Project - PulsarDetection

Daniele

SeyedHossein Javadizavieh

S289615@studenti.polito.it

S290144@studenti.polito.it

Politecnico di Torino

Outline

* Introduction
* Pulsar Features
* Building a classifier for the Avila task
* Experimental validation
* Conclusions

Introduction

The Pulsar dataset is taken from the UCI repository (Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science)

HTRU2 is a data set which describes a sample of pulsar candidates collected during the High Time Resolution Universe Survey (South) [1].

Pulsars are a rare type of Neutron star that produce radio emission detectable here on Earth. They are of considerable scientific interest as probes of space-time, the inter-stellar medium, and states of matter (see [2] for more uses).

As pulsars rotate, their emission beam sweeps across the sky, and when this crosses our line of sight, produces a detectable pattern of broadband radio emission. As pulsars rotate rapidly, this pattern repeats periodically. Thus pulsar search involves looking for periodic radio signals with large radio telescopes.

Each pulsar produces a slightly different emission pattern, which varies slightly with each rotation (see [2] for an introduction to pulsar astrophysics to find out why). Thus 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 info, 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.

We keep the original train / evaluation split

The training set contains 17,898 total examples with 639 positive examples and 16,259 negative examples.

We need the data to Train and Test the methods. Regarding to the two datasets offered, we will use all of the Train Dataset for Training and all of the Test Dataset for Testing.

----------------------------------------------------------------------------------------------------------------------------------------------

Source:

Dr. Robert Lyon, University of Manchester, School of Physics and Astronomy, Alan Turing Building, Manchester M13 9PL, United Kingdom, robert.lyon '@' manchester.ac.uk

[1] M. J. Keith et al., 'The High Time Resolution Universe Pulsar Survey - I. System Configuration and Initial Discoveries',2010, Monthly Notices of the Royal Astronomical Society, vol. 409, pp. 619-627. DOI: 10.1111/j.1365-2966.2010.17325.x

[2] D. R. Lorimer and M. Kramer, 'Handbook of Pulsar Astronomy', Cambridge University Press, 2005.

Pulsar Features

Each candidate is described by 8 continuous variables, and a single class variable. The first four are simple statistics obtained from the integrated pulse profile (folded profile).

This is an array of continuous variables that describe a longitude-resolved version of the signal that has been averaged in both time and frequency (see [3] for more details). The remaining four variables are similarly obtained from the DM-SNR curve (again see [3] for more details). These are summarized 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

Histogram of the Pulsar Detection dataset features without preprocessing (training set). Features are sorted by their order, from left to right, top to bottom.

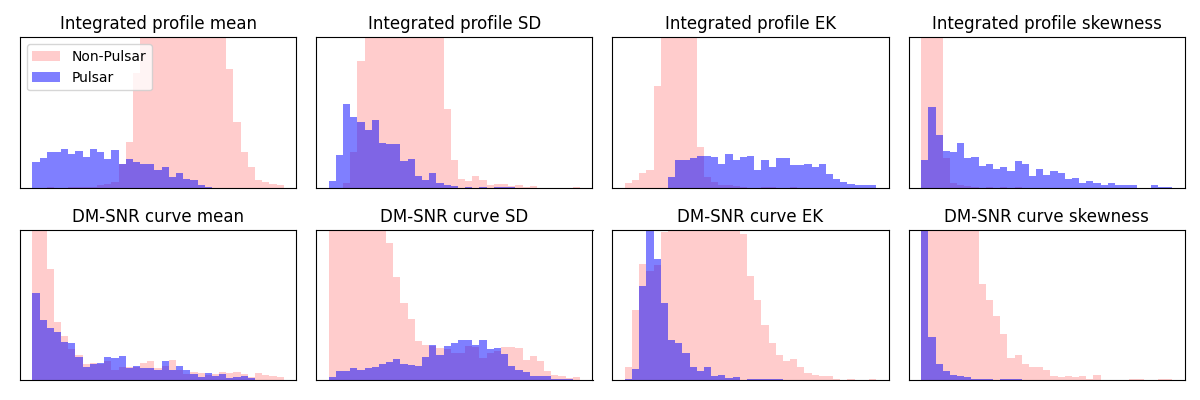


Figure 1: Histogram of the Pulsar Detection dataset features without preprocessing

