Pulsar Project

CLASSIFYING CANDIDATES INTO PULSAR AND NON-PULSAR CLASSES TO AID DISCOVERY

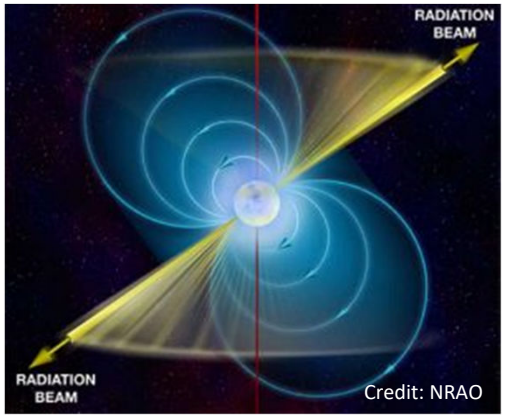
By Sheldon Kasper

Jupyter Notebook Available at:

https://github.com/analyticalbadger/Pulsar-Project

PULSAR PROJECT

Background



Pulsars are rotating neutron stars observed to have pulses of radiation at very regular intervals that typically range from milliseconds to seconds and are of considerable scientific interest. As pulsars rotate, their emission beam sweeps across the sky. When this beam crosses a line of sight, it produces a detectable pattern of broadband radio emission. As pulsars rotate rapidly, this pattern repeats periodically.

PULSAR PROJECT

Background

The Parkes telescope in Australia.
Picture taken by Roy Smits (JBCA)

Each pulsar produces a slightly different emission pattern, which varies slightly with each rotation. A potential signal detection known as a 'candidate', is averaged over many rotations of the pulsar, as determined by the length of an observation. In the absence of additional information, each candidate could potentially describe a real pulsar. However, in practice almost all detections are caused by radio frequency interference (RFI) and noise, making legitimate signals hard to find.

This project involves classifying candidates into pulsar and non-pulsar classes from the High Time Resolution Universe Survey (South).

HTRU2 Data Set from UCI Machine Learning Repository
https://archive.ics.uci.edu/ml/datasets/HTRU2

Source: Dr Robert Lyon, University of Manchester, School of Physics and Astronomy, Alan Turing Building, Manchester M13 9PL, United Kingdom

PULSAR PROJECT

Objective

To explore the HTRU2 dataset and determine ways to classify candidates into pulsar and non-pulsar classes. Various classification models will be built with the goal being to find the best model that classifies the candidates.

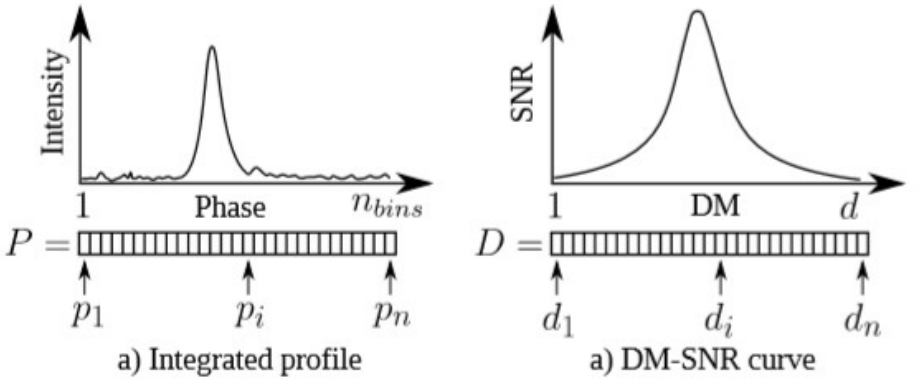
Exploratory Data Analysis

The dataset contains 17,898 data points.

There are eight continuous features and one categorical target.

Features

Mean of the Integrated Profile
Standard Deviation of the Integrated Profile
Excess Kurtosis of the Integrated Profile
Skewness of the Integrated Profile
Mean of the DM-SNR Curve
Standard Deviation of the DM-SNR Curve
Excess Kurtosis of the DM-SNR Curve
Skewness of the DM-SNR Curve



