Classification of the HTRU2 dataset with Machine Learning models

Fabio Grillo

Politecnico di Torino
s287873@studenti.polito.it

Abstract

Through this paper we aim to try to analyze the HTRU2 dataset in order to better understand its peculiarities, through machine learning algorithms. The initial part will focus on analyzing the features of the dataset, trying to understand what they are and how they are distributed; then the performance of different models will be evaluated. The results will be presented in the final conclusions.

I. Introduction

The HTRU2 data set describes a sample of pulsar candidates collected during the High Time Resolution Universe Survey.
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.
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. 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. Machine learning tools are now being used to automatically label pulsar candidates to facilitate rapid analysis.
Classification systems in particular are being widely adopted, which treat the candidate data sets as binary classification problems. Here the legitimate pulsar examples are a minority positive class, and spurious examples the majority negative class.

II. Feature Analysis

In this section we will try to analyze the characteristics of the data set and their distributions. First of all, each candidate is described by 8 continuous variables, in order:

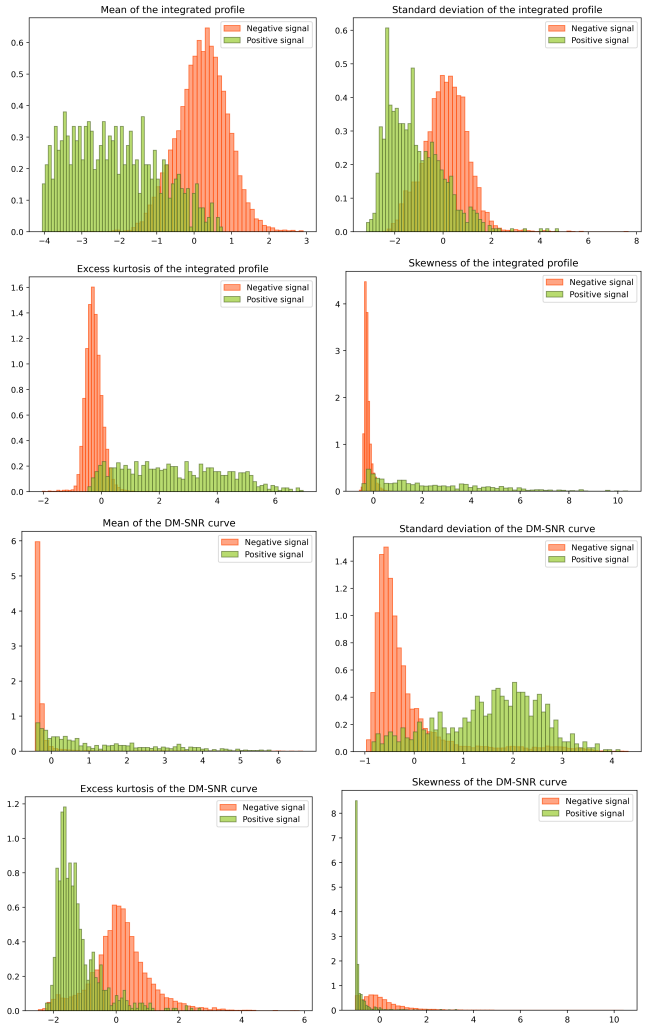
1. Mean of the integrated profile
2. Standard deviation of the integrated profile
3. Excess kurtosis of the integrated profile.
4. Skewness of the integrated profile.
5. Mean of the DM-SNR curve.
6. Standard deviation of the DM-SNR curve.
7. Excess kurtosis of the DM-SNR curve.
8. Skewness of the DM-SNR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

The data set consists of 16.259 spurious examples as well as negative pulsar signals caused by RFI/noise (*Class0*) and 1.639 real pulsar examples (*Class1*); with 17.898 total examples.
In order to achieve simplification in the operations, normalization was applied, transforming each candidate x_i with the formula:

$$x_i = \frac{x_i - \mu}{\sigma}$$

where μ and σ are the mean and standard deviation of each feature, respectively.
Training was performed on a first split of the dataset consisting of 8929 total samples, while another split, consisting of 8969 samples, was successively used in the validation phase.
Plots of the raw features (just Z-normalization applied) are shown below.

