



Pulsar classification project

machine learning and pattern recognition

academic year 2021-2022



Contents (1)

- Introduction
 - dataset topic and origin
 - classes distribution
 - sample features
 - features characteristics and distributions
 - correlation and covariance matrices
 - feature distributions
 - preprocessing and results



Contents (2)

- General informations about the classifiers
 - classification models used
 - validation set and working points
- Training and validation phase
 - multivariate Gaussian
 - logistic regression
 - support vector machines
 - Gaussian mixture model



Contents (3)

- Testing phase
- Conclusion



Introduction – dataset topic and origin

The dataset under consideration was taken from the online repository of the School of Information and Computer Science of the University of California.

The data themselves were collected by a professor at the School of Physics and Astronomy of the University of Manchester.

The main focus of this dataset is about pulsars: a type of Neutron star that produce different types of radio emission which can be detected using specialized tools (mainly large radio telescopes).

The main obstacles to the study of pulsars are mainly due to two factors:

- each pulsar generates radio emissions which are completely stochastic in nature: there is the need to average its emission in order to reach some kind of common pattern belonging to pulsars
- most of the detections of possible pulsars are due to noise or radio interferences instead

This second problem is the reason of the imbalance presented in the following page.



Introduction – classes distribution

The pulsar data is made of a total amount of 8929 samples concerning the training set, plus other 8969 that make up the test set.

One characteristic that is immediately evident is that both datasets are extremely unbalanced when it comes to the distribution of samples belonging to the “negative” (0) class with respect to the one of the “positive” class (1).

In fact it can be noted that in the training set the amount of samples belonging to class 0 is almost 10 times larger than the class 1 samples and in the test set the situation is pretty similar.

This behaviour is due to the point raised in the previous page: a critical problem in this field is the lack of quick methods to distinguish effective pulsar presence from different kind of noises.

In this case class 0 represents the presence of noise, while class 1 indicates that we are dealing with a “true” pulsar.



Introduction – sample features

Each sample is characterized by 8 distinct features and each of them are represented as floating point numbers:

- ◆ mean integrated profile
- ◆ standard deviation integrated profile
- ◆ excess kurtosis integrated profile
- ◆ skewness integrated profile
- ◆ mean DM-SNR
- ◆ standard deviation DM-SNR
- ◆ excess kurtosis DM-SNR
- ◆ skewness DM-SNR



