Predicting Pulsar Stars

ML Final project – Classification and cross validation

Course code: CHEMENG:787

Submitted by

Sriaravindh Rajagopal (Mac ID: 400271285)

Yash Dham (Mac ID: 400279698)

Contents

List of images:

List of tables:

Introduction to the dataset:
A pulsar (as in quasar and pulse) is a strongly magnetised rotating neutron star or white dwarf emitting an electromagnetic radiation beam. Neutron stars are very dense and have short rotational cycles that are normal. As pulsars rotate, their emission beam sweeps across the sky, producing a detectable pattern of broadband radio emission when this crosses our line of sight. This pattern repeats periodically as pulsars rotate rapidly. Thus, pulsar discovery includes the hunt with large radio telescopes for periodic radio signals. Each pulsar produces a barely exclusive emission pattern, which varies barely with every rotation. Thus, a potential sign detection known as a 'candidate', is averaged over many rotations of the pulsar, as determined by means of the length of an observation. In the absence of additional info, every candidate could doubtlessly describe a real pulsar. However, in practice almost all detections are resulting from radio frequency interference (RFI) and noise, making valid signals difficult to find.

Figure 1: Neutron star / Pulsar star

Machine learning tools are actually getting used to automatically label pulsar candidates to facilitate rapid analysis. Classification systems especially are being broadly adopted, which treat the candidate facts units as binary class problems. Here the valid pulsar examples are a minority high-quality class, and spurious examples the majority negative class.

The statistics set shared here carries 16,259 spurious examples caused by RFI/noise, and 1,639 real pulsar examples. These examples have all been checked through human annotators. Each row lists the variables first, and the elegance label is the final entry. The class labels used are 0 (negative) and 1 (positive).

Attribute information:
Each candidate is defined through 8 continuous variables, and a single class variable. The first four are simple information obtained from the incorporated pulse profile (folded profile). This is an array of continuous variables that describe a longitude-resolved version of the sign that has been averaged in each time and frequency. The remaining 4 variables are similarly received from the DM-SNR curve.

These are summarised below:

1. Mean of the integrated profile.
2. Standard deviation of the integrated profile.
3. Excess kurtosis of the integrated profile.
4. Skewness of the integrated profile.
5. Mean of the DM-SNR curve.
6. Standard deviation of the DM-SNR curve.
7. Excess kurtosis of the DM-SNR curve.
8. Skewness of the DM-SNR curve.
9. Class

1,639 positive samples and 16,259 negative samples.

Data Exploration and Pre-processing:

We are splitting the data as dependent (y) and independent (x) variables. As an initial step we are finding the correlation of the variables.