Figure 1: *Raw features distribution of HTRU2 data set*

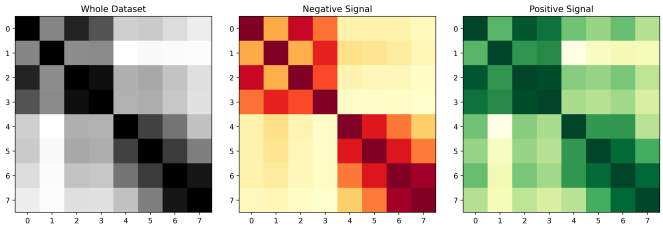


The absence of outliers can be easily observed from the graphs in 1, thus highlighting the fact that preprocessing techniques, such as Gaussanization, will not be necessary for manipulating the dataset.
A correlation analysis was pursued next to highlight whether there were strong correlations between the features. To do this, Pearson's correlation coefficient, given by the formula

$$corr = \left| \frac{Cov(X,Y)}{\sqrt{Var(X)}\sqrt{Var(Y)}} \right|$$

was used.
This provided an understanding of how PCA could be applied to the dataset and how it could be benefited, to reduce the number of parameters to be estimated. Next, the resulting heatmaps are shown.

Figure 2: *Heatmaps of HTRU2 features with Z-Normalization*



From what can be seen, darker colors indicate a Pearson coefficient value close to 1, thus a stronger correlation between features (certainly the color will be the darkest on the diagonal). From 2 it can be seen that the features most strongly correlated are 0-2, 2-3 and 6-7.
Therefore, the PCA dimensionality reduction technique was necessary to bring the dimension from 8 to a lower degree, observing whether or not performance remains stable.

III. Classifiers

In order to evaluate which model performs better and to better understand the effects of using PCA, two diffferent methodologies were adopted:

- Single split: consists in splitting the training set into two subsets, one useful for training the other for validation, in propoportion 66% and 33%, respectively; here the final classifier corresponds to the same one evaluated on the validation set; moreover, the training is faster

- K-Fold Cross Validation: the final clasifier is obtained by re-training on the whole training set and decisions are made on the basis of the validation set; it consists of splitting the training set K times into K subsets, each time K - 1 subsets are used for training and 1 for validation

In addition, PCA was tested with different values of m, from 7 down to 5, below which performance was expected to drop dramatically. Again, dimensionality reduction was applied by using only the training set.
Both approaches were considered with the expectation that better and robust results would come from the K-Fold given the high degree of imbalance in the dataset.
The main application was a uniform prior one

$$(\tilde{\pi}, C_{fp}, C_{fn}) = (0.5, 1, 1)$$

but also unbalanced ones were considered

$$(\tilde{\pi}, C_{fp}, C_{fn}) = (0.1, 1, 1)$$
$$(\tilde{\pi}, C_{fp}, C_{fn}) = (0.9, 1, 1)$$

where $\tilde{\pi}$ is the effective prior.

Each model was evaluated by considering the minimum detection cost, which represents the cost one would have to pay if optimal decisions were made using the optimal threshold for the validation subset.

III.I. Gaussian Classifiers

We began by considering a Gaussian classifier such as MVG (Full Covariance) and Naive Bayes (Diagonal covariance). The worst results were expected from the Naive Bayes classifier given the assumption that its features are independently distributed; under this assumption, precisely, the features should have been poorly correlated with zero covariance: therefore, poor results were expected from this classifier. Instead, the Tied version of the MVG should have provided better results as it was more effective in capturing correlations between features.

	Single Split			5-Folds		
$\tilde{\pi}$	0.5	0.1	0.9	0.5	0.1	0.9
No PCA						
Full-Cov	0.155	0.244	0.739	0.141	0.286	0.672
Diag-Cov	0.210	0.424	0.722	0.193	0.315	0.747
Tied Full-Cov	0.155	0.247	0.498	0.112	0.224	0.573
Tied Diag-Cov	0.171	0.288	0.627	0.161	0.267	0.579
PCA m = 7						
Full-Cov	0.135	0.271	0.697	0.139	0.304	0.641
Diag-Cov	0.208	0.474	0.655	0.214	0.506	0.724
Tied Full-Cov	0.105	0.234	0.477	0.112	0.223	0.572
Tied Diag-Cov	0.135	0.267	0.544	0.138	0.271	0.601
PCA m = 6						
Full-Cov	0.156	0.261	0.663	0.152	0.289	0.650
Diag-Cov	0.217	0.477	0.673	0.223	0.526	0.721
Tied Full-Cov	0.148	0.253	0.509	0.140	0.259	0.580
Tied Diag-Cov	0.167	0.276	0.577	0.164	0.298	0.589
PCA m = 5						
Full-Cov	0.155	0.244	0.739	0.150	0.250	0.642
Diag-Cov	0.210	0.424	0.722	0.220	0.454	0.733
Tied Full-Cov	0.155	0.247	0.498	0.150	0.262	0.574
Tied Diag-Cov	0.171	0.288	0.627	0.171	0.311	0.610

