A Comparison of Naïve Bayes and Random Forest Applied to the HTRU2 Dataset

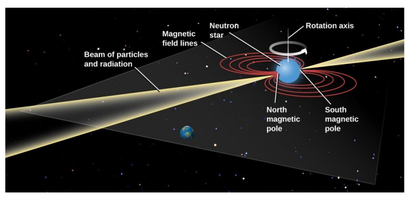
CITY UNIVERSITY OF LONDON EST 1894

Jack Bardsley and Jamie Gallagher

MSc Data Science

Description and motivation

Pulsars, a type of Neutron star of considerable scientific interest for analysis of space-time, produce periodic radio emissions detectable by radio telescopes on earth. As they rapidly rotate they emit a detectable pattern of broadband radio emissions. Telescopes search for this periodic radio signal, with each pulsar producing a unique pattern. However, pulsar radio signal is diluted by radio frequency interference and noise, making legitimate signals hard to identify. Binary classification using machine learning tools have been developed to automatically label pulsars, allowing analysis to focus on real pulsar examples. The use of machine learning has been driven by the need for real-time assessment to reduce data storage given the large volumes of data collected. We will compare two machine learning algorithms, Naïve Bayes and Random Forest, to the classification analysis results of R.J. Lyon et al to assess how optimisation of hyperparameters can improve performance metrics. [1]

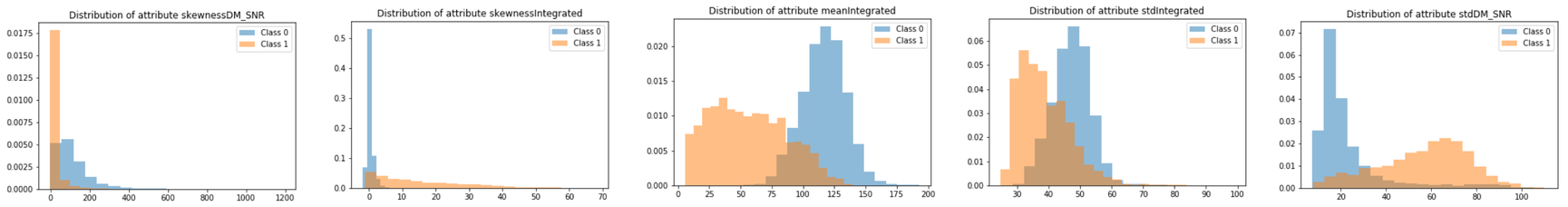


Model of a pulsar; showing the beam of radiation as the star rotates. [7]

Dataset description and initial analysis

- The data consists of a set of features derived from radio frequency signals from candidate pulsars.
- These features are a set of statistics obtained from the profile of the pulse from these processed signals[1].
- The classes to predict are “pulsar (1)” and “non-pulsar/noise” (0).
- There is an imbalance in classes with 16259 “non-pulsar/noise” and 1639 “pulsar”.

	Class = Noise (0)								Class = Pulsar (1)							
	meanInt	stdInt	kurtosisInt	skewnessInt	meanDM_SNR	stdDM_SNR	kurtosisDM_SNR	skewnessDM_SNR	meanInt	stdInt	kurtosisInt	skewnessInt	meanDM_SNR	stdDM_SNR	kurtosisDM_SNR	skewnessDM_SNR
mean	116.56	47.34	0.21	0.38	8.86	23.29	8.86	113.62	56.69	38.71	3.13	15.55	49.83	56.47	2.76	17.93
std	17.48	6.18	0.33	1.03	24.41	16.65	4.24	106.72	30.01	8.03	1.87	14.00	45.29	19.73	3.11	50.90
min	17.21	28.70	-1.88	-1.79	0.21	7.37	-3.14	-1.98	5.81	24.77	-0.09	-1.14	0.49	7.66	-1.86	-1.87
max	192.62	98.78	4.79	24.87	223.39	110.64	34.54	1191.00	139.26	83.80	8.07	68.10	199.58	109.66	30.88	1017.38



Naïve Bayes

Description;

- Probabilistic classifier based on Bayes’ theorem.
- For each attribute, a class conditional probability is calculated.
- This classifier assumes that features are independent and so the probability of new data belonging to a particular class can be calculated from the product of probabilities from each distribution of features for a particular class (likelihood) multiplied by the prior class probability.
- MAP (maximum a posterioiri) rule assigns the output of the classifier.

Pros;

- Simplicity
- Fast

Cons;

- Assumes independence between features (however this has still been shown to work well in many cases[3]).