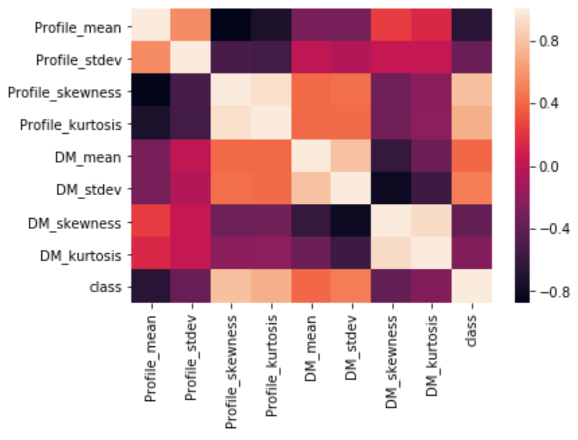

Figure 2: Correlation heatmap of variables

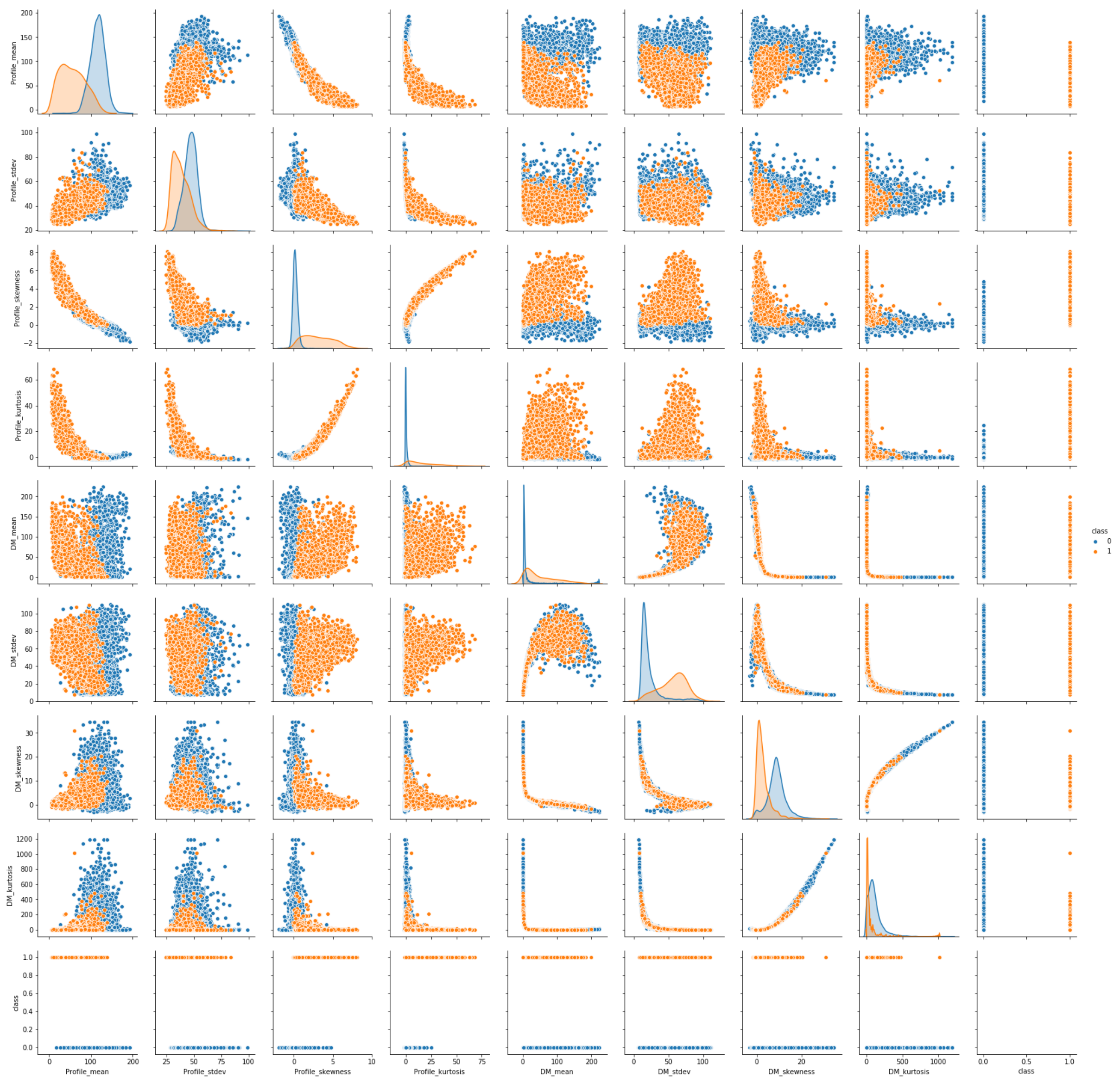

Figure 3: Pair plot analysis

Initial Model building:

Initially we are building the base models of Logistic regression, Naïve Bayes', Decision tree and random forest and we find the accuracy scores and ROC_AUC which are showed in table 1. For each base model the hyper parameters are tuned using the GridSearchCV method from sklearn and optimal parameters are found to get the maximum possible accuracy.

Table 1: Base model results

Sl. No.	Algorithm	Train time	Test time	True Positive	False Positive	False Negative	True Negative	Accuracy Initial (%)
1	Logistic Regression	0.078076	0.001717	16173	86	289	1350	97.904794
2	KNN	0.016884	0.172204	16259	0	0	1639	100
3	Naïve Bayes	0.007273	0.004705	15528	731	249	1390	94.524528
4	Decision Tree	0.126298	0.001854	16192	67	254	1385	98.206504
5	Random Forest	0.412672	0.031005	16192	67	254	1385	99.793273