----------------------------------------------------------------------------------------------------------------------------------------------

[3] R. J. Lyon, 'Why Are Pulsars Hard To Find?', PhD Thesis, University of Manchester, 2016

Pulsar Features

We have to decide which machine learning method would be best. also, we will use Gaussian Model, Logistic Regression, Support Vector Machines and Gaussian Mixture Model.

Then, we can compare methods by seeing how well each one categorized the test data. We assume that this data for training and testing are the best way to divide up the data.

In many cases an analysis of the training data shows that, we have many irregular distributions in the raw features, characterized by a presence of significantly large outliers.

Due to the presence of outliers, we expect that classification approaches may produce sub-optimal results (especially Gaussian-based methods)

We therefore further pre-process data and "Gaussianize" the features.

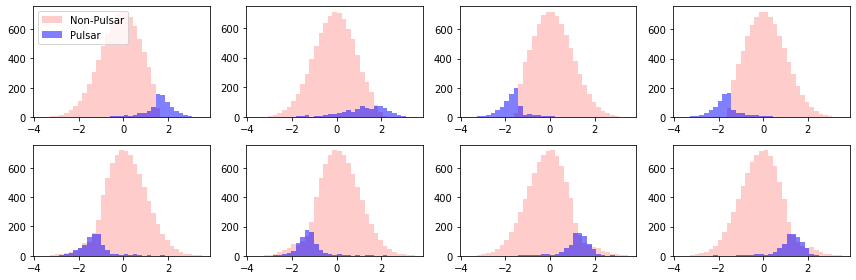


Figure 2: Histogram of the Pulsar Detection dataset features after gaussianize

Pulsar Features

Gaussianization is a procedure that allows mapping a set of features to values whose empirical cumulative distribution function is well approximated by a Gaussian c.d.f.

The processing consists in mapping the features to a uniform distribution and then transforming the mapped features through the inverse of Gaussian cumulative distribution function

Data preprocessing allows for the removal of unwanted data with the use of data cleaning, this allows to have a dataset to contain more valuable information after the preprocessing stage for data manipulation later in the data mining process. Also preprocess reduced the minimum DCF in many models.

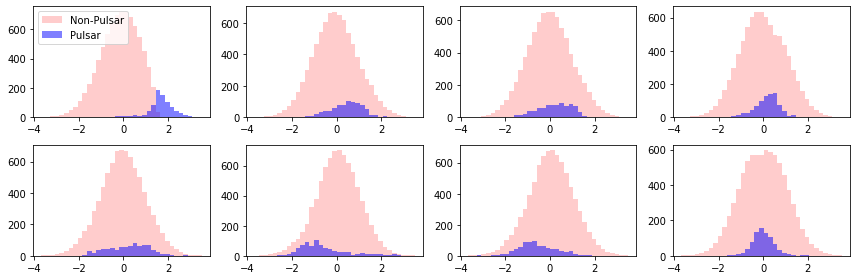
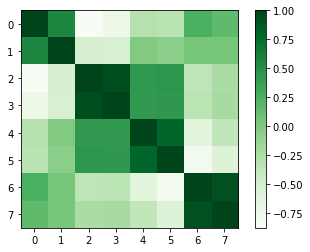
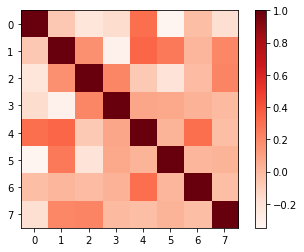
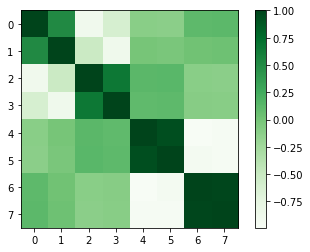
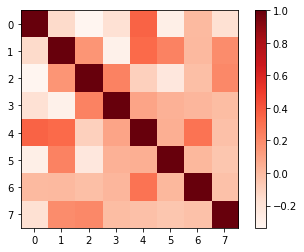