Introduction - features characteristics and distributions

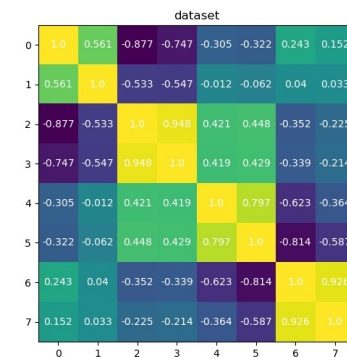
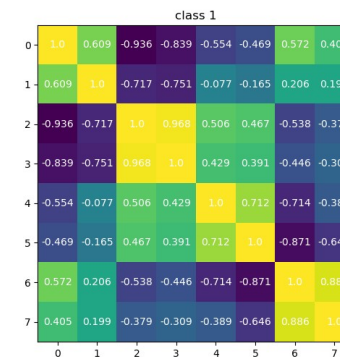
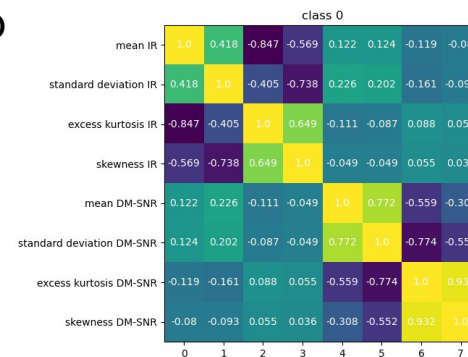
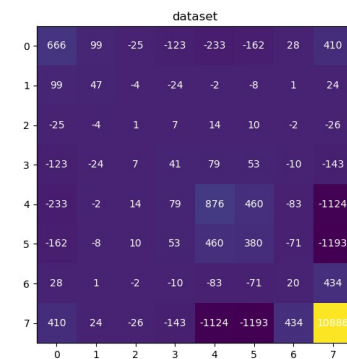
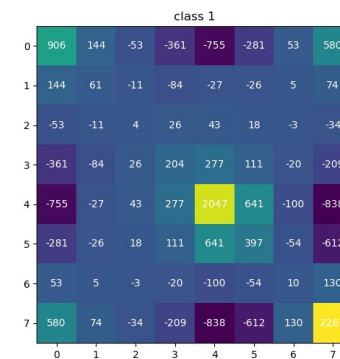
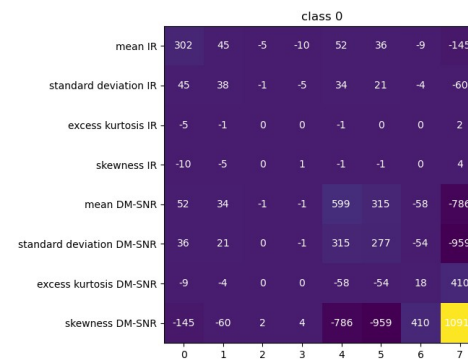
In order to gain a preliminary general idea about the characteristics of the input features, it could be useful to analyze the samples in the training set, hoping that they will give us some informations that could allow us to predict the behaviour of the classifier (to a certain extent). The approach was to gain some knowledge about the distributions of the features and if there exists a correlation between features and classes and between different features.

In total 3 different plot categories have been taken into considerations:

- feature distribution
- covariance matrices
- correlation matrix

Introduction - correlation and covariance matrices

From these matrices we can see that even though the two classes share some similarities, there are also some difference, for example class 1 shows a strong dependence of features 2,3 from features 4,5 while in the case of class 0, this doesn't happen at all. Furthermore, when we will deal with dimensionality reduction techniques we expect to be able to reduce the dimensionality only to a small extent without compromising too much the accuracy as the third matrix doesn't show an extremely large number of strongly correlated features.



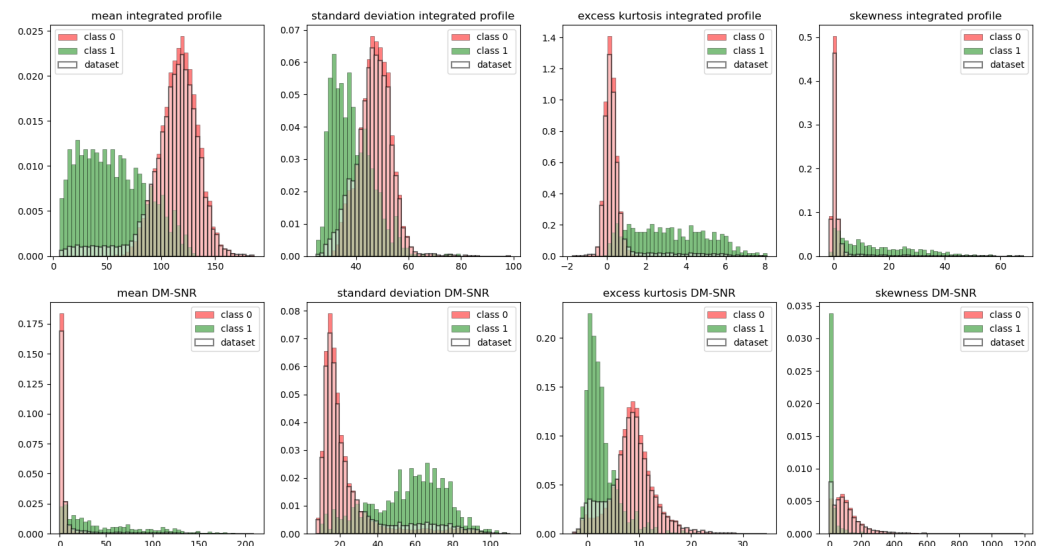
Introduction – feature distributions

Another interesting plot is the one on the bottom, which shows the range of each feature in each class.