Hypothesis statement

- We expect Random Forest to give better results than Naïve Bayes, but performance is known to vary depending on the dataset. [2]
- R.J. Lyon et al using a Decision Tree C4.5 classifier achieved G-mean of 0.926 and recall of 0.904, and using Naïve Bayes achieved G-mean of 0.902 and recall of 0.863. We expected to achieve similar results for our Naïve Bayes model and to achieve higher performance metrics for our Random Forest model. [1]

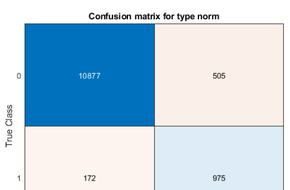
Choice of parameters and experimental results

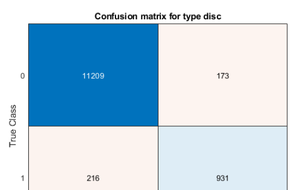
Naïve Bayes

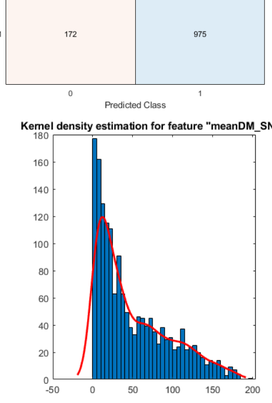
- Literature supports that performance improvements may be possible by replacing Gaussians with the use of kernel density estimation[4] or the discretisation of continuous values[5].
- Replacing Gaussians with a kernel density estimation would be desirable when the data is not normally distributed (as shown in the figure below).
- These methods have been implemented with the cross-validation results for each shown below.

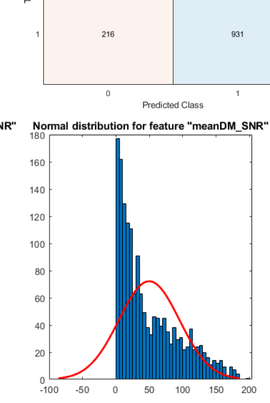
Model	Accuracy	G-mean	Recall
Gaussian	0.94543	0.90129	0.85004
KDE	0.96025	0.91291	0.85876
Discretisation	0.96895	0.89406	0.81168

- Following these results, the model implementing the kernel density estimation will be chosen as it provides higher recall and g-mean scores. This method however is the most computationally complex. Confusion matrices for the other models are show below.





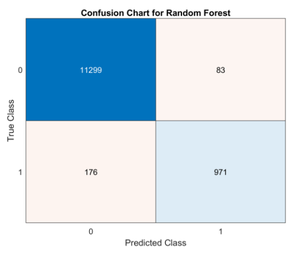


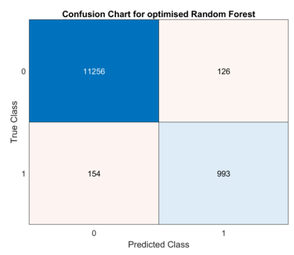


Random Forest

- Literature offers guidance for optimal range of hyperparameters for random forest;
  - minLeafSize; setting it lower leads to trees with larger depth but increases computation time[6]
  - numFeatures; mathematically the square root of total number of features is a reasonable value and convenient regarding error rate, but sometimes it can be improved. [6]
  - numTrees; Breiman [2] advises that you can’t overfit with number of trees, but larger numbers increase computation time. Studies have shown the more trees trained increases variable importance stability. [6]
- Bayesian Optimisation was run 10 times and hyperparameters with lowest classification error were chosen; minLeafSize of 3, number features 6 and number of trees 22.
- The model with optimised hyperparameters was implemented with cross validation, results below.

Model	Accuracy	G-mean	Recall
Random Forest	0.9793	0.9130	0.8466
Random Forest with Bayesian Optimisation	0.9777	0.9253	0.8657





- The optimised random forest achieved improvements in G-mean and recall when compared to the baseline random forest. The number of pulsars rejected by the model was reduced by 12.5%.
- While this improvement was desired, the number of false positives also increased with the optimised model.

Random Forest

Description;

- Introduced in 2001 by Leo Breiman following his research on bagging, Random Forest is an ensemble of uncorrelated decision trees. [2]
- Random samples are drawn with replacement (bootstrapping) to create a training set.
- From M input features of the dataset a random subset of m features are selected for splitting each node, where m < M (m is constant during forest growth).
- No pruning, trees are grown to their largest extent.
- Compromise on m size is required for optimisation. Reducing m reduces tree correlation which reduces forest error rate, but also reduces strength which increases forest error rate.
- Random Forest is an eager learner.

Pros;

- Can assess the prediction accuracy by assessing how many m subsets
- Creates high variance in each tree, but a lower variance across the entire forest once predictions are averaged (bagging, or bootstrap aggregating) without increasing bias.
- Performs well on large datasets and doesn’t require pre-processing.

Cons;

- With n testing sets, it takes n-times as long to test.
- Other algorithms can perform better for specific datasets.

