Finding the Best Classifier for Predicting Pulsars Using Hyperparameter Tuning

Caspar William Bruenech

December 18, 2019

Abstract

A grid search was performed in order to find the best possible classification algorithm out of five chosen machine learning models. The data set was observed to be easily classified, with all models producing AUC scores of > 0.95, with a 4-layered neural network producing the best result with a score of 0.9786. However, due to the scarcity of pulsar stars in the universe, and therefore a substantial imbalance in the data set, the result was deemed unsatisfactory due to the relatively high false negative rate, even when varying the threshold rate. With a highest true positive rate of 92%, a false positive rate of 2.5% was produced. This is a poor result when compared to the SPINN-network[1], which managed to successfully classify every pulsar in the data set while keeping a false positive rate of only 0.64%.

1 Introduction

When the most massive stars in the universe exhaust their hydrogen and helium supply, heavier elements begin to fuse inside the core of the star. As the star burns out its produced silicon elements, the resulting iron can no longer produce energy from fusion, and the radiation pressure from the core decreases, causing the gravitational forces to compress the star. The force is strong enough to merge protons and electrons, producing a core of neutrons. Following a supernova blast which ejects the outer layers into space, a neutron star is the only thing that remains. This pure mass of neutrons can have a radius of as little as 10 kilometers, but weigh up to 2.5 times as much as the Sun. As a result of the conservation of angular momentum, the rotation rate of the star increases dramatically, rotating up to hundreds of times per second. Some of these rotating neutron stars emit strong beams of electromagnetic from their magnetic poles, which can be observed when the beam is pointing towards Earth. These are called pulsars, and due to their short, regular rotational periods, these celestial objects can be used as astronomical tools for a multitude of uses, such as a universal mapping system [2], clocks that are more accurate than atomic clocks [3], and gravitational wave detectors. As a result of their usefulness within modern astronomy, being able to accurately identify pulsars is an important task. However, with such an overwhelming amount of celestial objects in the sky and electromagnetic radiation data, doing such identification by hand can become a tedious and impossible task. In this article we will use supervised learning to produce a classification model which can predict whether a radiation candidate is a pulsar star or not. Using hyperparameter tuning on five different machine learning algorithms, we aim to find a model which can successfully classify pulsars with a satisfactory accuracy.

2 Data

The data set to be analyzed in this article is named HTRU2[1] and contains a total of 17898 samples of pulsar star candidates, with 16259 negative samples (not pulsar stars), and 1639 positive samples (pulsar stars). The data was gathered during the High Time Resolution Universe Survey [4] and contains eight features; whereas the first four describe statistical properties of the candidates from the integrated pulse profile, while the final four describe the DM-SNR (signal-to-noise ratio) curve of each candidate. Additionally, the data set contains a column of binary labels, 0 and 1, indicating whether the candidate is a result of radio frequency interference and noise, or whether it is a true pulsar which has been checked by human annotators. A pairplot of the features is shown in figure 3. The data will be standardized (mean subtracted and divided by the standard deviation) in order to reduce potentially large differences in feature values, which should ensure stability in the classification models.

3 Method

3.1 Classification Model Selection

In this article we will be studying the following five classification methods:

- Logistic Regression,
- Support Vector Machine Classifier,
- Random Forest Classifier,
- K-Nearest Neighbors,
- Dense Neural Network.

The reason for choosing these specific models, is to get a variety of complexity into the analysis in order to observe whether the increased complexity of, for example, a neural network is worth the extra computational cost when compared to the K-nearest neighbor model. To find the best model within each of the classifiers we need to perform hyperparameter tuning. For all the classifiers except the neural network we will be performing an exhaustive grid search over all the specified parameter ranges with cross-validation resampling. The classifiers will be produced using the functionality of scikit-learn[2], while the neural network will be created using Keras [5] with hyperparameter tuning with Talos[3], a module created specifically for optimizing Keras neural networks. This module, when given a set of parameters for a neural network, such as number of nodes/hidden layers, learning rate, gradient descent optimizer etc,. trains the model by minimizing the given loss function (another parameter) using a specified optimizer (such as Adam, Stochastic Gradient Descent, RMSProp), then computes the validation score using the metric of choice. Each model and their resulting validation score is then presented for further analysis. Due to the limited available computing capacity combined with the large number of available hyperparameters for the various classification models, in particular the neural network, we will not be able to search all possible hyperparameters. As a result, we need to make sure