Figure 3: Histogram of the Pulsar Detection dataset features after preprocessing (center\_features() -

standardize\_variance() - whiten\_covariance() - normalize\_lenght()) and gaussianize

In correlation we are using Pearson Correlation Coefficient, we can see that before preprocessing there is correlation between data. Also the correlation of Gaussianization have been reduced a bit.

Pulsar Features

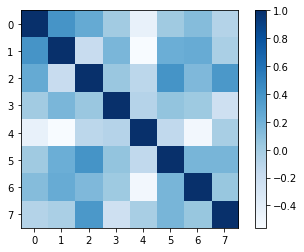
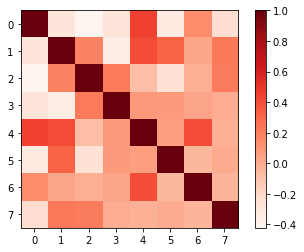
   

Left green shows the correlation coefficient of training features whole dataset

Left red shows the correlation coefficient of all preprocessed features whole dataset

Right green shows the correlation coefficient gaussianized of training features whole dataset

Right red shows the correlation coefficient of gaussianized and preprocessed features whole dataset

Blue plot shows the correlation coefficient of the true Pulsar detection

Red plot shows the correlation coefficient of the false Pulsar detection

Classifying Pulsar features

We start considering simple Gaussian classifiers

Although we have used Gaussianization over the whole dataset, the histograms for each class show, in some cases, moderate deviations from the Gaussian assumptions.

Since within-class covariance matrices are almost diagonal, we consider both diagonal and full-covariance models

To understand which model is most promising we can adopt two methodologies:

* We can split the training dataset into development (for model training) and validation subsets (single-fold in the following)
* We can employ K-Fold cross-validation

By the way, K-Fold cross-validation uses them all one at a time to which block would be best for testing, and summarizes the results at the end. It keeps track of how well the method did with the test data. Then it uses combination of blocks to train the method.

Cross-validation allows us to compare different machine learning methods and get a sense of how well they will work in practice.

Single split:

* The final classifier will be the same that we evaluate on the validation set: model selection and hyper-parameter will be optimal at least for the validation set
* We need to train fewer models, so training is faster
* We have fewer data for validation and model training

Classifying Pulsar features

K-Fold cross-validation:

* More data available for training and validation
* The final classifier will be obtained by re-training over the whole training set, so it will leverage additional data

Classifying Pulsar features

* Decisions are made over the validation set for the models trained using folds. They may not be optimal for the model learned from all training data

In this method, we divided the Train Dataset into 5 blocks (4 blocks for train, 1 for test). The K-Fold is implemented with K=5. Data has been shuffled before splitting.

In the end, every block of data is used for testing and we can compare methods by seeing how well they performed.

Min DCF measures the cost we would pay if we made optimal decisions for the test set (in our case the validation set) using the recognizer score.

Here we should put MVG DCF

Classifying Pulsar features

According to the table of parameters below, we get that the data of pulsar and non-pulsar don’t work well with the gaussian distribution assumption and that is clear for the raw data.