Table 1: *Gaussian Classifiers*

As can be seen in Table 1, our previous assumptions were correct. The values obtained assume the same behaviors for both single split and k-fold; as far as PCA is concerned, we note that for values of m equal to 6 and 5 the performance worsens, thus inducing us to use a value equal to 7 with which the best results were obtained; lastly, we observe that Tied Full-Cov outperforms models with quadratic separation rule: we therefore infer that linear classifiers perform better than quadratic ones, and we also observe that significantly better results are given for balanced applications.

III.II. Logistic Regression

Next, discriminative approaches were used: linear regularized logistic regression. Given the good results obtained from Gaussian tied-covariance models, the expectation was that other linear classifiers would work well. The objective function used is:

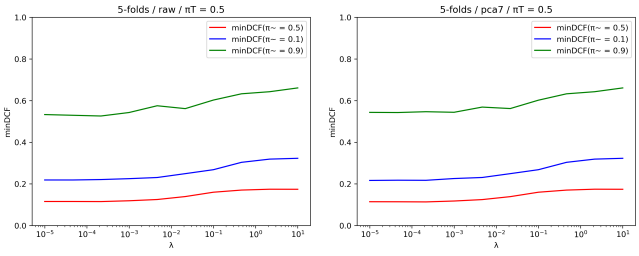
$$J(w,b) = \frac{\lambda}{2}||w||^2 + \frac{\pi_T}{n_T}\sum_{i=1|c_i=1}^{n} log(1+e^{-z_i(w^T x_i+b)}) + \frac{1-\pi_T}{n_F}\sum_{i=1|c_i=0} log(1+e^{-z_i(w^T x_i+b)})$$

λ is a parameter, called the regularization coefficient, which can be understood in the following ways:

- $\lambda >> 0$: poor separation of classes and a small $\|\omega\|$
- $\lambda \simeq 0$: good separation of classes, poor generalization on unseen data

The choice of lambda is given by the following graphs.

Figure 3: *minDCF plots of Logistic Regression for different priors*



It is evident in 3 that the differences are minimal between the "raw" plot and the plot with PCA m = 7; in addition $\lambda = 10^{-5}$ was a good value that we decided to select. Below are the results with different priors and $\pi_T = 0.5$.

	5-Folds		
$\tilde{\pi}$	0.5	0.1	0.9
	No PCA		
LogReg($\lambda = 10^{-5}, \pi_T = 0.5$)	0.126	0.238	0.571
LogReg($\lambda = 10^{-5}, \pi_T = 0.1$)	0.126	0.243	0.588
LogReg($\lambda = 10^{-5}, \pi_T = 0.9$)	0.134	0.241	0.520
	PCA m = 7		
LogReg($\lambda = 10^{-5}, \pi_T = 0.5$)	0.114	0.217	0.543
LogReg($\lambda = 10^{-5}, \pi_T = 0.1$)	0.112	0.212	0.561
LogReg($\lambda = 10^{-5}, \pi_T = 0.9$)	0.117	0.217	0.524

Table 2: *Logistic Regression Classifiers*

In general Logistic Regression performs well from as previously inferred, where linear models perform better than quadratic models. As can be seen from Table 2, PCA $m = 7$ performs slightly better than the raw model, which is why we continued to consider both approaches. Also better performance is achieved for values of π_T = 0.1, this is due to the unbalance toward Class0 samples.

III.III. Support Vector Machines

In this section, performance was tested by allowing the separation of samples up to a margin, strategy called Support Vector Machine. For linear SVM we needed to tune the hyperparameter C through cross-validation.
For this purpose, the dual problem was considered through the formula:

$$J^D(\alpha) = -\frac{1}{2}\alpha^T H\alpha + \alpha^T 1$$

In addition, balancing was done by considering different values of C for each class.

$$where\ C_i = \begin{cases} C\frac{\pi_T}{\pi_F^{emp}} if i \in Class_1 \\ C\frac{\pi_F}{\pi_F^{emp}} if i \in Class_0 \end{cases}$$

From here, the following plots emerged:

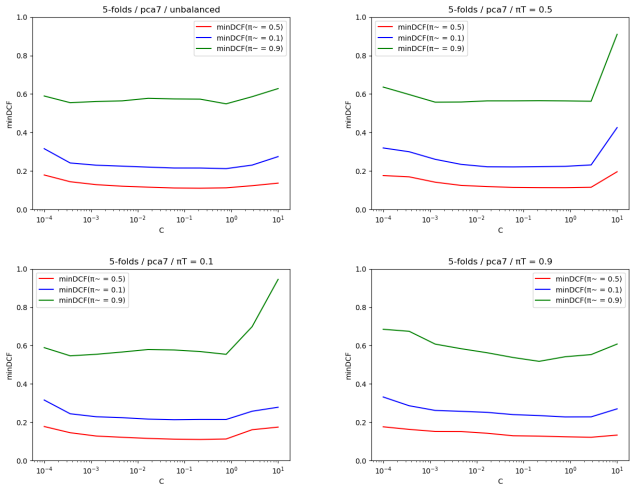


Figure 4: *minDCF plots of linear SVM*

Both the unbalanced and balanced models are roughly similar in terms of performance, so $C = 10^-2$ was chosen. After choosing that value for C it was possible to observe that PCA m = 7 had not worsened the results, so rebalancing was not decisive, allowing only the unbalanced application to be considered. It can be seen from the table below that $\pi_T = 0.1$ provided slightly better results.

	5-Folds		
$\tilde{\pi}$	0.5	0.1	0.9
	No PCA		
Linear SVM($C = 10^{-2}, unbalanced$)	0.118	0.224	0.575
Linear SVM($C = 10^{-2}, \pi_T = 0.5$)	0.121	0.223	0.564
Linear SVM($C = 10^{-2}, \pi_T = 0.1$)	0.118	0.222	0.573
Linear SVM($C = 10^{-2}, \pi_T = 0.9$)	0.145	0.253	0.557
	PCA m = 7		
Linear SVM($C = 10^{-2}, unbalanced$)	0.118	0.224	0.575
Linear SVM($C = 10^{-2}, \pi_T = 0.5$)	0.121	0.223	0.564
Linear SVM($C = 10^{-2}, \pi_T = 0.1$)	0.118	0.222	0.573
Linear SVM($C = 10^{-2}, \pi_T = 0.9$)	0.145	0.253	0.557

Table 3: *Linear Support Vector Machines*

III.IV. Quatric SVMs

At this point, through the use of kernel function k, it was possible to compute a linear separating surface corresponding to a non-linear separating surface in the original feature space.

$$k(x_i, x_j) = \Phi(x_i)^T \Phi(x_j{}^T)$$

In any case, poor results were expected given how well the linear classifiers fit the dataset. Two different types of kernel functions were used:

- **Radial Basis Function (RBF):** $e^{-\gamma\|x_i - x_j\|^2}$
- **Polynomial:** $(x_i{}^T x_j + c)^d$

From the minDCF graphs below, the value of C, γ, and c were chosen.

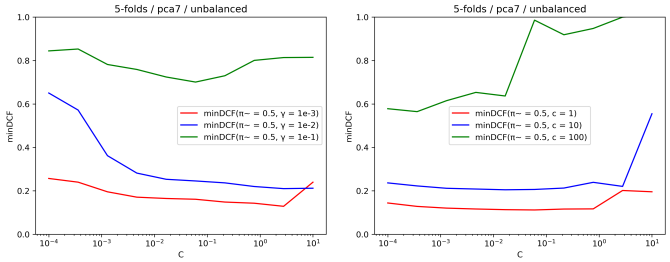


Figure 5: *minDCF plots of quadratic SVM. Left: RBF. Right: Polynomial*

$\tilde{\pi}$	5-Folds 0.5	0.1	0.9
	No PCA		
RBF SVM($C = 10^-1, \gamma = 10^-3$)	0.160	0.265	0.568
Poly SVM($C = 10^-3, c = 1, d = 2$)	0.122	0.227	0.707
	PCA m = 7		
RBF SVM($C = 10^-1, \gamma = 10^-3$)	0.160	0.265	0.568
Poly SVM($C = 10^-3, c = 1, d = 2$)	0.122	0.227	0.707

Table 4: *Quadratic SVMs*

As expected, the performance of the quadratic classifiers was worse than that of the linear classifiers, having provided poor results.

III.V. Gaussian Mixture Model

We now treat Gaussian Mixture Models. A GMM can approximate generic distributions, so it was expected to perform as well as MVGs, or even better. GMMs operate under the assumption that each sample is generated from a mixture of a finite number of Gaussian distributions with unknown parmeters. The number of these distributions is in fact a hyperparameter. Graphs of different types of GMM (Full Covariance, Diagonal Covariance and Tied Covariance), so as to estimate the correct number of components, can be found next.