[1]`https://archive.ics.uci.edu/ml/datasets/HTRU2`
[2]`https://scikit-learn.org/stable/index.html`
[3]`https://github.com/autonomio/talos`

that we choose the most important parameters to include in our grid search or Talos scan. The following describes the parameters which were deemed most important when grid searching the different models.

- Logistic Regression
  - Strength of regularization parameter (L2),
  - Maximum number of iterations to run for solvers to converge,
  - What solver to use.
- Random Forest Classifier
  - Number of estimators (trees) in the forest,
  - Number of features to consider when looking for the best split,
  - Maximum depth of the tree.
- k-nearest neighbors
  - Number of neighbors (value of k),
  - Weight function,
  - Distance metric for the tree,
  - What algorithm to use to compute the nearest neighbors.
- Support Vector Classifier
  - Strength of L2 regularization parameters,
  - What kernel to use for transforming the data to a higher dimension for producing separable data.
- Neural Network with two hidden layers
  - Number of neurons in first (input) layer,
  - Learning rate,
  - Number of training epochs,
  - Size of the training batches,
  - What gradient descent optimizer to use,
  - Dropout rate,
  - Network weight initializer,
  - Number of neurons in the two hidden layers,
  - Activation function in the input and hidden layers,
  - Whether to weight the classes or not.

The choice for setting the number of hidden layers for the neural network to 2 came from comparing the features in the data set. As we see in figure 3, there a substantial tendency for clustering for almost all the features, but they are not all necessarily linearly separable, which indicates that we need more than zero hidden layers. Observing that most features could be separated with a 2+ degree polynomial, two hidden layers were chosen for the network.

3.2 Hyperparameter tuning

Before performing the hyperparameter tuning using the cross-validation grid search and talos, the data was preprocessed by standardizing and splitting the data into training and validation sets with a ratio of, respectively, 80% and 20%. To ensure that we are training the model as best as possible, the split data will be checked to ensure it contains the same ratio of true/false samples as the whole data set. For each classifier, a parameter grid with various values for the parameters as described in 3.1 will be created. The classifiers will then be initialized using the functionality of scikit-learn, before a grid search with 5-fold cross-validation and ROC-AUC scoring metric will be performed on the training set. Finally, the resulting data of variation of each classifier will be collected, and the classifier with the parameters that produce the highest metric score will be saved as an object file for later analysis and comparison.

For the artificial neural network, a series of experiments will be performed, in which a range of the chosen parameters will be defined. The network will then be initialized, trained, and validated for combinations of the chosen parameters. As a result of the large quantity of parameter combinations, the talos module introduces stochasticity by only choosing a certain percentage of the combinations in a random configuration. In this article we will select 10% of the possible combinations. After each experiment, an analysis will be performed on the resulting models, including a study of the metric score-parameter correlation and box plots for each parameter in order to get an idea of which parameter values score the highest and has the lowest variance. The first experiment will contain a large range of each parameter, such that we can tune in on the best values in the following experiments. The number of experiments will stop once we find the parameters which produce a satisfactory evaluation metric score.