Source: Dr Robert Lyon, University of Manchester

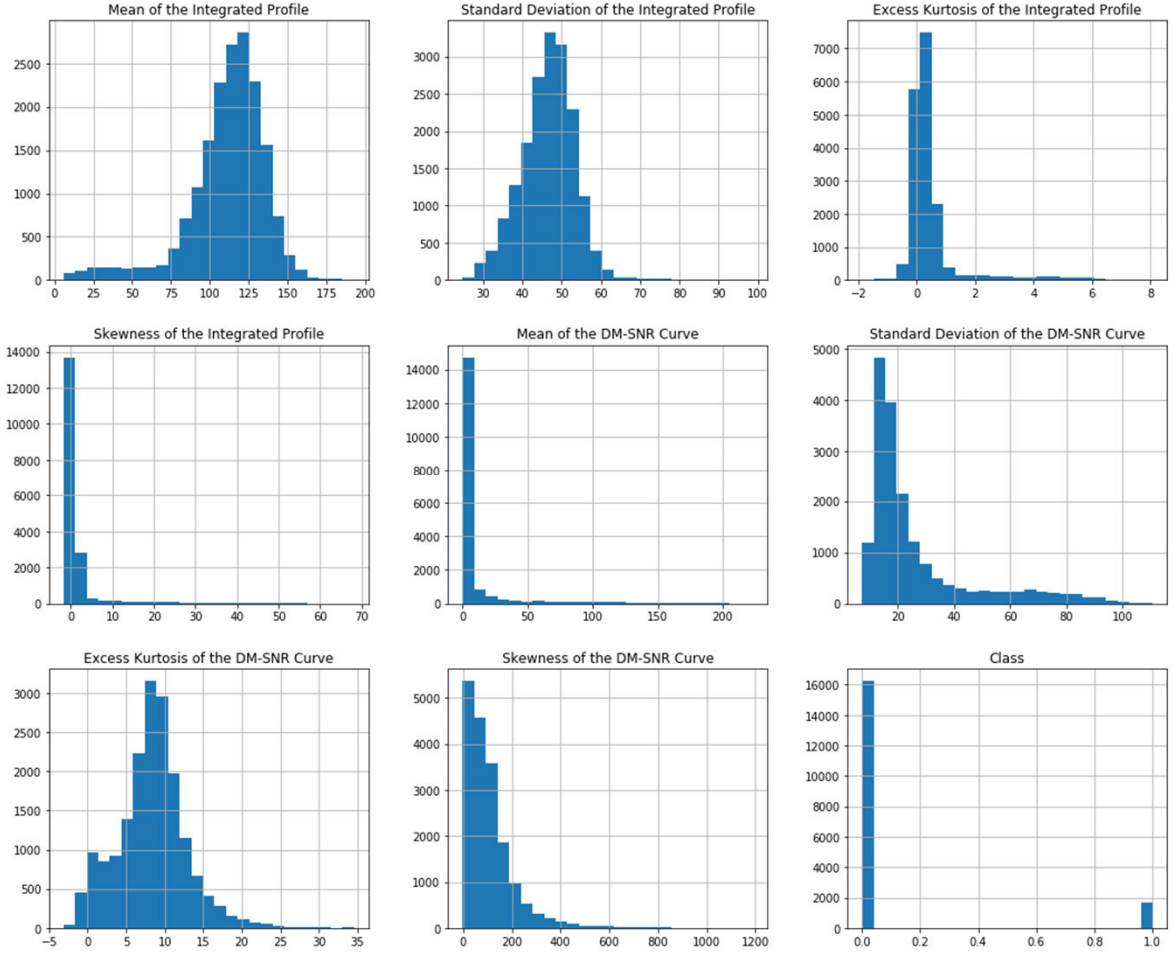
Target

Class

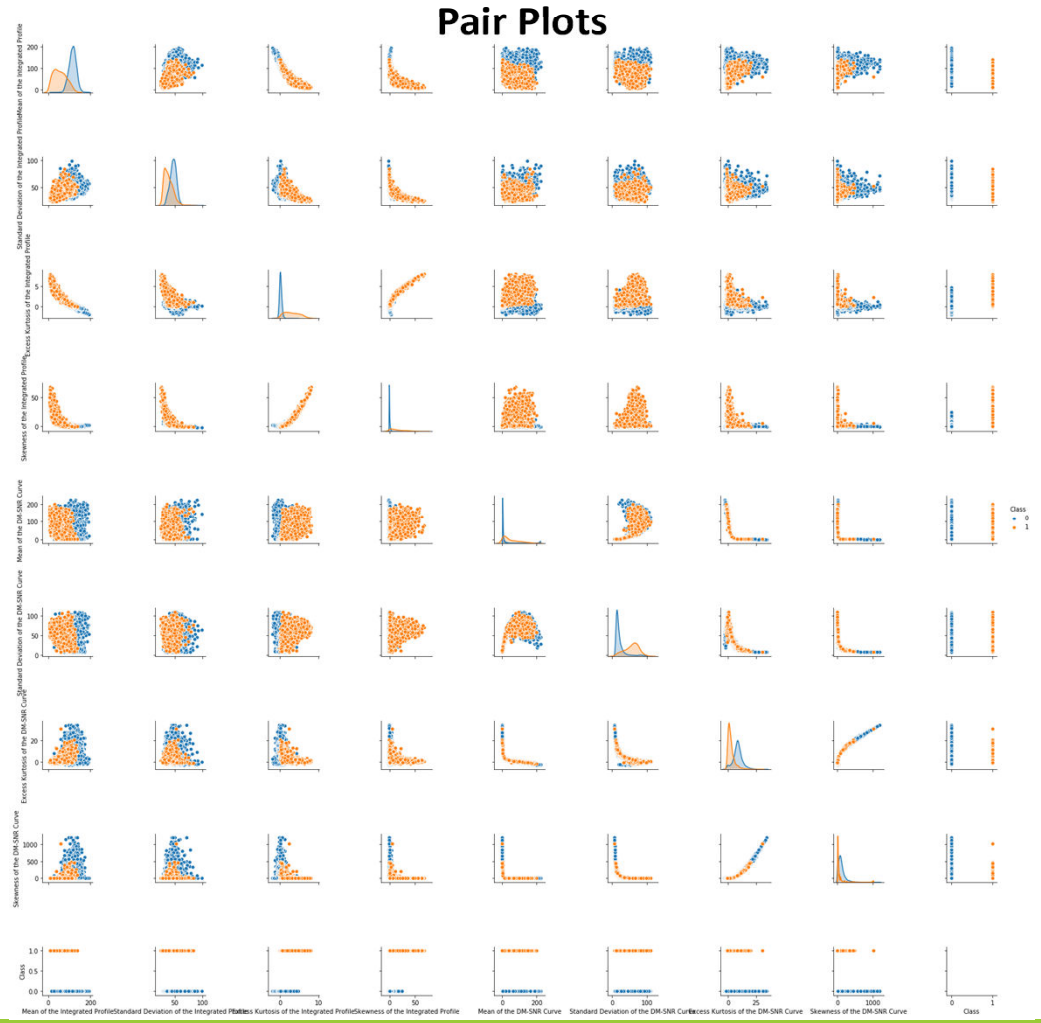
Overall, the data seems fairly well behaved. There are no null entries or duplicate rows. There are some outliers but none seem too extreme.