In this case we can see that also in this case the two classes are quite different one to another as if each class has its own range of values for each feature, apart from a few exceptions (namely features 3, 4 and 7). This fact can lead us to think that the misclassification rate could be not very large.

Of course the entire dataset tends to follow the behaviour of class 0, being the most present in the datasets.

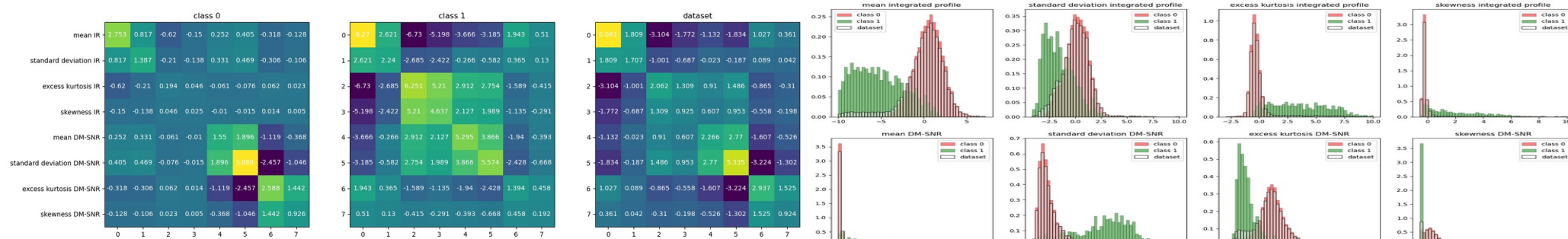
Furthermore all features follows some distribution which can be seen quite close with the Gaussian one, so we shouldn't have too bad results even with a standard Gaussian classifier



Intoduction – preprocessing and results

In order to make computations easier and to better compare the two classes, I've added two very simple preprocessing steps, namely features centering and rescaling. First of all I centered the dataset in order that the dataset as a whole has 0 as its mean. This is not terribly useful as in many cases we are interested in the means of each single class which is still different from 0 (even though class 0's mean is pretty close to 0).

The second operation was that of rescaling all features in such a way that their values fall between -10 and 10.





General informations about the classifiers – classification models used (1)

In this project I used 4 different types of classifiers and for each of them different values for their parameters were analyzed:

- multivariate Gaussian classifier
 - dimensionality reduction technique
 - covariance matrix types of the generated Gaussians
 - working points
- logistic regression
 - lambda factor
 - working points



General informations about the classifiers – classification models used (2)

- support vector machines
 - kernel type
 - values of C and K (with additional hyperparameters if a nonlinear kernel is used)
 - working points
- Gaussian mixture models
 - working points
 - maximum amount of total GMMs (as the sum of both classes)
 - covariance matrix types (like the multivariate Gaussian but in this case also with a fixed eigenvalue constraint)




General informations about the classifiers – validation set and working points

In order to try to make more accurate choices concerning the best model, a validation set has been introduced and used in each of the classification models presented before. It is implemented via a K-fold cross-validation approach, with K equal to five. This means that each time that we execute a training phase, we will divide the whole training set in 5 blocks and in the next 5 iterations one of them in turn will be designed as the validation set.

Moreover all classifiers will be trained and tested by taking into account three different working points:

- a balanced one, having prior 0.5, false negative cost of 1 and false positive cost of 1
- one favouring class 0 by having prior 0.092, negative cost of 1 and positive cost of 1
- a last one which on one hand tends to prefer class 1 (having prior 0.9), but on the other hand the model is punished very harshly for generating false positive predictions.