3.3 Metrics

Due to the substantial imbalance in the data set, with 91% 0-targets, and 9% 1-targets, the choice of metric for evaluating the different classification models has to be considered. The accuracy score, which is the most common metric used for classification cases, would not be suitable in this instance, as it would give a false indication of how well the model predicts. As a result of this, we will be be utilizing the area under the Receiver operating characteristic (ROC) curve, hereby denoted as AUC, as the metric for evaluating all the classifiers. The ROC curve tells us how good the model is able to distinguish between the two classes by visualizing the false positive rate as function of true positive rate (or recall). However, in order to compute the AUROC score, a whole data set is required. Since the neural network is trained with minibatches, we can not utilize this metric while searching for the best hyperparameters for the network. There are, thankfully, other metrics we can utilize for giving us an idea of how well the network can predict on this unbalanced data set. The f1 score is one of these examples, and is a weighted mean of the precision and recall, and while it is often a good metric for unbalanced data sets, it suffers from the fact that the recall and precision both consider one class, the positive, to be the class we are interested in. This means that the f1 score is asymmetric. To deal with this, we will be implementing the Matthews Correlation Coefficient metric (MCC) [6], which treats the true and predicted classes as two binary variables and calculates the correlation coefficient between them. A higher correlation coefficient indicates then a better prediction. This metric has a range from -1 to 1, with a score of 0 being equivalent to performing random guesses. Once we have tuned the hyperparamaters of the Neural Network to produce a network with a satisfactory high MCC score, we can then compute the AUC score and the confusion matrix by predicting on the test set, and compare the results with the other classifiers.

Model	No. of parameter combs.	Time per parameter [s]	Total time
Logistic Regression	360	0.69	4m 10s
Random Forest	96	10.5	19m 50s
K-nearest neighbor	608	1.81	18m 22s
Support Vector Machines	300	30	2hr 28m

Table 1: Results of the grid search for each classifier. The second column indicates how many different models were tested.

Experiment no.	No. of parameter combs. (total)	Time per parameter [s]	Total time
1	307 (3072)	37.47	03:11:45
2	459 (4590)	42.13	05:22:18
3	18 (180)	10.22	00:03:04

Table 2: Results of the three grid search experiments performed by Talos to find the optimal Keras Neural Network parameters.

As mentioned in [1], the number of pulsars in the universe is a substantially small amount, which means that emphasis must be put on maximizing the true positive rate, or recall. However, we can not blindly look at the recall, as we can easily get a recall score of 100% by simply predicting all values to be 1. Therefore, we also need to look at another metric which takes into account false positive rates. As a result, we will not necessarily choose the network which has the best MCC score, but rather the one that has the highest combination of the recall and MCC.

4 Results

The total number of parameters and the time taken for each grid search for each standard classifier is shown in table 1, while the equivalent numbers for the talos scan experiments for the neural network are shown in table 2.

The models which achieved the highest AUC score of the standard classifiers were then chosen and trained and evaluated on the test and training set of the data, which had been put aside when grid searching. For the neural network, the parameters which produced the best highest mean MCC score and smallest variance were saved and used to initialize a new network which was then also trained and evaluated on the training and test set. The training time for each classifier was recorded, and the resulting AUC score, including the fraction AUC score/training time, is listed in table 3.

Model	AUC score	Training time [s]	Score/time
Logistic Regression	0.9750	0.323	3.026
Random Forest	0.9728	3.590	0.272
K-nearest neighbors	0.9689	0.008	115.711
Support Vector Machine	0.9740	0.254	3.848
Neural Network	0.9786	59.72	0.016

Table 3: Results from predicting on the test set for each of the best-scoring classifiers.

Confusion matrices, displaying the number of true/false positives a negatives for all the classifiers are shown in figure 1, while a plot of the ROC curve with the corresponding AUC score is shown in figure 2.