**Gaussian Model**

|  |  |  |  |
| --- | --- | --- | --- |
| **Method \ prior** | **0.1** | **0.5** | **0.9** |
| Without Preprocessing | 1.0 | 1.0 | 1.055 |
| With Preprocessing | 1.0 | 0.613 | 0.771 |
| Gaussianized | 1.0 | 1.0 | 1.055 |

**Naïve**

|  |  |  |  |
| --- | --- | --- | --- |
| **Method \ prior** | **0.1** | **0.5** | **0.9** |
| Without Preprocessing | 1.0 | 1.0 | 1.055 |
| With Preprocessing | 1.0 | 1.0 | 1.055 |
| Gaussianized | 1.0 | 1.0 | 1.055 |

**Tied covariance**

|  |  |  |  |
| --- | --- | --- | --- |
| **Method \ prior** | **0.1** | **0.5** | **0.9** |
| Without Preprocessing | 1.0 | 1.0 | 1.055 |
| With Preprocessing | 1.0 | 0.997 | 1.008 |
| Gaussianized | 1.0 | 0.999 | 1.055 |

The tied models perform in general worse. Again, the covariance matrices are fairly similar, however, since we have enough data, estimating separately the matrices provides a better model

Gaussianization did not give us the desired performance. Overall, the best candidate is currently the Linear Regression and we start considering regularized (linear) Logistic Regression.

Classifying Pulsar features

**Logistic Regression**

For Analyzing the Log-Reg risk, we assume that the natural choice seems $\lambda \to 0$

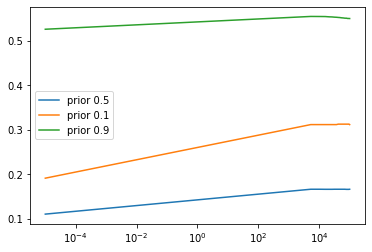
**Linear Case:**

|  |  |
| --- | --- |
| C:\Users\Hossein.JvdZ\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Log Reg Prep Plot.png  Log-Reg Preprocessed, Risk 0.5 | C:\Users\Hossein.JvdZ\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Linear Log-Reg Prep 0.1 Risk Plot.png  Log-Reg Preprocessed, Risk 0.1 |
| C:\Users\Hossein.JvdZ\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Log Reg Gaussianized Plot.png  Log-Reg Gaussianized, Risk 0.5 | C:\Users\Hossein.JvdZ\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Log Reg Plot.png  Log-Reg Neither Gaussianized nor Prep, Risk 0.1 |

Classifying Pulsar features

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| $\lambda = 1e-5$, rebalanced 0.5   |  |  |  |  |  | | --- | --- | --- | --- | --- | | Method \ prior |  | 0.1 | 0.5 | 0.9 | | **Without Preprocessing** |  | 0.579 | 0.191 | 0.700 | | **With Preprocessing** |  | 0.207 | 0.124 | 0.554 | | **Gaussianized** |  | 0.440 | 0.173 | 0.464 | | $\lambda = 1e-5$, rebalanced 0.1   |  |  |  |  | | --- | --- | --- | --- | | Method \ prior | 0.1 | 0.5 | 0.9 | | **Without Preprocessing** |  |  |  | | **With Preprocessing** | 0.198 | 0.107 | 0.542 | | **Gaussianized** | 0.194 | 0.104 | 0.503 | |

**Quadratic Case:**



Log-Reg Preprocessed, Risk 0.1

Natural choice seems $\lambda \to 0$

$\lambda = 1e-5$, rebalanced 0.1

|  |  |  |  |
| --- | --- | --- | --- |
| Method \ prior | 0.1 | 0.5 | 0.9 |
| **Without Preprocessing** |  |  |  |
| **With Preprocessing** | 0.195 | 0.108 | 0.481 |
| **Gaussianized** |  |  |  |

Classifying Pulsar features

Regularization provides little to no benefit. Best results are obtained with small values of λ. Setting λ = 0 provides the same performance.

Gaussianization seems much less relevant. Indeed, Logistic Regression does require assumptions on the data distribution.

Overall, the Log-Reg model perform better other models.

The conclusions are similar as for the linear case and Regularization is not required, Pre-processing (Gaussianization) is significantly more helpful in this case

If we are interested in different applications it may be worth using a different model for each of the three applications, although the benefits are small in this case.