Training and validation phase – multivariate Gaussian (1)

Starting from standard Gaussian classifiers, I took the opportunity to check if dimensionality reduction techniques would be useful to improve the classification task.

The following reduction algorithm were used:

- linear discriminant analysis with reduced datasets having 2, 3 or 4 features
- principal component analysis with eigenvalue retain percentage threshold of 95%, 99% and 100% (no reduction but still change of basis)
- also the original dataset was kept (with no reduction) in order to be able to compare it to the reduced ones in the classification tasks



Training and validation phase – multivariate Gaussian (2)

Since at the moment we're dealing with Gaussians, another parameter that we can modify is the way in which we calculate the covariance matrices for the Gaussians that better approximate the two classes distributions.

In this scenario 4 different covariance matrices were tested:

- standard
- naive Bayes
- tied Bayes
- naive + tied Bayes

Training and validation phase – multivariate Gaussian (3)

Putting together all combinations of dimensionality reduction strategies, working point and covariance matrix types, the following tables shows the results of the most interesting combinations of the three. Each row represents the covariance type, each column represents the working point and the tuple inside each cell has the format (DCF, minimum DCF).

PCA 100% retain rate

	(0.9, 1, 10)	(0.5, 1, 1)	(0.092, 1, 1)
naive	(0.22, 0.192)	(0.213, 0.185)	(0.559, 0.476)
tied	(0.195, 0.137)	(0.194, 0.136)	(0.263, 0.214)
naive + tied	(0.192, 0.14)	(0.19, 0.136)	(0.292, 0.256)
full	(0.166, 0.144)	(0.161, 0.14)	(0.363, 0.312)

LDA 3D

	(0.9, 1, 10)	(0.5, 1, 1)	(0.092, 1, 1)
naive	(0.169, 0.154)	(0.167, 0.151)	(0.314, 0.279)
tied	(0.271, 0.163)	(0.267, 0.162)	(0.351, 0.239)
naive + tied	(0.358, 0.188)	(0.356, 0.188)	(0.418, 0.235)
full	(0.17, 0.148)	(0.166, 0.145)	(0.28, 0.254)

no reduction

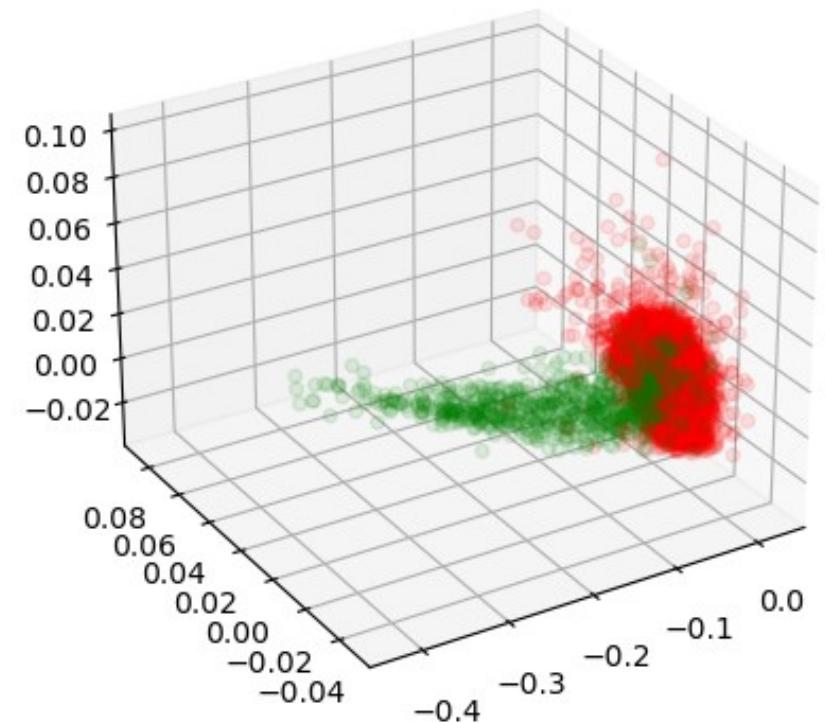
	(0.9, 1, 10)	(0.5, 1, 1)	(0.092, 1, 1)
naive	(0.199, 0.187)	(0.194, 0.181)	(0.623, 0.532)
tied	(0.195, 0.137)	(0.194, 0.136)	(0.263, 0.214)
naive + tied	(0.24, 0.2)	(0.238, 0.199)	(0.313, 0.261)
full	(0.166, 0.144)	(0.161, 0.14)	(0.363, 0.312)