Confusion Matrix

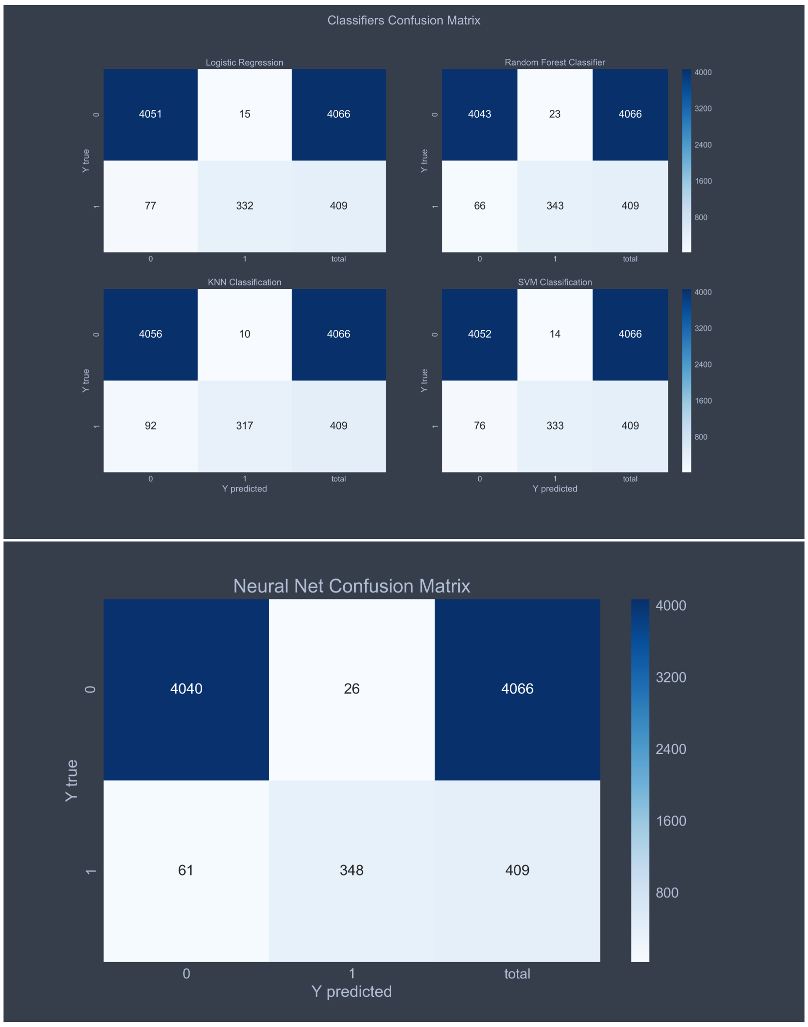

Figure 1: Confusion matrices for all classifiers, with the topmost figure showing the prediction results for the standard classifiers, while the bottom figure shows the result of the neural network. The x-axis denotes the predicted classes, while the y-axis denotes the true labeled classes.

The classifiers themselves can be found as python objects in the attached github repository.

5 Analysis

For the hyperparameter tuning, we can rank the different classifiers in the total time taken to find the best parameters. Since there was a varying number of parameters for each classifier, we