PULSAR PROJECT

Histograms of the Various Features and Target

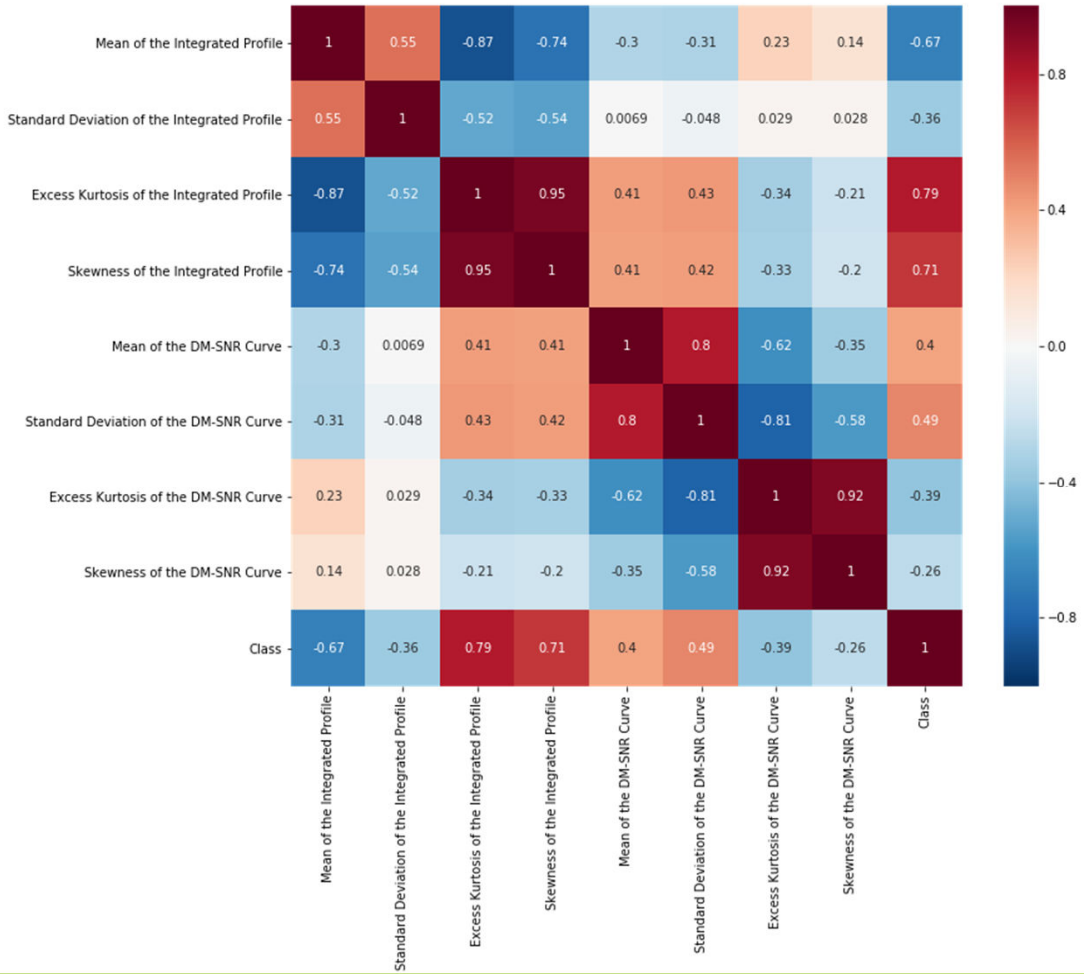


PULSAR PROJECT



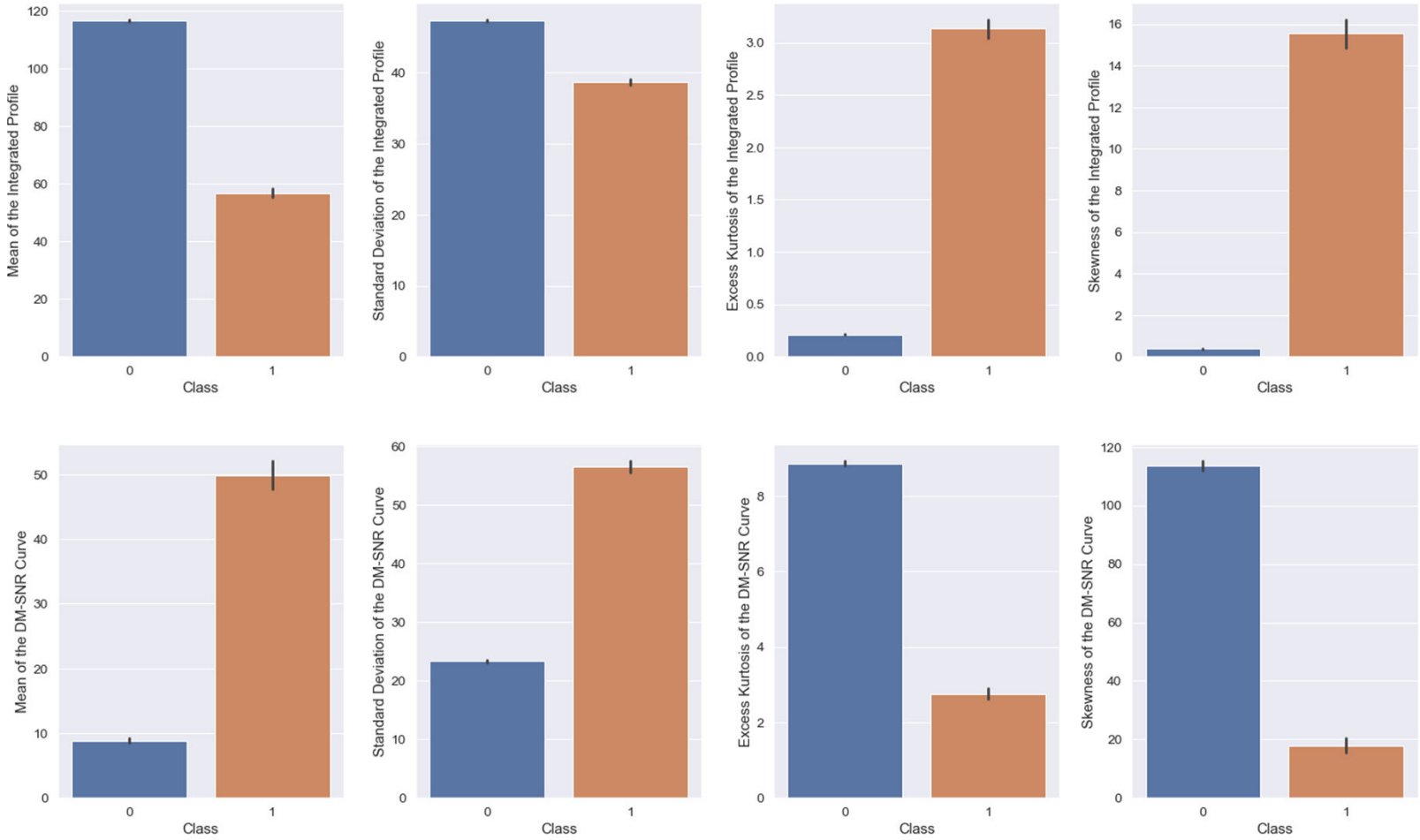
PULSAR PROJECT

Correlations



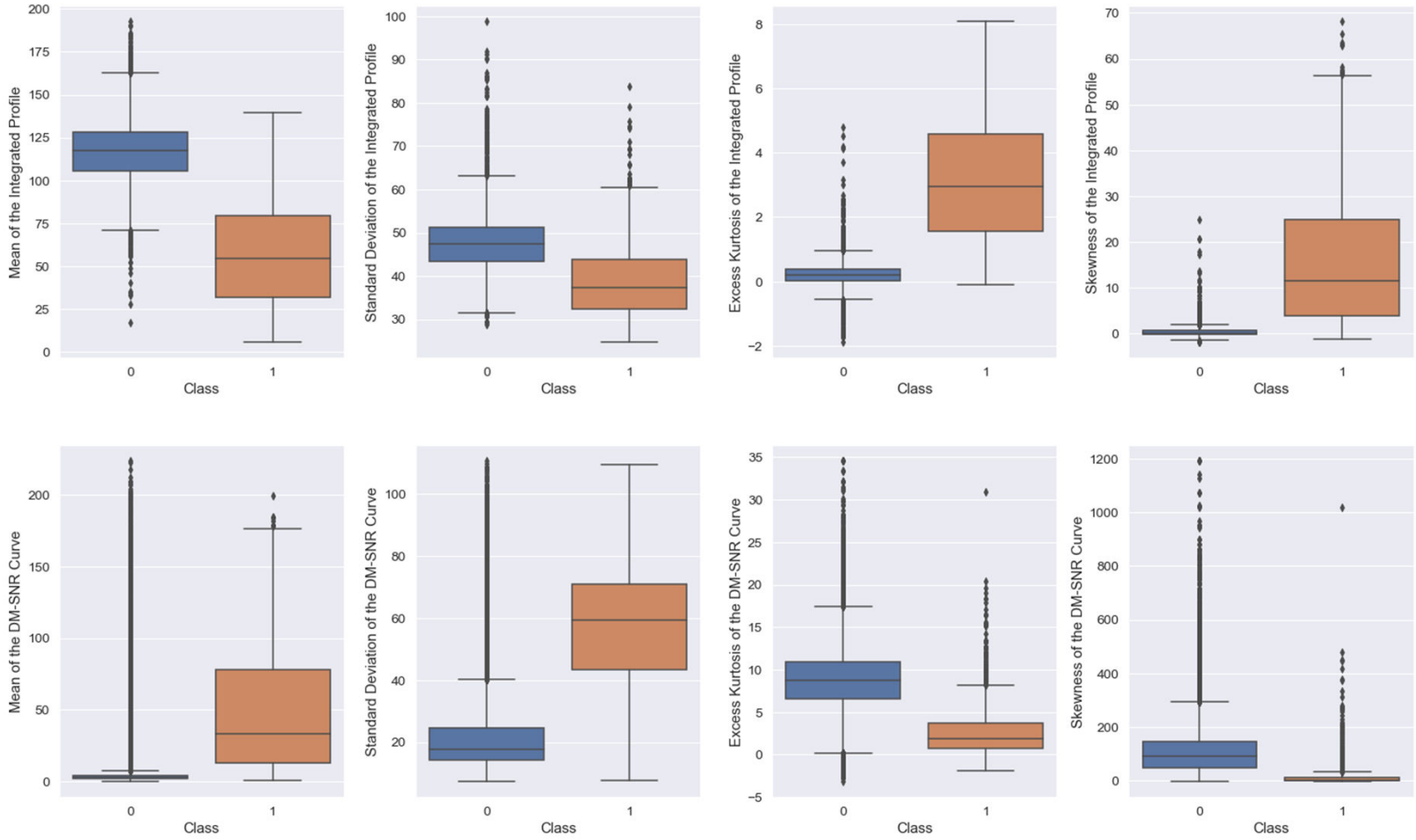
PULSAR PROJECT

Bar Plots of Class for Each Feature



PULSAR PROJECT

Box Plots of Class for Each Feature



Models

There are a number of machine learning models that can be used for this project. Since this is a binary classification problem, the following models were built:

- **Logistic Regression**
- **K-Nearest Neighbors (KNN)**
- **Support Vector Machine (SVM)**
- **Decision Tree**
- **Random Forest**
- **Extra Trees**
- **AdaBoost**
- **Gradient Boosting**
- **Gaussian Naive Bayes**
- **Bernoulli Naive Bayes**
- **Stochastic Gradient Descent**
- **XGBoost**

Models

In addition, a few artificial neural networks were also built.