The previous analysis suggests that the Gaussian assumption may not be sufficiently accurate for our features

Furthermore, classes cannot be well separated by linear decision rules

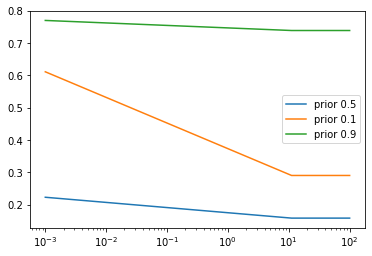
Therefore we turn our attention to SVMs and to Gaussian Mixture Models, We start with analyzing linear SVM, For linear SVM, we need to tune the hyper-parameter C.

Classifying Pulsar features

### Support Vector Machine

For Analyzing the SVM risk, we assume that the natural choice seems $\C \to 0$ for preprocessed and gaussianized data

**Linear Case:**



SVM Neither Prep nor Gauss, Prior 0.5

|  |  |
| --- | --- |
| C:\Users\Hossein.JvdZ\Desktop\ML\Pulsar\Plots\SVM\Linear\SVM Prep Plot Prior 0.5.png  SVM Preprocessed, Prior 0.5 | C:\Users\Hossein.JvdZ\Desktop\ML\Pulsar\Plots\SVM\Linear\SVM Prep Plot NoPrior.png  SVM Preprocessed, NoPrior |
| C:\Users\Hossein.JvdZ\Desktop\ML\Pulsar\Plots\SVM\Linear\SVM Gaussianized Plot Prior 0.5.png  SVM Gaussianized, Prior 0.5 | C:\Users\Hossein.JvdZ\Desktop\ML\Pulsar\Plots\SVM\Linear\SVM Gaussianized Plot NoPrior.png  SVM Gaussianized, NoPrior |

Classifying Pulsar features

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| $C = 1e-3$, rebalanced 0.5   |  |  |  |  | | --- | --- | --- | --- | | Method \ prior | 0.1 | 0.5 | 0.9 | | **Without Preprocessing** | 0.607 | 0.221 | 0.774 | | **With Preprocessing** | 0.226 | 0.110 | 0.517 | | **Gaussianized** | 0.202 | 0.111 | 0.531 | | $C = 1e-3$, no rebalanced   |  |  |  |  | | --- | --- | --- | --- | | Method \ prior | 0.1 | 0.5 | 0.9 | | **Without Preprocessing** |  |  |  | | **With Preprocessing** | 0.226 | 0.167 | 0.702 | | **Gaussianized** | 0.227 | 0.150 | 0.686 | |

We can compare Preprocessed linear models in terms of min DCF:

|  |  |  |  |
| --- | --- | --- | --- |
|  | π = 0.5 | π = 0.1 | π = 0.9 |
| MVG (Tied Full-Cov) | … | … | … |
| Log-Reg (λ = 10-5 , πT = 0.1) | 0.207 | 0.124 | 0.554 |
| Log-Reg (λ = 10-5 , πT = 0.5) | 0.198 | 0.107 | 0.542 |
| Linear SVM (C = 0.1) | 0.226 | 0.167 | 0.702 |
| Linear SVM (C = 0.1, πT = 0.5) | 0.226 | 0.110 | 0.517 |

Linear SVM performs similarly to other linear approaches, as expected.

Class re-balancing is not necessary, therefore we will use the default SVM formulation.

Since non-linear models perform better on this dataset, we consider two non-linear SVM formulations

The first will use a polynomial quadratic kernel (similar to the quadratic Logistic Regression model, again we expect similar results)

The second will employ a Radial Basis Function kernel

Classifying Pulsar features

For the RBF kernel we also need to estimate the kernel width Gamma.