ROC Curve

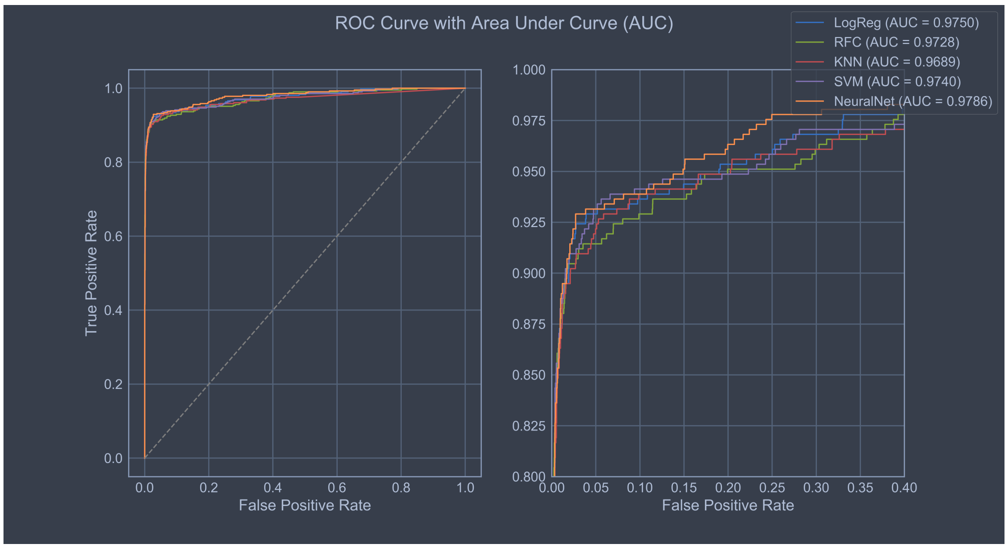

Figure 2: Receiver operating characteristic curve for each classifier showing how the models perform with varying discrimination threshold. The left plot shows the whole curve, while the right figure is zoomed in to the top left corner.

rank them according to the time taking to train plus validate each model per parameter. The ranking from fastest to slowest then goes as follows

1. Logistic regression - 0.69 s/parameter,
2. K-nearest neighbors - 1.91 s/parameter,
3. Random Forest Classifier - 10.5 s/parameter,
4. Support Vector Classifier - 30 s/parameter,
5. Artificial Neural Network $\approx$ 39.8 s/parameter.

The time/parameter for the neural network was computed by averaging the first two experiments, as the time taken per parameter for the final experiment was a significant outlier compared to the first two. Looking at these results, it is not surprising that the neural network comes out with the slowest time, as it is a significantly more complex model than the other classifiers, especially considering that the network has two hidden layers, for a total of 256 neurons whose weights and biases need to be tuned. However, we also see that the Support Vector Machines have a substantially higher time than the other classifiers, taking three times as long as the random forest classifier. This is not that surprising as Support Vector classifiers are notoriously slow for larger data sets when using a non-linear kernel. This was confirmed by looking at the specific training times for each kernel parameter, which showed a higher number of outliers when using a polynomial kernel, with substantially higher training times than the linear or Radial Basis Function (RBF) kernel. The most interesting result is, nevertheless, how the final chosen classifiers performed on the test set. In ascending order, from best to worst, the results are

1. Neural Network - $AUC = 0.9786$,
2. Logistic Regression - $AUC = 0.9750$,
3. Support Vector Machines - $AUC = 0.9740$,
4. Random Forest - $AUC = 0.9728$.
5. K-nearest neighbors - $AUC = 0.9689$

The results are not spectacular, as the difference in the AUC score between the best and worst performing classifiers is only 0.0058. This means that the best performing classifier, the neural network, will be able to successfully distinguish the positive class from the negative class 0.6% more than the worst performing classifier, K-nearest neighbors. Nevertheless, if the data set were to contain millions of samples, this small increase could result in the successful classification of hundreds of pulsars, and with a classification problem as important as identifying these, even the smallest increase in accuracy is very welcome. The neural network did have the highest score, but it also had the highest training time by a large margin, with 59.72 seconds compared to the second slowest classifier, the random forest, with 3.59 seconds. It should be noted that this is absolutely not the best possible score the different classifiers could achieve. Not only did we have to limit what hyperparameters to test for due to lack of computational power available, but also the range of the selected continuous hyperparameter values had to be limited. Ideally, a large as possible range with a small step size is ideal, but this increases the grid search time significantly. As a result, the hyperparameters which were found to produce the best results will more than likely not be the best possible values. In addition to this, the extremes of the parameter ranges were chosen with the default value used by scikit-learn as the middle point. This is not necessarily the range in which the best parameter lies, and more research should be done in order to optimize this.