Using Scikit-learn:

- **Multilayer Perceptron (MLP)**

Using Keras:

- **MLP with 1 Hidden Layer**
- **MLP with 2 Hidden Layers**
- **MLP with 3 Hidden Layers**
- **MLP with 1 Hidden Layer and 1 Dropout Layer**
- **MLP with 2 Hidden Layers and 1 Dropout Layer**

Due to time and computing restraints, full model optimization by tuning all hyperparameters was not possible. Instead, where applicable, a few top hyperparameters will be tuned before building the final model. Some tuning will be done separately in steps to allow quicker results. Ideally, everything would tune together, but this quickly results in excessive amounts of model cases to be tested.

Model Optimization of Selected Models

K-Nearest Neighbors (KNN) - n_neighbors
Support Vector Machine (SVM) - kernel, C, and gamma
Decision Tree - max_depth and min_samples_split
Random Forest - criterion; n_estimators; max_depth and min_samples_split
Extra Trees - n_estimators; max_depth and min_samples_split
AdaBoost - n_estimators
Gradient Boosting - n_estimators
Stochastic Gradient Descent - alpha and penalty
XGBoost - max_depth and min_child_weight

Using Scikit-learn:
Multilayer Perceptron (MLP) - hidden_layer_sizes; activation and solver; batch_size

Using Keras:
MLP with 1 Hidden Layer - batch_size and epochs; optimizer; init_mode; activation; neurons
MLP with 2 Hidden Layers - neurons
MLP with 3 Hidden Layers - neurons
MLP with 1 Hidden Layer and 1 Dropout Layer - weight_constraint and dropout_rate
MLP with 2 Hidden Layers and 1 Dropout Layer - neurons

Model Evaluation

There are a number of metrics that were used to evaluate the performance of the final models.

Confusion Matrix

- True Negative (TN) - The cases where it is actually negative and the model predicts negative.
- False Positive (FP) - The cases where it is actually negative but the model predicts positive.
- False Negative (FN) - The cases where it is actually positive but the model predicts negative.
- True Positive (TP) - The cases where it is actually positive and the model predicts positive.

Accuracy - Ratio of correct predictions over total predictions **(TP+TN)/(TP+TN+FP+FN)**

Precision - Expresses the proportion of the data points the model says was relevant actually were relevant **TP/(TP+FP)**

Recall - Expresses the ability to find all relevant instances in a dataset **TP/(TP+FN)**

F1 Score - The harmonic mean of precision and recall taking both metrics into account **2*(precision*recall)/(precision+recall)**

False Positive Rate - (aka Specificity) Corresponds to the proportion of negative data points that are mistakenly considered as positive, with respect to all negative data points **FP/(TN+FP)**

Note: In this project, negative represents the non-pulsar class and positive represents the pulsar class.

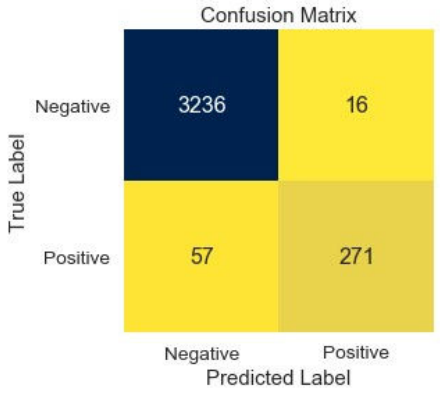
Model Evaluation

For this project, the final ranking will be based on accuracy score.
When model optimization is performed the models will be tuned to achieve the best accuracy score.
A final model will be built using optimized hyperparameters.
To reduce bias, a 5-fold cross-validation will be run on the final model and the mean accuracy score will be used.