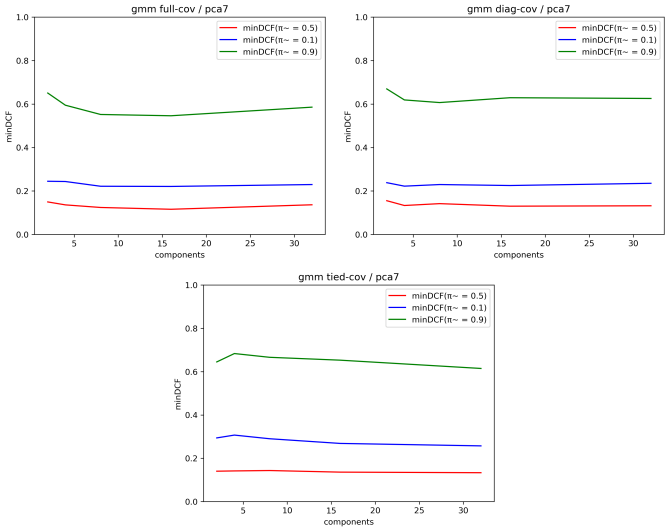


Figure 6: *minDCF plots of GMM. In order: Full Covariance, Diagonal Covariance, Tied Covariance*

$\tilde{\pi}$	5-Folds 0.5	0.1	0.9
	No PCA		
GMM Full Cov (8 Components)	0.124	0.222	o.552
GMM Diagonal Cov (16 Components)	0.130	0.225	0.629
GMM Tied Covariance (32 Components)	0.133	0.257	0.615
	PCA m = 7		
GMM Full Cov (8 Components)	0.124	0.222	o.552
GMM Diagonal Cov (16 Components)	0.130	0.225	0.629
GMM Tied Covariance (32 Components)	0.133	0.257	0.615

Table 5: *Gaussian Mixture Models*

Finally, from the data provided by Table 5, it was possible to consider GMM Full Cov with 8 components as one of the best models along with Tied Full-Cov, Logistic Regression and Linear SVM

IV. Scores Calibration

The best models in terms of performance are:

- MVG Tied Covariance
- LogReg ($\lambda = 10^-5, \pi_T = 0.5$)
- Linear SVM (C = 10^-2, unbalanced)
- GMM Full Covariance (8 Components)

So far the minDCF has been considered, i.e. the empirical Bayes cost if we use the scores provided by our recognizers in making optimal decisions. However, in the binary case the optimal decision is to choose a good threshold to compare with log-likelihoods ratios or to recalibrate the scores so that the optimal threshold becomes the theoretical one:

$$t = -log\frac{\tilde{\pi}}{1-\tilde{\pi}}$$

The actual cost we would pay depends on the goodness of the threshold, and that is why the actualDCF is introduced, which is different from minDCF because the classification is carried out using the threshold corresponding to $\tilde{\pi}$. The results can be seen from the graphs below.

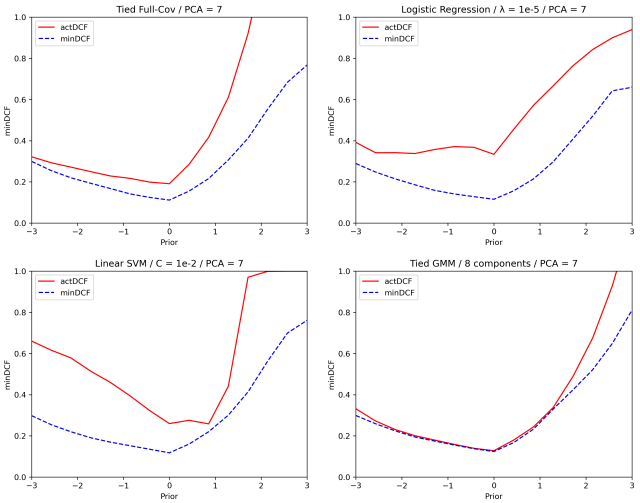


Figure 7: *minDCF and actDCF for uncalibrated classifiers*

It was possible to see from 7 that there is a large difference between minDCF and actualDCF, and for that reason, it was possible to employ a score calibration technique consisting of computing a function that maps uncalibrated scores s to calibrated scores s_c. It is possible to write it as:

$$f(s) = \alpha s + \beta$$

f(s) can be interpreted as the log-likelihood ratio for the two classes, writing the class posterior for a prior $\tilde{\pi}$ as:

$$log(\frac{P(C = Class_1|s}{p(C = Class_0|s}) = \alpha s + \beta + log\frac{\tilde{\pi}}{1-\tilde{\pi}}$$

Then K-Fold and Logistic Regression were applied to the scores, allowing for a better calibration for all applications. Plots for $\tilde{\pi} = 0.5$ are shown below.