We will use a grid search to jointly optimize C and Gamma

We will mainly focus on Gaussianized features

**Quadratic Case:**

|  |  |
| --- | --- |
| C:\Users\Hossein.JvdZ\Desktop\ML\Pulsar\Plots\SVM\Quadratic\Quadratic SVM Prep Prior 0.5 Plot.png  SVM Preprocessed, Prior 0.5 | C:\Users\Hossein.JvdZ\Desktop\ML\Pulsar\Plots\SVM\Quadratic\Quadratic SVM Prep Prior 0.1 Plot.png  SVM Preprocessed, Prior 0.1 |
| C:\Users\Hossein.JvdZ\Desktop\ML\Pulsar\Plots\SVM\Quadratic\Quadratic SVM Prep Prior 0.1 Plot.png  SVM Gaussianized, Prior 0.5 |

Classifying Pulsar features

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| $C = 1e-3$, rebalanced 0.5   |  |  |  |  | | --- | --- | --- | --- | | Method \ prior | 0.1 | 0.5 | 0.9 | | **Without Preprocessing** |  |  |  | | **With Preprocessing** | 0.203 | 0.110 | 0.489 | | **Gaussianized** | 0.221 | 0.110 | 0.499 | | $C = 1e-3$, rebalanced 0.1   |  |  |  |  | | --- | --- | --- | --- | | Method \ prior | 0.1 | 0.5 | 0.9 | | **Without Preprocessing** |  |  |  | | **With Preprocessing** | 0.194 | 0.107 | 0.518 | | **Gaussianized** |  |  |  | |

We can compare Preprocessed quadratic models in terms of min DCF:

|  |  |  |  |
| --- | --- | --- | --- |
|  | π = 0.5 | π = 0.1 | π = 0.9 |
| MVG (Tied Full-Cov) | … | … | … |
| Log-Reg (λ = 10-5 , πT = 0.1) | 0.195 | 0.108 | 0.481 |
| Quadratic SVM (C = 0.1) | 0.194 | 0.107 | 0.518 |
| Quadratic SVM (C = 0.1, πT = 0.5) | 0.203 | 0.110 | 0.489 |

Quadratic kernel SVM provides slightly worse results than Logistic Regression.

#### Cubic Case

* Analizing the risk the natural choice seems $C \to 0$
* From the table we don’t see improvements from quadratic model

$C = 1e-3$, rebalanced 0.1

| **Method \ prior** | **0.1** | **0.5** | **0.9** |
| --- | --- | --- | --- |
| No prep |  |  |  |
| Prep | 0.198 | 0.109 | 0.622 |
| Gaussianized |  |  |  |

#### RBF

* Analizing the risk the natural choice seems $C \to 0$ and $\gamma = 0.1$ for Gaussianized and $\gamma = 10$ for preprocessed

$C=1e-3, \gamma = 10$

| **Method \ prior** | **0.1** | **0.5** | **0.9** |
| --- | --- | --- | --- |
| No prep |  |  |  |
| Prep | 0.200 | 0.107 | 0.581 |
| Gaussianized | 0.217 | 0.115 | 0.644 |

$C = 1e-3, \gamma = 0.1$

| **Method \ prior** | **0.1** | **0.5** | **0.9** |
| --- | --- | --- | --- |
| No prep |  |  |  |
| Prep | 0.290 | 0.150 | 0.537 |
| Gaussianized | 0.214 | 0.113 | 0.647 |

* With prior for rebalancing 0.1

$C=1e-3, \gamma = 10$

| **Method \ prior** | **0.1** | **0.5** | **0.9** |
| --- | --- | --- | --- |
| No prep |  |  |  |
| Prep | 0.199 | 0.119 | 0.609 |
| Gaussianized | 0.912 | 0.391 | 0.898 |

$C = 1e-3, \gamma = 0.1$

| **Method \ prior** | **0.1** | **0.5** | **0.9** |
| --- | --- | --- | --- |
| No prep |  |  |  |
| Prep | 0.208 | 0.122 | 0.580 |
| Gaussianized | 0.204 | 0.123 | 0.577 |

$C= 1e-3, \gamma = 100$

| **Method \ prior** | **0.1** | **0.5** | **0.9** |
| --- | --- | --- | --- |
| No prep |  |  |  |
| Prep | 0.308 | 0.151 | 0.817 |
| Gaussianized |  |  |  |