Training and evaluation methodology

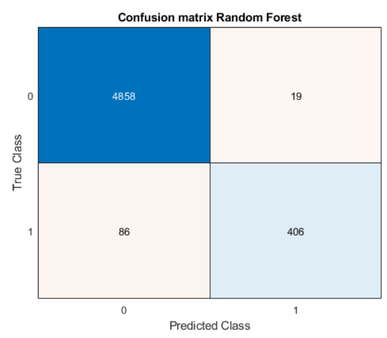
- The dataset was split randomly into 70% training data and 30% test data.
- 5-fold cross-validation was used to evaluate the skill of each model on the training data, prior to final assessment of the test data.
- Hyperparameter optimisation was analysed for both models.
  - Naïve Bayes; compared Gaussian distribution with kernel density estimation and discretisation of continuous values for Naïve Bayes.
  - Random Forest; Bayesian Optimisation was chosen over a grid search due to computational efficiency. Three hyperparameters were optimised; minimum number of observations per tree leaf (minLeafSize), number of features to sample at each leaf node (numFeatures) and number of trees in the forest (numTrees).
- Classification error was calculated for hyperparameter optimisation of both models and performance metrics extracted for analysis.
- G-mean and recall have been chosen as the accuracy metrics due to the imbalance in class types in the dataset (predicting all of the test data as class 0/noise would give a classification rate of around 90%). G-mean describes the ratio between positive and negative accuracy[1].

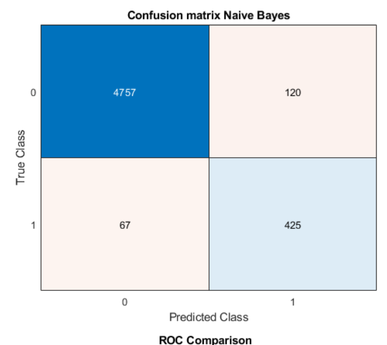
$$Gmean = \sqrt{\frac{TP}{TP+FN} \times \frac{TN}{TN+FP}} \qquad Recall = \frac{TP}{TP+FN}$$

Analysis and evaluation of results

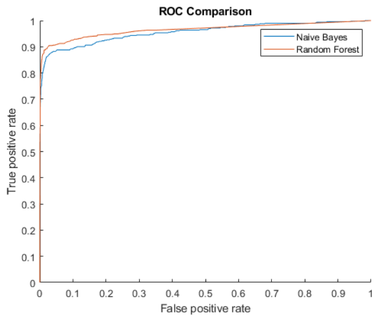
For the best models:

Model	Accuracy	G-mean	Recall
Naïve Bayes	0.96517	0.91791	0.86382
Random Forest	0.98044	0.90664	0.8252





- Whilst both models produced very high classification accuracy scores, the recall of the naïve Bayes model is slightly higher. However, the naïve Bayes also gives a much higher rate of false positives. This would incur a cost in telescope time and resources that may outweigh the benefits of fewer missed pulsar candidates.
- The ROC curve shows that the rate of true positives to false positives is slightly higher for the random forest as the threshold increases.
- The random forest model comes with the expense in complexity and computational time in comparison to the naïve Bayes for both training and optimisation (even for the most computationally expensive variant of the naïve Bayes model implementing kernel density estimation). As stated in the original paper[1], the need for a machine learning solution arises from the imminent increase in the volume of data that would need to be processed in real time as storage becomes too costly.
- Bayesian optimisation of the random forest hyperparameters proved to have little effect on the performance of the model in this case, however this model was chosen as it provided a slight improvement in recall (+0.191).



Lessons for the future

Further optimisations should be made with regards to the bin sizes used in the discretisation process across different features for the naïve Bayes classifier. Before implementing either models in a real time processing application, the time taken to calculate the predictions should be considered when selecting the appropriate method. A second larger dataset (HTRU 1) of pulsar data could be used to test our models.

References

1. 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, Monthly Notices of the Royal Astronomical Society 459 (1), 1104-1123
2. L. Breiman, Random Forests. Machine Learning 45, pp. 5–32, 2001.
3. P. Domingos, M. Pazzani. On the Optimality of the Simple Bayesian Classifier under Zero-One Loss, Machine Learning November 1997, Volume 29, Issue 2–3, pp 103–130
4. G. H. John, P Langley. Estimating Continuous Distributions in Bayesian Classifiers. 2013. arXiv:1302.4964
5. J. Dougherty, R. Kohavi, and M. Sahami. Supervised and Un-supervised Discretization of Continuous Features, Machine Learning Proceedings 1995. pp. 194–202
6. P. Probst, A. Boulesteix, M.N. Wright. Hyperparameters and Tuning Strategies for Random Forest, Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, January 2019, Volume 9, Issue 3
7. T. Hisgett, Model of a Pulsar, Lumen Learning, https://courses.lumenlearning.com/astronomy/chapter/pulsars-and-the-discovery-of-neutron-stars/