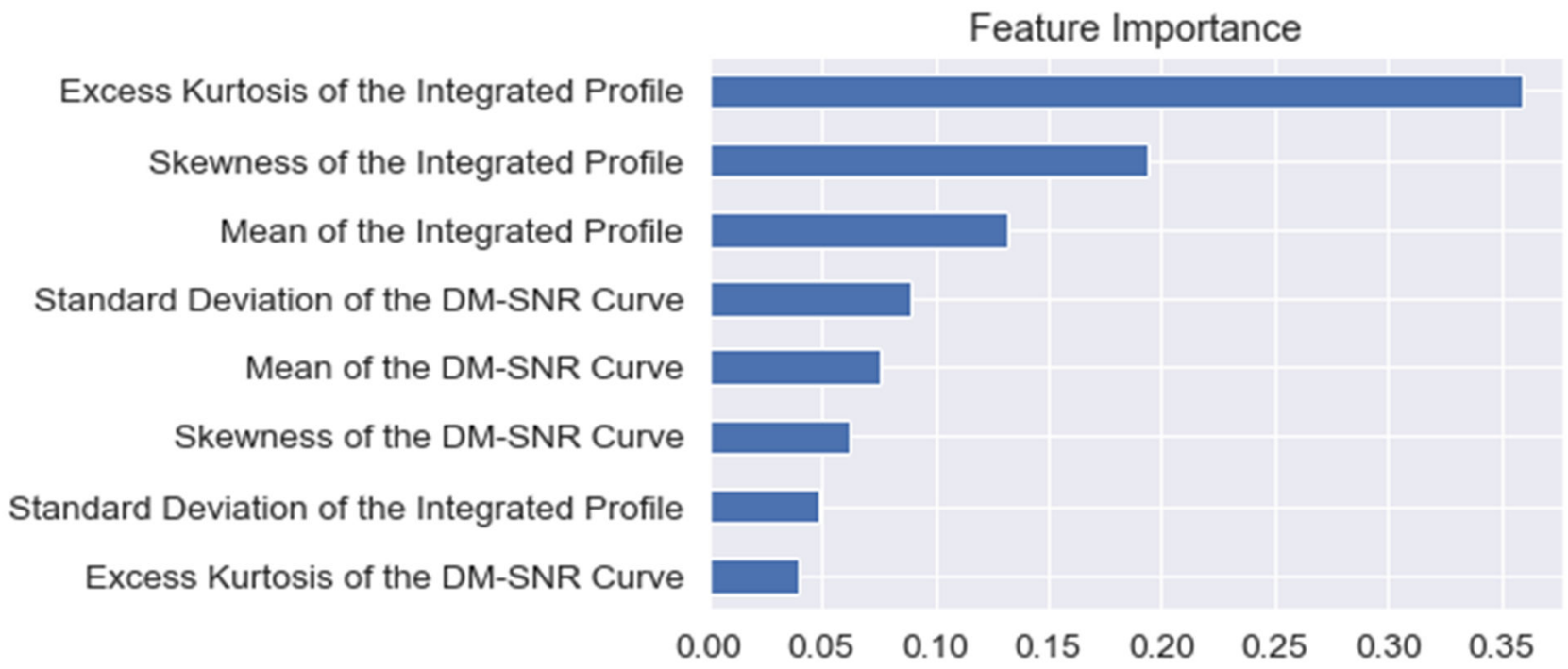
Sample Output
For Logistic Regression
Model

```
5-Fold Cross-Validation Accuracy Scores:  [0.97458101 0.97905028 0.98212291 0.97765363 0.97792063]

Model                                  Logistic Regression
Accuracy                                          0.979609
Precision                                         0.944251
Recall                                             0.82622
F1 Score                                          0.881301
False Positive Rate                             0.00492005
5-Fold Cross-Validation Accuracy Score            0.978266
dtype: object
```



Feature Importance



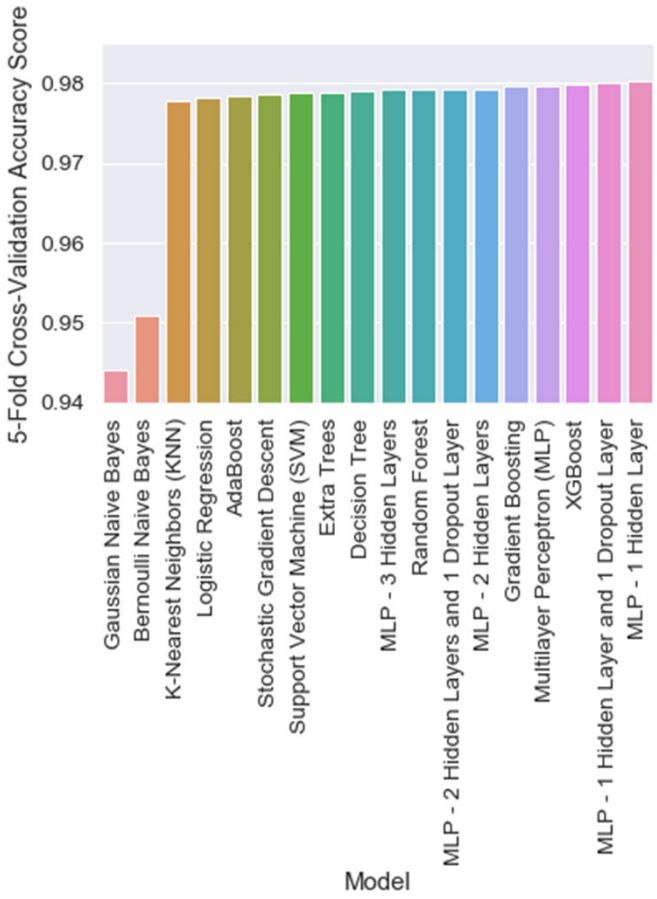
Results

Model	Accuracy	Precision	Recall	F1 Score	False Positive Rate	5-Fold Cross-Validation Accuracy Score
Logistic Regression	0.979609	0.944251	0.82622	0.881301	0.00492005	0.978266
K-Nearest Neighbors (KNN)	0.980447	0.932886	0.847561	0.888179	0.00615006	0.977707
Support Vector Machine (SVM)	0.980168	0.950877	0.82622	0.884176	0.00430504	0.978713
Decision Tree	0.980726	0.930233	0.853659	0.890302	0.00645756	0.978936
Random Forest	0.980726	0.919094	0.865854	0.89168	0.00768758	0.979271
Extra Trees	0.980726	0.921824	0.862805	0.891339	0.00738007	0.97888
AdaBoost	0.980447	0.935811	0.844512	0.887821	0.00584256	0.978321
Gradient Boosting	0.979888	0.923841	0.85061	0.885714	0.00707257	0.979607
Gaussian Naive Bayes	0.942737	0.638202	0.865854	0.734799	0.049508	0.944018
Bernoulli Naive Bayes	0.952514	0.687204	0.884146	0.773333	0.0405904	0.950833
Stochastic Gradient Descent	0.979888	0.941379	0.832317	0.883495	0.00522755	0.978601
XGBoost	0.981006	0.924837	0.862805	0.892744	0.00707257	0.979886
Multilayer Perceptron (MLP)	0.981844	0.931148	0.865854	0.897314	0.00645756	0.979718
MLP - 1 Hidden Layer	0.982123	0.925806	0.875	0.899687	0.00707257	0.980166
MLP - 2 Hidden Layers	0.981006	0.933333	0.853659	0.89172	0.00615006	0.979328
MLP - 3 Hidden Layers	0.980726	0.911111	0.875	0.892691	0.00861009	0.979216
MLP - 1 Hidden Layer and 1 Dropout Layer	0.981844	0.925566	0.871951	0.897959	0.00707257	0.98011
MLP - 2 Hidden Layers and 1 Dropout Layer	0.981285	0.904025	0.890244	0.897081	0.0095326	0.979328