### Gaussian Mixture Model

* Gaussianization doesn’t achieve good results in general
* Preprocessing is useful only for prior 0.5 e 0.9
* Gaussian Mixture Model is not benefiting from increasing the number of gaussians per class

number of gaussian = 2

| **Method \ prior** | **0.1** | **0.5** | **0.9** |
| --- | --- | --- | --- |
| No prep | 0.248 | 0.135 | 0.647 |
| Prep | 0.315 | 0.125 | 0.568 |
| Gaussianized | 0.840 | 0.277 | 0.680 |

number of gaussian = 4

| **Method \ prior** | **0.1** | **0.5** | **0.9** |
| --- | --- | --- | --- |
| No prep | 0.248 | 0.135 | 0.647 |
| Prep | 0.315 | 0.125 | 0.568 |
| Gaussianized | 0.840 | 0.277 | 0.680 |

number of gaussian = 8

| **Method \ prior** | **0.1** | **0.5** | **0.9** |
| --- | --- | --- | --- |
| No prep | 0.248 | 0.135 | 0.647 |
| Prep | 0.315 | 0.125 | 0.568 |
| Gaussianized | 0.840 | 0.277 | 0.680 |

number of gaussian = 32

| **Method \ prior** | **0.1** | **0.5** | **0.9** |
| --- | --- | --- | --- |
| No prep | 0.249 | 0.130 | 0.635 |
| Prep | 0.325 | 0.127 | 0.552 |
| Gaussianized |  |  |  |

number of gaussian = 64

| **Method \ prior** | **0.1** | **0.5** | **0.9** |
| --- | --- | --- | --- |
| No prep | 0.253 | 0.133 | 0.618 |
| Prep | 0.318 | 0.125 | 0.555 |
| Gaussianized | 0.741 | 0.315 | 0.741 |

Beside, we need to summarize how each method performed on the Training data. One way to do this is by creating a **Confusion Matrix** for each method.

The rows in a Confusion Matrix (It is especially called here predicted\_labels) corresponds to what the machine learning algorithm predicted and the columns (It is especially called here true\_labels) corresponds to the know truth.

Since there are only two categories to choose from: “Positives” or “Negatives”, then the bottom right-hand corner contains True Positives. These are the pulsars that had “Positives” that were correctly identified by the algorithm.

The True Negatives are in the top left-hand corner. These are the pulsars that did not have “Negatives” that were correctly identified by the algorithm.

The bottom left-hand corner contains the False Positives. These are pulsars has “Positives”, but the algorithm says they are.

Lastly, the top right-hand corner contains the False Negatives. These are when a pulsar has “Negatives”, but the algorithm said they didn’t.

The numbers along the diagonal (The True Positives and True Negatives) tell us how many times the samples were correctly classified.

The numbers not on the diagonal (the False Positives and False Negatives) are samples the algorithm messed up.

We can apply Logistic Regression to the Testing Dataset and create a Confusion Matrix.

----------------------

SVM

We can use Cross-Validation to determine that allowing this misclassification results in better classification in the long run.

--------------------

The first thing we do with the Gaussian Naive Bayes classifier is making an initial guess that they are detected as Pulsar. This guess can be any probability that we want, but a common guess is estimated from the training data. That initial guesses are called Prior Probabilities.

Note: the Likelihood is the y-axis coordinate on the curve that corresponds to the x-axis coordinate. And we multiply that by the Maximum Likelihood.

To talk about a Likelihood, we assume that we have already weighed the Pulsar (or Pulsars, if it is weighed more than one). We logged transforms the individual Likelihood functions.

Step 1) We have moved the log of the first Likelihood function for reference.

Step 2) We have converted the multiplication into addition.

Step 3) We have converted 1 over the square root into the exponent -0.5 and convert the exponent into multiplication.

Step 4) We have converted the -0.5 exponent into multiplication and the log of e = 1

Step 5) The log can convert the multiplication of 2 into addition of and 2

Step 6) We have converted the exponent log(2) into 2 log()

Step 7) Lastly, the 2 divided by 2 term cancels out.