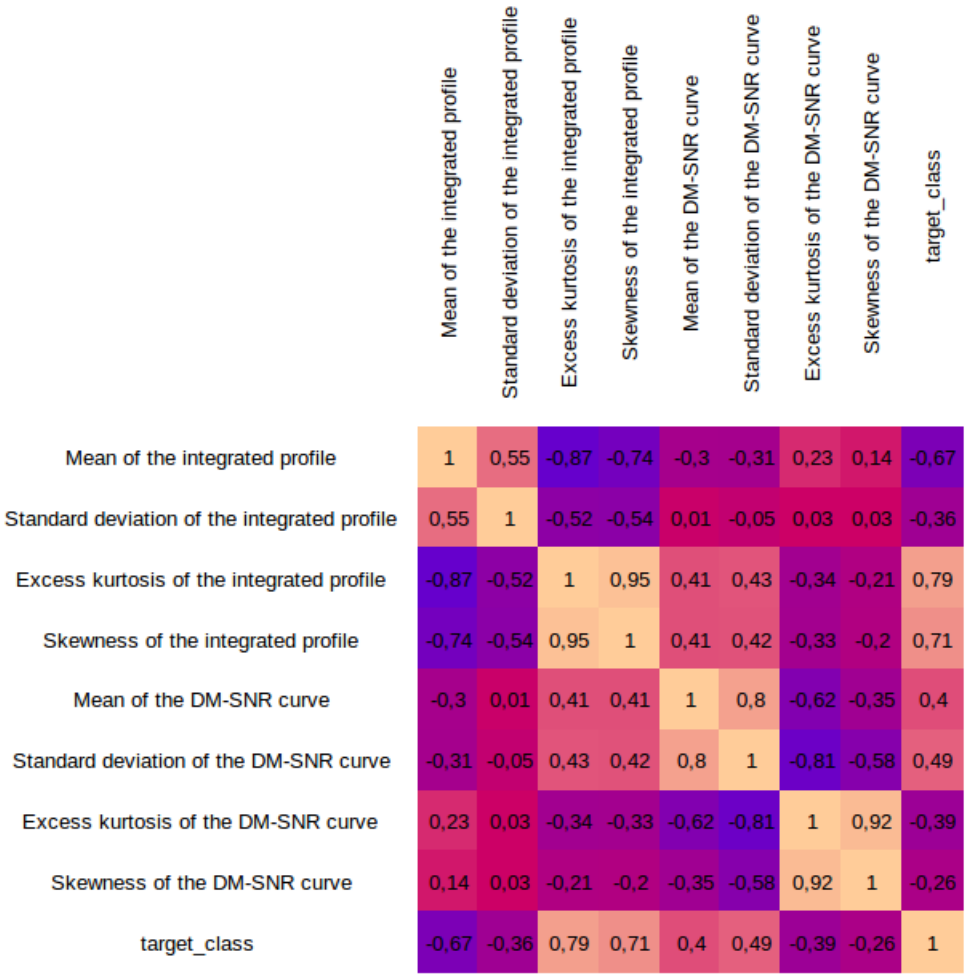


Figure 5.2: Correlation matrix between the parameters of our data (calculated using *pandas*

Finally, we have at our disposal the confusion matrix of our random forest, giving us a greater picture of the accuracy of our algorithm:

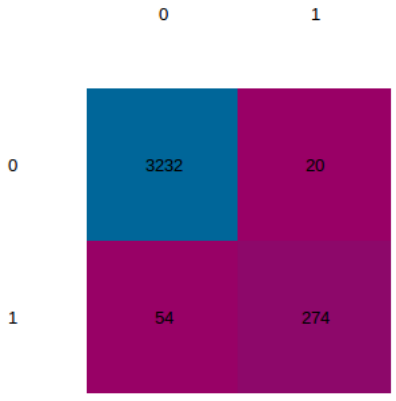


Figure 5.3: Confusion matrix outputted by our neural network

5.3 Interpretation of the results

First of all, we can see on the correlation matrix (see figure 5.2) and on the shap output that the most important parameters in determining the *targetclass* are the excess kurtosis and the skewness of the integrated curve. They both have correlation factors greater than 0.7, meaning that they have a strong impact on the determination. We also have a strong anti-correlation with the mean of this same integrated pulse. This means that we can model our data by only using these three parameters. As a first order approximation, we think that they would give decent results, cutting down greatly the time needed to model and classify our data.

The confusion matrix (see figure 5.3) allows us to monitor a very important metric, the false discovery rate. As researchers, we want to detect pulsars while putting aside false positives. This means having the less false positives we can in our analysis. In ours, we used the confusions matrix of our neural network, decision tree and our random forest.

The false discovery rate (FDR) can be formulated as such :

$$FDR = \frac{FalsePositives}{FalsePositives + TruePositives} \tag{5.1}$$

Using that formula, we can calculate the FDR for all three methods:

$$FDR_{NN} = 16.46\% \tag{5.2}$$

$$FDR_{DT} = 15.91\% \tag{5.3}$$

$$FDR_{RF} = 16.57\% \tag{5.4}$$

With FDR_{NN} the false discovery rate for our neural network, FDR_{DT} for our decision tree and FDR_{RF} for our random forest.

Overall, we can safely say that our various machine learning techniques gives results that are better for test data than what was shown in the paper by Bates et al.[1] which had a result of 85% with blind testing. However, with our FDR being over 15% this accuracy hides a very important fact when doing scientific research, the quantity of false positive is way to high for us to safely guarantee our results.

6 Summary

The analysis of the star-pulsar data set allowed us to study a vast array of machine learning techniques to study large quantity of data and parameters. We ended up with decent results for all techniques, and we managed to improve our accuracy even more using random forest methods. However, there are plenty of other techniques that we could have used, like Support Vector Machines which allows a more natural fit of our data, or a gradient descent which pinpoints more effectively the minimum of the MSE allowing a faster analysis and a better result. Finally, we could use grid search to also optimize the number of hidden layers and neurons in our neural network, which would have required more time. Finally, we hoped to be able to do a general verdict using all three analysis methods, but due to a lack of time we were not able to complete it on time.

Nonetheless, we are pleased with our overall results. We encountered some issues but we were able to fix all of them, specifically for the neural networks which took us a few try before having a working version.

References

[1] S. D. Bates, M. Bailes, B. R. Barsdell, N. D. R. Bhat, M. Burgay,S. Burke-Spolaor, D. J. Champion, P. Coster, N. D'Amico, A. Jameson, S. Johnston, M. J. Keith, M. Kramer, L. Levin, A. Lyne, S. Milia, C. Ng, C. Nietner, A. Possenti, B. Stappers, D. Thornton1, W. van Straten, "The High Time Resolution Universe Survey VI: An Artificial Neural Network and Timing of 75 Pulsars," *Cornell University*, 2012.

[2] Google LLC., "Tensorflow library,"

[3] F. Chollet, Google LLC., "Ridge regression,"

[4] S. Lundberg, Microsoft Research, "Shap,"