# HTRU2 dataset classification using Machine Learning models

Edoardo Alberti
s305922@studenti.polito.it

July 13, 2022

**Abstract**

*The goal of this paper is to analyze the HTRU2 dataset through the use of machine learning algorithms and to highlight how these behave on a highly unbalanced dataset. We will start by analyzing relationships between features and how those are distributed and then concentrate on evaluating the performance of the various models we are going to use. We will find that linear models, in general, will outperforms the quadratic ones, and that a correct score calibration will lead to even more precise results.*

## 1 Introduction to the Dataset

HTRU2 is a dataset which 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.

## 2 HTRU2 features

Each candidate in the HTRU2 dataset is described by 8 continuous variables:

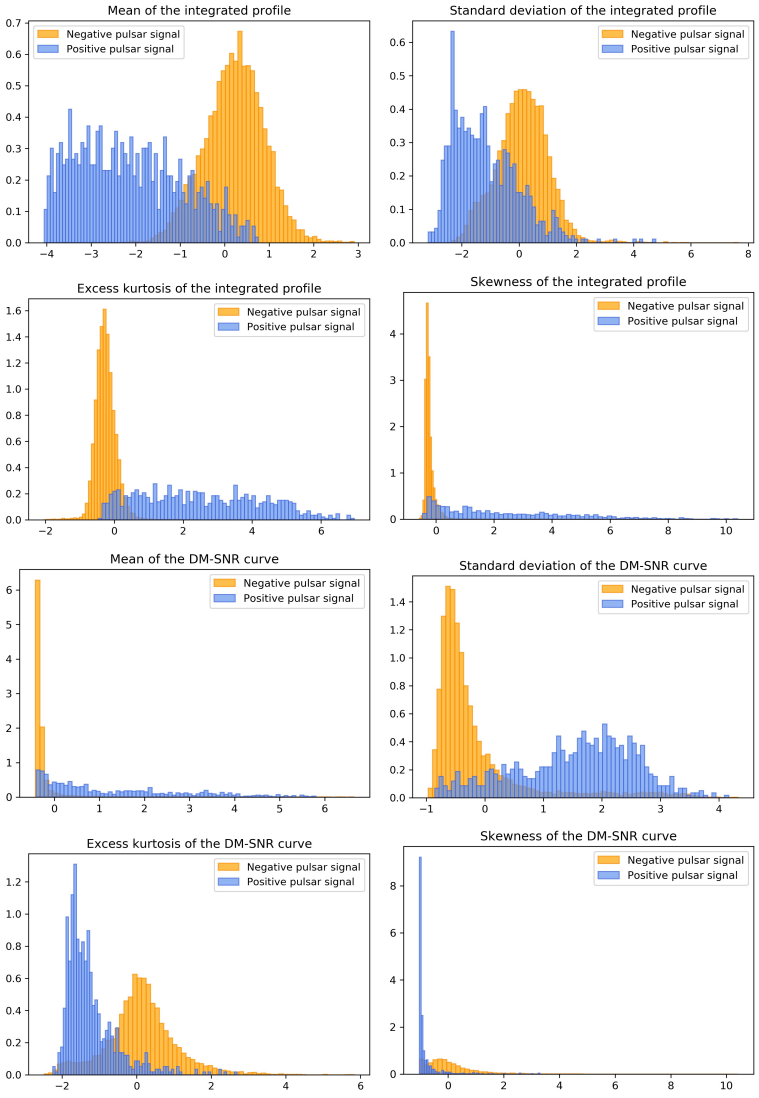
1. Mean of the integrated profile.
2. Standard deviation of the integrated profile.
3. Excess kurtosis of the integrated profile.
4. Skewness of the integrated profile.
5. Mean of the DM-SNR curve.
6. Standard deviation of the DM-SNR curve.
7. Excess kurtosis of the DM-SNR curve.
8. Skewness of the DM-SNR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