As mentioned earlier, in the case of classifying pulsars, we want to reduce the false negative rate, or the number of true pulsars that are predicted as not pulsars. In the confusion matrices in figure 1, we see that the neural net does indeed have the lowest false negative rate, which is the element in the bottom left of the matrix, with 61 true candidates being classified as negative, while the random forest classifier had the second best outcome with 66. We observe that the other three classifiers, the logistic regressor, KNN classifier and SVM classifier, all had a substantially higher false negative rate, but also a very low false positive rate. In other words, the neural network and random forest were both better at predicting pulsars, but this also resulted in the classifiers predicting more of the negative samples as positive. This is the throw off when performing classification, as visualized in the ROC-curve. It is close to physically impossible to achieve a 100% true positive rate while maintaining a 0% false positive rate, which equivalates to having an AUC score of 1, and indicates that the model is able to separate the two classes 100% of the time. As a result of this, one needs to consider whether we want the model to predict more samples in general as true, which will result in more successfully classified pulsars, but more negative candidates will also be classified as true, or whether we want the model to predict fewer pulsars successfully, which also reduces the number of false samples being classified as positive. This can be tweaked by tuning the threshold at which to separate the classes. With the neural network, we were able to get the number of false negatives down to 34, which gave the number of false positives as 97, by setting the threshold to 0.1. On the other side, by setting the threshold to 0.9, the number of false positives was reduced to only 7 while the false number of false negatives rose to 124. We see therefore that these are the best and worst possible values of the false positive and negative rate for the neural network with the AUC score of 0.9786. Normally this is a rather good result, but even by tuning the threshold to produce the best recall score on the positive score, the number of false negatives is not acceptable for such an important task. Especially when compared to the *Straightforward Pulsar Identification using Neural Networks* (SPINN) network [1], which managed to produce a recall of 100% while maintaining a false positive rate of only 0.64%. One of the things that were done in the production of the SPINN classifier was to perform dimensionality reduction by studying the most important features in the data set, removing features with strong correlation such that additional information was captured by the network, increasing the performance of the classifier. This is something that was not done in this experiment, and should be included in future studies. Another source of error in this experiment is the fact that we did not tune the amount of hidden layers in the neural network. Two hidden layers was predicted to produce good results based on the visual separability of the data. However, the SPINN network was able to produce their results using only a two-layered network with no hidden layers [1], which shows that the number of hidden layers should have been included in the parameters to tune. As we see in the results table 3, all the classifiers produced high AUC scores, which means that one could potentially rule out the neural network as the best classifier for further data analysis due to its high training time. However, a neural network has significantly more flexibility in its parameter combinations, which means that it has the possibility to produce much better results (as seen with the SPINN network), while the other classifiers may not have the same potential due to their more limited number of parameters. The training time might not be substantially high with only 60 seconds, but it should be noted that this is a rather small data set, and the training time for the neural network could be a potential problem for a data set with a substantial increase in the number of samples. Nevertheless, for a classification problem as important as this, this is something that should be worth the computational cost.

6 Conclusion

In this article, a hyperparameter tuning of five different classification algorithm has been performed in order to find the classifier which best predicts the presence of a pulsar star. The results show that the data set is easy to classify, with significantly small differences in the score produced by the best and worst models. As a result, we were able to produce high AUC scores, but because of the importance of accurately labeling pulsar stars due to their scarcity in the universe, the results are not necessarily as good as they might seem at first glance. All classifiers therefore produced The neural network produced the most accurate predictions, but also featured the highest training time by a large margin. Yet, the neural network has more potential to increase its accuracy due to the increased flexibility from having more parameters to tune than the other classifiers, and we can therefore conclude that the neural network is the best classifier for this data set.

References

[1] V. Morello et al. "SPINN: A straightforward machine learning solution to the pulsar candidate selection problem". In: *Monthly Notices of the Royal Astronomical Society* (2014). ISSN: 13652966. DOI: 10.1093/mnras/stu1188. arXiv: 1406.3627.

[2] Angelo Tartaglia, Matteo Luca Ruggiero, and Emiliano Capolongo. "A null frame for spacetime positioning by means of pulsating sources". In: *Advances in Space Research* (2011). ISSN: 02731177. DOI: 10.1016/j.asr.2010.10.023.