Results Sorted By 5-Fold Cross-Validation Accuracy Score

Model	Accuracy	Precision	Recall	F1 Score	False Positive Rate	5-Fold Cross-Validation Accuracy Score
MLP - 1 Hidden Layer	0.982123	0.925806	0.875	0.899687	0.00707257	**0.980166**
MLP - 1 Hidden Layer and 1 Dropout Layer	0.981844	0.925566	0.871951	0.897959	0.00707257	**0.98011**
XGBoost	0.981006	0.924837	0.862805	0.892744	0.00707257	**0.979886**
Multilayer Perceptron (MLP)	0.981844	0.931148	0.865854	0.897314	0.00645756	**0.979718**
Gradient Boosting	0.979888	0.923841	0.85061	0.885714	0.00707257	**0.979607**
MLP - 2 Hidden Layers	0.981006	0.933333	0.853659	0.89172	0.00615006	**0.979328**
MLP - 2 Hidden Layers and 1 Dropout Layer	0.981285	0.904025	0.890244	0.897081	0.0095326	**0.979328**
Random Forest	0.980726	0.919094	0.865854	0.89168	0.00768758	**0.979271**
MLP - 3 Hidden Layers	0.980726	0.911111	0.875	0.892691	0.00861009	**0.979216**
Decision Tree	0.980726	0.930233	0.853659	0.890302	0.00645756	**0.978936**
Extra Trees	0.980726	0.921824	0.862805	0.891339	0.00738007	**0.97888**
Support Vector Machine (SVM)	0.980168	0.950877	0.82622	0.884176	0.00430504	**0.978713**
Stochastic Gradient Descent	0.979888	0.941379	0.832317	0.883495	0.00522755	**0.978601**
AdaBoost	0.980447	0.935811	0.844512	0.887821	0.00584256	**0.978321**
Logistic Regression	0.979609	0.944251	0.82622	0.881301	0.00492005	**0.978266**
K-Nearest Neighbors (KNN)	0.980447	0.932886	0.847561	0.888179	0.00615006	**0.977707**
Bernoulli Naive Bayes	0.952514	0.687204	0.884146	0.773333	0.0405904	**0.950833**
Gaussian Naive Bayes	0.942737	0.638202	0.865854	0.734799	0.049508	**0.944018**

Results Sorted By 5-Fold Cross-Validation Accuracy Score

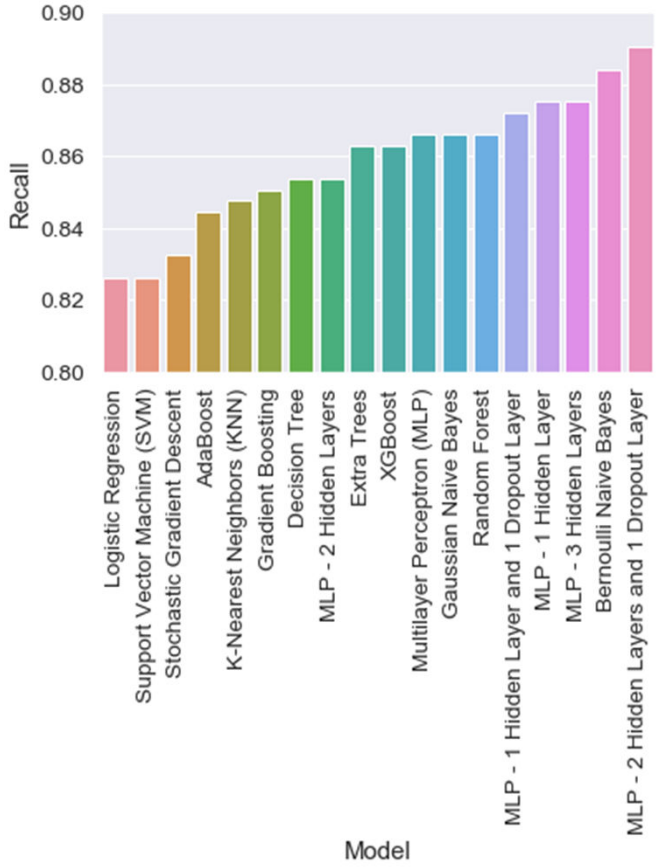


Using 5-Fold Cross-Validation Accuracy Score as the metric, **MLP - 1 Hidden Layer** appears to be the best model.

Results Sorted By Recall

Model	Accuracy	Precision	Recall	F1 Score	False Positive Rate	5-Fold Cross-Validation Accuracy Score
MLP - 2 Hidden Layers and 1 Dropout Layer	0.981285	0.904025	**0.890244**	0.897081	0.0095326	0.979328
Bernoulli Naive Bayes	0.952514	0.687204	**0.884146**	0.773333	0.0405904	0.950833
MLP - 1 Hidden Layer	0.982123	0.925806	**0.875**	0.899687	0.00707257	0.980166
MLP - 3 Hidden Layers	0.980726	0.911111	**0.875**	0.892691	0.00861009	0.979216
MLP - 1 Hidden Layer and 1 Dropout Layer	0.981844	0.925566	**0.871951**	0.897959	0.00707257	0.98011
Random Forest	0.980726	0.919094	**0.865854**	0.89168	0.00768758	0.979271
Gaussian Naive Bayes	0.942737	0.638202	**0.865854**	0.734799	0.049508	0.944018
Multilayer Perceptron (MLP)	0.981844	0.931148	**0.865854**	0.897314	0.00645756	0.979718
Extra Trees	0.980726	0.921824	**0.862805**	0.891339	0.00738007	0.97888
XGBoost	0.981006	0.924837	**0.862805**	0.892744	0.00707257	0.979886
Decision Tree	0.980726	0.930233	**0.853659**	0.890302	0.00645756	0.978936
MLP - 2 Hidden Layers	0.981006	0.933333	**0.853659**	0.89172	0.00615006	0.979328
Gradient Boosting	0.979888	0.923841	**0.85061**	0.885714	0.00707257	0.979607
K-Nearest Neighbors (KNN)	0.980447	0.932886	**0.847561**	0.888179	0.00615006	0.977707
AdaBoost	0.980447	0.935811	**0.844512**	0.887821	0.00584256	0.978321
Stochastic Gradient Descent	0.979888	0.941379	**0.832317**	0.883495	0.00522755	0.978601
Support Vector Machine (SVM)	0.980168	0.950877	**0.82622**	0.884176	0.00430504	0.978713
Logistic Regression	0.979609	0.944251	**0.82622**	0.881301	0.00492005	0.978266