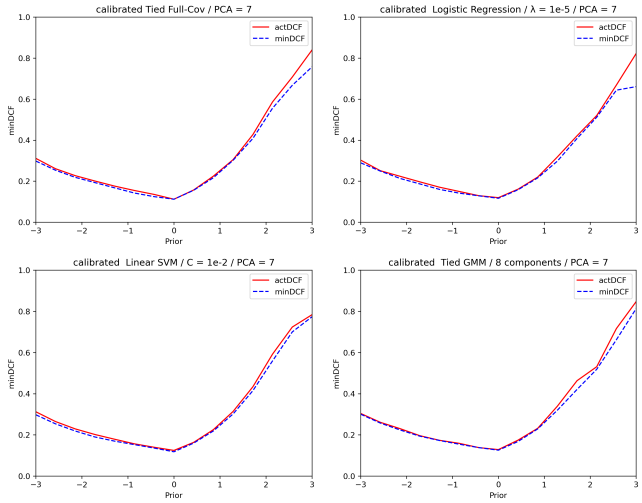


Figure 8: *minDCF and actDCF for calibrated classifiers*

	minDCF			actDCF		
$\tilde{\pi}$	0.5	0.1	0.9	0.5	0.1	0.9
No PCA						
Tied Full-Cov	0.112	0.223	0.573	0.112	0.228	0.596
LogReg(10^-5, 0.5)	0.116	0.218	0.526	0.120	0.225	0.547
SVM(C = 10^-2)	0.118	0.223	0.582	0.125	0.232	0.593
GMM Full Cov (8)	0.126	0.227	0.534	0.128	0.231	0.550
PCA m = 7						
Tied Full-Cov	0.112	0.223	0.570	0.113	0.228	0.606
LogReg(10^-5, 0.5)	0.114	0.217	0.541	0.117	0.225	0.548
SVM(C = 10^-2)	0.118	0.224	0.579	0.124	0.232	0.593
GMM Full Cov (8)	0.124	0.221	0.567	0.128	0.229	0.606

Table 6: *Selected Models with score calibration*

Now miscalibrated scores have been mapped into well-calibrated scores.

V. Evaluation & Conclusions

Proceeding with the analysis, considering that PCA m = 7 worked well as a preprocessing method, we continued considering this strategy.

	minDCF			actDCF		
$\tilde{\pi}$	0.5	0.1	0.9	0.5	0.1	0.9
PCA m = 7						
Tied Full-Cov	0.110	0.208	0.587	0.115	0.221	0.606
LogReg(10^-5, 0.5)	0.106	0.201	0.541	0.113	0.221	0.579
SVM(C = 10^-2)	0.116	0.213	0.551	0.121	0.226	0.582
GMM Full Cov (8)	0.115	0.226	0.527	0.117	0.227	0.542

Table 7: *Selected Classifiers with score calibration on Evaluation Set*

It can be seen from Table 7 that the results remained consistent with those obtained from the training set. In our case the best performing is Logistic Regression.
Using ROC (Receiver Operating Characteristic) curve to compare the models, the best ones have the highest Area Under Curve.

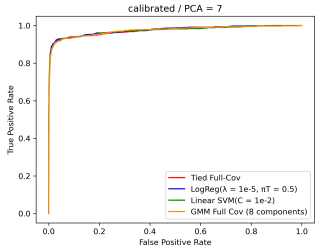


Figure 9: *ROC for selected classifiers*

Here the curve is almost the same for all classifiers. The initial slope represents the classifiers' ability to correctly classify $Class_1$ (True Pulsar) while keeping $Class_0$ (False Pulsar) at minimum.
So, the test data provided decidedly similar results to the training data highlighting how the linear models as well as the GMMs were fit for the dataset.
In conclusion, for the HTRU2 dataset the perfromances evaluated on the test set provided evidence that our previous assumptions and subsequent choices were correct.

VI. References

- 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 MNRAS, 2016.

- 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

- R. J. Lyon, "PulsarFeatureLab", 2015, https://dx.doi.org/10.6084/m9.figshare.1536472.v1.