Training and validation phase – multivariate Gaussian (3)

Even though the reduction using a 3D LDA is not perfect, it seems close enough to the complete dataset representation (based on the tables in the previous page) and so we could nevertheless plot the dataset in order to gain some more insight about the classifiers' future behaviour.

From what we are able to see from this plot, most of the points are very well separated, which confirms the predictions done during the initial dataset analysis. Since each class has its own unique pattern, we expect once again that also the following classifiers will obtain good classification scores.

plot of entire training set after applying 3D LDA





Training and validation phase – multivariate Gaussian (4)

To conclude, for the moment the best models are either one of these two multivariate Gaussian classifiers with the corresponding hyperparameters:

- working points: (0.5, 1, 1)
- covariance type: tied
- dimensionality reduction strategy: none
- or
- working points: (0.5, 1, 1)
- covariance type: tied or tied + naive
- dimensionality reduction strategy: no actual reduction, just modifying the reference axis through a pca with 100% variability retain rate

In the following I will use the first approach, i.e. there will be no modification to the dataset.



Training and validation phase – logistic regression (1)

To analyze the logistic regression classifier, I've tweaked only two parameters: the working point as usual and the value of lambda which is the regularization coefficient for this classifier.

Even though I tried to implement a kind of “kernelized” version of the logistic regression, I wasn't successful and so only the linear case is available in this case.


The working points are the same as before and the chosen lambdas are $1e-6$, $1e-5$ and $1e-4$.

Training and validation phase – logistic regression (2)

As in the case of the Gaussian classifier, also in this case the balanced working point leads to far better results than the one favoring class 0 and it is still slightly better than the remaining one.

I've also checked that setting the lambda in the range $[1e-6, 1e-4]$ gives the best rewards in terms of score, in fact for lower values the min-dcf doesn't change while with higher values the min-dcf increases drastically.

	(0.9, 1, 10)	(0.5, 1, 1)	(0.092, 1, 1)
lambda = 1e-6	(0.204, 0.117)	(0.201, 0.115)	(0.333, 0.210)
lambda = 1e-5	(0.204, 0.118)	(0.201, 0.116)	(0.332, 0.211)
lambda = 1e-4	(0.209, 0.117)	(0.205, 0.115)	(0.337, 0.208)




Training and validation phase – support vector machines

	(0.9, 1, 10)	(0.5, 1, 1)	(0.092, 1, 1)
C = 0.1, K = 1	(0.234, 0.129)	(0.23, 0.156)	(0.437, 0.256)
C = 0.1, K = 10	(0.244, 0.201)	(0.242, 0.198)	(0.612, 0.297)

In the case of linear SVM we can notice how a smaller K tends to give better results.

The versions of SVM including both Gaussian kernel and the polynomial one are present in the code but I wasn't able to track the accuracy of these solutions.




Training and validation phase – Gaussian mixture models (1)

When it comes to building the Gaussian mixture models, there are many similarities with the multivariate Gaussian with the additions of some other peculiarities.

In addition to the 3 covariance types described earlier, I've also added in this model the constrained covariance matrix which acts a sort of lower bound for the computed eigenvalues.