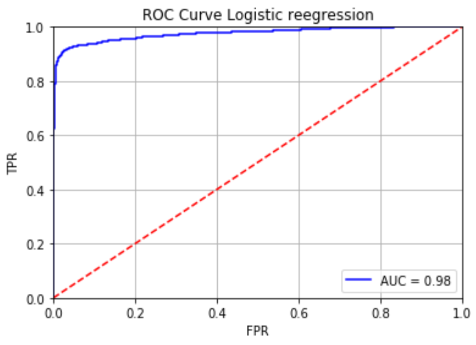

Figure 4: ROC curve for Logistics regression

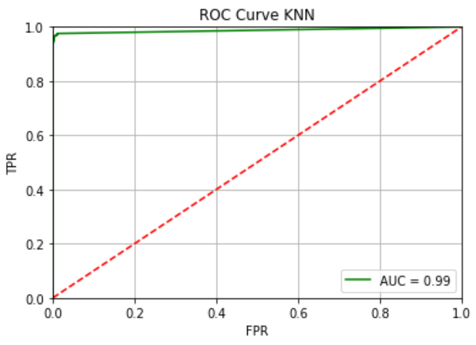

Figure 5: ROC curve for KNN

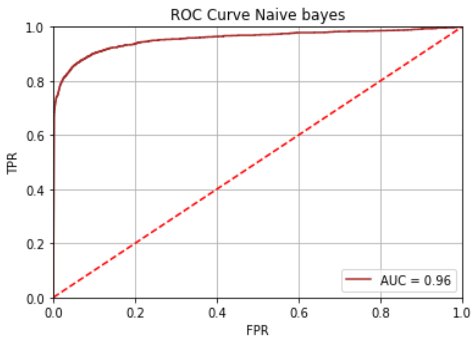

Figure 6: ROC curve for Naive bayes'

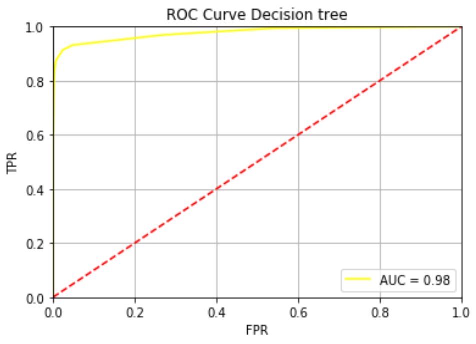

Figure 7: ROC curve for Decision tree

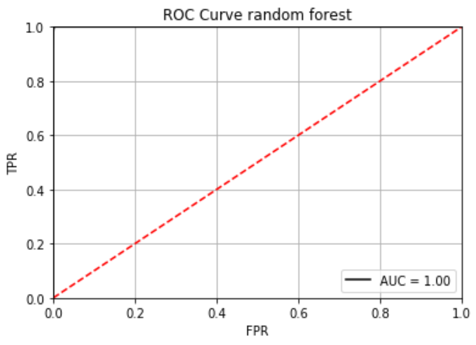

Figure 8: ROC curve for Random forest

Bagged models building:
A Bagging classifier is an ensemble estimator that fits base classifiers each on random subsets of the original dataset and then aggregate their individual predictions (either by voting or by averaging) to form a final prediction. This method of ensembling is used to reduce the high variance error.

Our aim is optimising the bias and variance error in the model by performing cross validation and obtaining the consistency of performance to avoid overfitting in the training data and ensure better performance through generalisation in the test data.

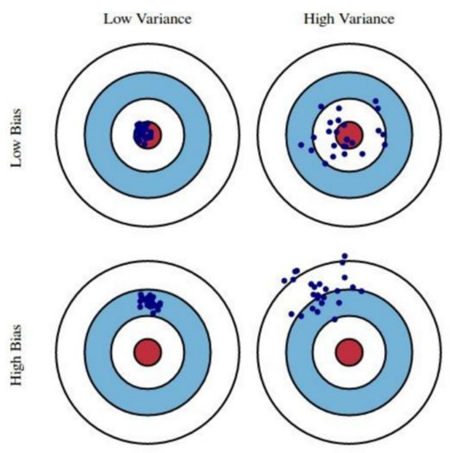

Figure 9: Bias and variance trade off

The results of the bagged models are given in table 2

Table 2: Bagged models results

Sl. No.	Algorithm	Train time	Test time	True Positive	False Positive	False Negative	True Negative	Accuracy Initial (%)
1	Bagged NB	0.022475	0.020942	15522	737	251	1388	94.47983
2	Bagged LR	1.112256	0.012086	16171	88	288	1351	97.899207
3	Bagged DT	0.082798	0.003647	16162	97	237	1402	98.13387

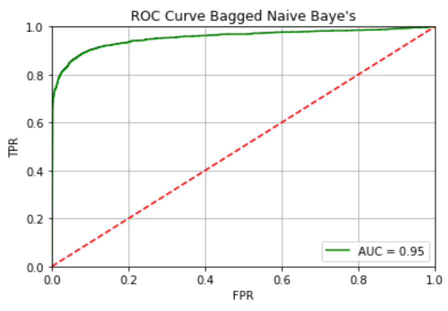

Figure 10: ROC curve for Bagged Naive Bayes'