Results Sorted By Recall

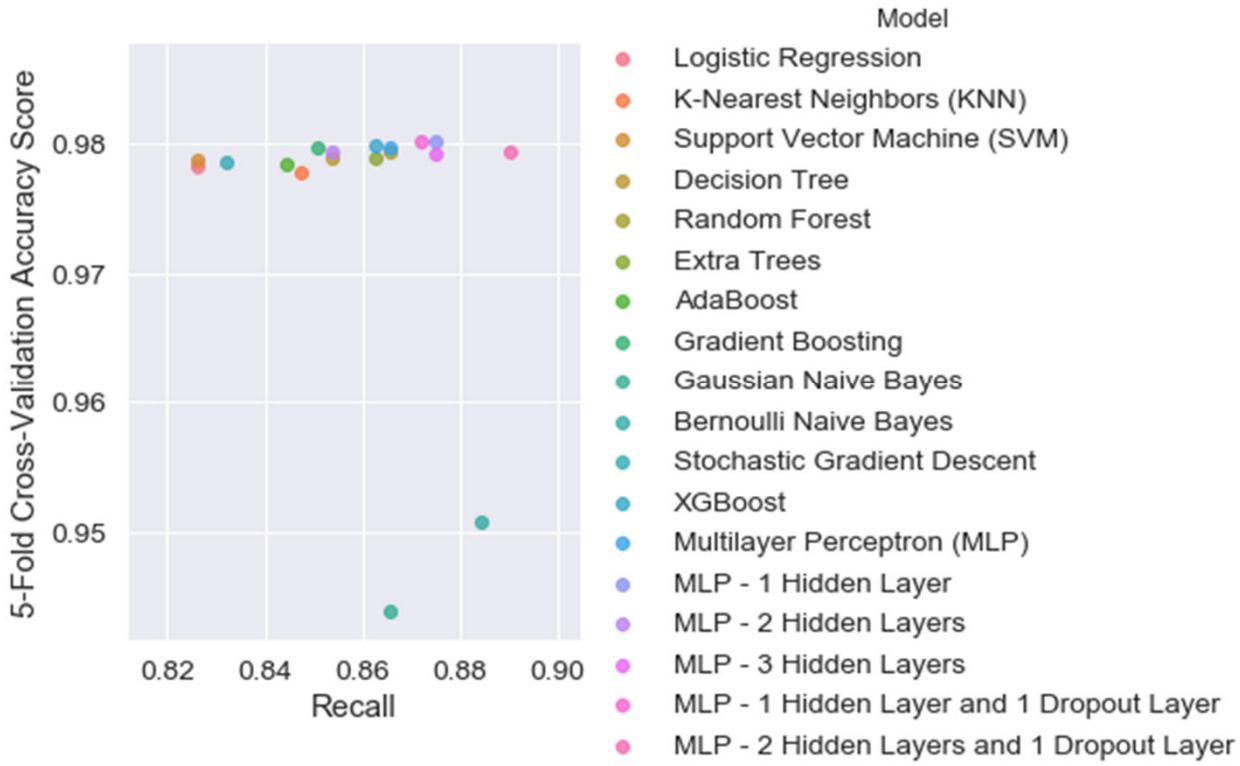


Using Recall as the metric, **MLP - 2 Hidden Layers and 1 Dropout Layer** appears to be the best model.

Note that the models were tuned with accuracy score in mind. The rankings could be different if they were tuned to achieve the best recall rate instead.

PULSAR PROJECT

Which Models Performed the Best with Both Accuracy Score and Recall?



The upper-right corner of the plot contains the best models for this project based on recall and accuracy score. An argument could be made for either MLP - 1 Hidden Layer or MLP - 2 Hidden Layers and 1 Dropout Layer.

Conclusion

The best model created to classify candidates based on non-pulsar and pulsar classes was the artificial neural network multilayer perceptron with one hidden layer (**MLP - 1 Hidden Layer**). This is based solely on the 5-fold cross-validation accuracy scores.

When recall, the percentage of positive cases that were caught, is considered, the artificial neural network multilayer perceptron with two hidden layers and 1 dropout layer (**MLP - 2 Hidden Layers and 1 Dropout Layer**) performs quite well. A better recall rate could be achieved by tuning various hyperparameters to maximize this instead of accuracy score.

Further hyperparameter tuning could produce even better models. Some hyperparameters were not tuned or tuned only at a basic level. More time and resources would allow more thorough testing of promising models. Feature engineering may also yield improved modelling. Additional layers and tuning with the artificial neural networks multilayer perceptron could also potentially generate better models.

The Future

The **SQUARE KILOMETRE ARRAY** will be two vast arrays of radio telescopes: one in Africa and one in Australia.