Unfortunately due to the high number of permutations of the already existing hyperparameters, I had to fix some of them and the choice was to set alpha (a factor used in the LBG algorithm) and psi (the lower bound of the covariances' eigenvalues) both to 0.1 and the stopping criterion of the EM algorithm to $1e-2$, as the minimum difference between the last iteration's log-likelihood ratio and the current one as a requirement to proceed with the algorithm.

In order to duplicate the number of GMMs of each class during an arbitrary LBG iteration, I followed the classic strategy of generating two GMMs “g1” and “g2” from the “parent” GMM “g” in such a way that they will both have the multiplier/coefficient equal to $0.5 \cdot \text{coefficient}(\text{“g”})$, the same covariance matrix of “g” and a mean of $\text{mean}(\text{“g”}) - d/2$ and $\text{mean}(\text{“g”}) + d/2$ respectively with d being a value derived from the eigenvalues.



Training and validation phase – Gaussian mixture models (2)

The search for a good model was executed by analyzing the results given the permutations of these hyperparameters:

- working points (as usual)
- maximum amount of Gaussians in both GMMs, in this case referring to the total amount of Gaussians (in such a way that at the end each class will have maximum/2 Gaussians in its GMM), in this case it was set to 8
- covariance matrix types, similar to the multivariate ones but now we have a lower bound on the values of the eigenvalues (“this type of covariance matrix was labelled as ‘constrained’”)

Training and validation phase – Gaussian mixture models (3)

Even though the results seem promising, with 4 Gaussians per class the GMM model is not able to reach the performance of the logistic regression model.

This is probably due to the low limit of the total number of Gaussian as with 16 total Gaussian a GMM with a constrained covariance matrix and one of the first two working points should be able to easily outperform all other models.

working point = (0.5, 1, 1)

	constrained	constrained, naive	constrained, tied	constrained, naive, tied
2	(0.163, 0.143)	(0.2, 0.182)	(0.163, 0.142)	(0.2, 0.182)
4	(0.198, 0.178)	(0.206, 0.183)	(0.163, 0.142)	(0.18, 0.165)
8	(0.137, 0.127)	(0.18, 0.154)	(0.163, 0.142)	(0.17, 0.157)

working point = (0.9, 1, 10)

	constrained	constrained, naive	constrained, tied	constrained, naive, tied
2	(0.166, 0.147)	(0.207, 0.188)	(0.166, 0.147)	(0.207, 0.188)
4	(0.2, 0.182)	(0.208, 0.187)	(0.166, 0.147)	(0.185, 0.169)
8	(0.138, 0.129)	(0.184, 0.16)	(0.166, 0.147)	(0.173, 0.161)

working point = (0.092, 1, 1)

	constrained	constrained, naive	constrained, tied	constrained, naive, tied
2	(0.36, 0.299)	(0.591, 0.53)	(0.36, 0.3)	(0.591, 0.53)
4	(0.281, 0.263)	(0.334, 0.292)	(0.36, 0.3)	(0.494, 0.373)
8	(0.255, 0.212)	(0.361, 0.296)	(0.36, 0.3)	(0.406, 0.326)



Testing phase

Now that we have obtained the best model for each classification algorithm thanks to the K-fold cross validation, we need to train the models on the entire training set and finally check the final result on the test set.

Even though we know which classification models are better than others due to the scores obtained in the validation set, I'd like to check the performance of all of them with respect to the test set.

This can help to find out if the test set and the training one are effectively balanced.



Testing phase and conclusion

From the results obtained from applying the models on the test set we can notice that the values of the dfcs are not very different from the ones obtained in the validation set but we can still say that the test set wasn't completely balanced with the training one (maybe the test set had more samples from class 1).

Another thing to notice is that all proposed models are not very well calibrated since the difference between dfcs and relative minimum dfcs are quite high and in this case there would be the need for some recalibration.

	dfc	minimum dfc
multivariate Gaussian	0.154	0.141
logistic regression	0.2	0.11
support vector machines	0.191	0.136
Gaussian mixture models	0.139	0.127