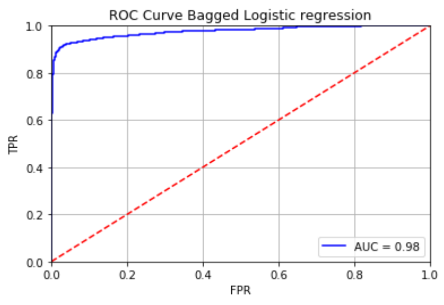

Figure 11: ROC curve for Nagged Logistic regression

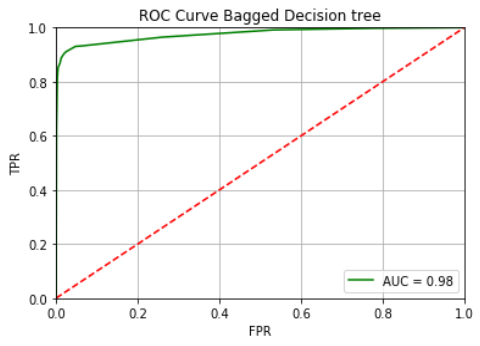

Figure 12: ROC curve for Decision tree

Here we are not bagging KNN and Random forest because they already give a high accuracy score.

Cross validation phase 1:
To check the generalisation capabilities or the performance of models with the test set we are performing cross validation experiment with 5 folds. The metrics use is recall and AUC of ROC. The recall is the ratio tp / (tp + fn) where tp is the number of true positives and fn the number of false negatives. The recall is intuitively the ability of the classifier to find all the positive samples. The best value is 1 and the worst value is 0. Here we are adopting recall since finding the class 1 as class 1 is important than finding class 0 as class 0.

Table 3: Recall based cross validation summary

Sl. No.	Model	Recall Bias	Recall Variance
1	Base Log Reg	0.797061	0.003935
2	Bagged Log reg	0.791576	0.004723
3	Base KNN	0.766149	0.002808
4	Base NB	0.848326	0.001038
5	Bagged NB	0.847003	0.001136
6	Base DT	0.824795	0.003245
7	Bagged DT	0.813274	0.006373
8	RF	0.800293	0.008526

Table 4: AUC based cross validation summary

Sl. No.	Model	AUC Bias	AUC Variance
1	Base Log Reg	0.91	0.00005
2	Bagged Log reg	0.91	0.00006
3	Base KNN	0.89	0.00007
4	Base NB	0.9	0.00006
5	Bagged NB	0.9	0.00007
6	Base DT	0.91	0.00008
7	Bagged DT	0.91	0.00008
8	RF	0.92	0.00009

From the above we can conclude that NB, bagged NB and DT, bagged DT (top 4 with optimal e-z trade off) are good. So, we are trying boosting techniques with those models. We are not going for a deep code and validation just building the models and doing cross validation.

Boosted models:
The term ‘Boosting’ refers to a family of algorithms which converts weak learner to strong learners. Boosting is an ensemble method for improving the model predictions of any given learning algorithm. The idea of boosting is to train weak learners sequentially, each trying to correct its predecessor. There are two types of boosting techniques namely

1. Adaptive boosting (AdaBoost)
2. Gradient Boosting (GBoost)

We are boosting the Naïve Bayes’, Decision tree (AdaBoost) base models (not going for boosted bagged models) and also doing a stand-alone gradient boost model. These models along with the previously determined models will be subjected to cross validation.

Cross validation phase 2:
The following table shows the results of cross validation

Table 5: Recall based cross validation results

Sl. No.	Model	Recall Bias	Recall Variance
1	Base NB	0.848326	0.001038
2	Bagged NB	0.847003	0.001136
3	Base DT	0.824795	0.003245
4	Bagged DT	0.813274	0.006373
5	Boost NB	0.451023	0.057511
6	Boost DT	0.802548	0.003584
7	Gradient Boost	0.830182	0.003187

Table 6: AUC based cross validation results

Sl. No.	Model	AUC Bias	AUC Variance
1	Base NB	0.9	0.00006
2	Bagged NB	0.9	0.00007
3	Base DT	0.91	0.00008
4	Bagged DT	0.91	0.00008
5	Boost NB	0.59	0.00914
6	Boost DT	0.91	0.00010
7	Gradient Boost	0.92	0.00007

Final Model Building:
Finally, we find Naïve Bayes’ as a better model for this dataset and building the final model, with better bias-variance trade off and generalisation effects. So we are building a model using NB with train test split of 75/25 ratio

Final Model (NB) accuracy: 95%

Train time: 0.0037

Test time: 0.0009

AUC: 0.95

Project take away:

1. High accuracy in the base models doesn’t mean that it is the best model
2. There is a possibility of error due to lack of generalisation/overfitting in the test set.
3. Cross validation helps in finding the overfitting issue in models.

References:

1. NASA: https://fermi.gsfc.nasa.gov/ssc/data/analysis/scitools/pulsar_analysis_tutorial.html
2. MEDIUM: https://medium.com/@nunorc/predicting-pulsar-stars-using-neural-networks-ecb3a527336b
3. KAGGLE: https://www.kaggle.com/pavanraj159/predicting-a-pulsar-star