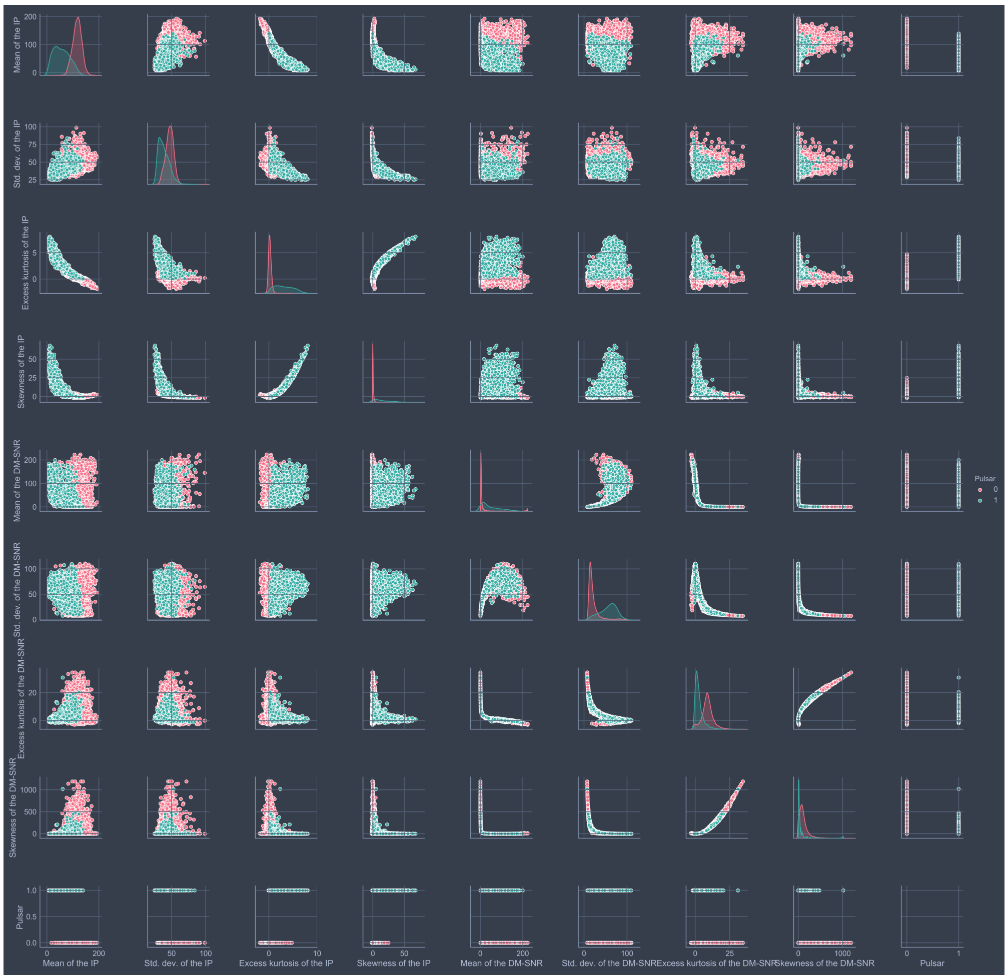

Figure 3: All combinations of the values of the features in the data set. The diagonal shows the distribution of negative and positive samples for each feature.

[3] Demetrios N. Matsakis, J. H. Taylor, and T. Marshall Eubanks. “A statistic for describing pulsar and clock stabilities”. In: *Astronomy and Astrophysics* (1997). ISSN: 00046361.

[4] M. J. Keith et al. “The High Time Resolution Universe Pulsar Survey - I. System configuration and initial discoveries”. In: *Monthly Notices of the Royal Astronomical Society* (2010). ISSN: 00358711. DOI: 10.1111/j.1365-2966.2010.17325.x. arXiv: 1006.5744.

[5] François Chollet et al. *Keras*. https://keras.io. 2015.

[6] B. W. Matthews. “Comparison of the predicted and observed secondary structure of T4 phage lysozyme”. In: *BBA - Protein Structure* (1975). ISSN: 00052795. DOI: 10.1016/0005-2795(75)90109-9.