The first four are simple statistics obtained from the integrated pulse profile (folded profile), the remaining four variables are obtained from the DM-SNR curve.
The dataset consists of 17,898 total samples: 1,639 are positive pulsar signals ($Class_1$) and 16,259 are negative pulsar signals ($Class_0$).
Before performing any operation the data has been preprocessed by means of Z-Normalization (centering and scaling to unit variance) in order to simplify computations and avoid overflows. Each candidate $\boldsymbol{x}_i$ has been transformed through the formula:

$$\boldsymbol{x}_i = \frac{\boldsymbol{x}_i - \boldsymbol{\mu}}{\boldsymbol{\sigma}}$$

Where $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are column vector of size 8 containing the mean and the standard deviation of each feature.
The dataset has been splitted in two, one split is used for Training (8929 samples in total) and one split for Testing (composed by 8969 samples). During our analysis on the Machine Learning models we will consider only the Training set (using methods like K-Folds and Single Split). The Test set will be used at the end in order to validate the obtained results.
Below features preprocessed with Z-Normalization have been plotted.

Figure 1: *Plots of HTRU2 features with Z-Normalization*

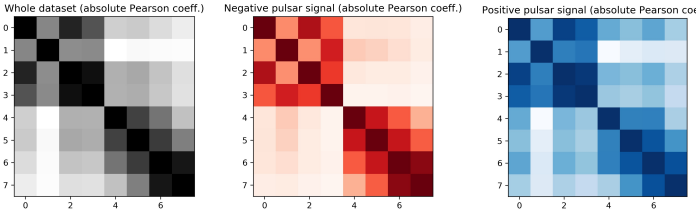



From the plots shown in Figure 1, we can observe that features seems already well distributed with no evident outliers. Therefore we wont apply other preprocessing techniques, such as Gaussianization.
We can now proceed by analyzing the correlation between features by plotting heatmaps that show correlations via the absolute value of the Pearson correlation coefficient:

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

This allows us to see which and how much features are correlated, giving an insight on the data and an hint on how PCA could be applied to the dataset.
Below are the three heatmaps representing the features analyzed on the entire dataset, on the positive samples set and on the negative samples set.
Darker color indicates an absolute value of the Pearson coefficient closer to 1.

Figure 2: *Heatmaps of HTRU2 features with Z-Normalization*
*Whole Dataset / Negative Pulsar Signal / Positive Pulsar Signal*




From the heatmaps in Figure 2 we can observe that, among all, the strongest correlations are between features 3-1, 4-3 and 8-7. Dimensionality reduction techniques, such as PCA, may come in handy for mapping samples from an $8^{th}$ dimensional space to a lower dimensional one. In the following we will use PCA up to $m = 5$ to see how models behave, but we expect that going that low in dimension may cause some major losses in terms of informations.

# 3 HTRU2 classification

To carry out our analysis on which model will perform better and to assess the effects of using PCA, we can adopt two methodologies:

- **Single Split**: Split the training set into two subset, one used for training and one for validation (we use a split that consists of 66% training and 33% validation).
  The final classifier will be the same one evaluated on the validation set.

- **K-Fold cross-validation**: Split the training set $K$ times into $K$ distinct sets. Each time $K-1$ sets are used for training and 1 for validation. The final classifier will be obtained by retraining the model over the whole training set.

For the moment we consider both approaches but we expect more robust results from K-Fold given the high imbalance of the HTRU2 dataset.
Our main application will be a uniform prior one but we also consider unbalanced applications where the prior is biased towards one of the two applications:

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

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

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

Models' quality will be measured in terms of *Minimum Detection Costs*, which correspond to the cost we would pay if we made optimal decision using the optimal threshold for the validation subset, extracted from the training set using the techniques mentioned above.
Class labels will be encoded as follows:

$$z_i = \begin{cases} +1 & \text{if } x_i \in Class_1 \\ -1 & \text{if } x_i \in Class_0 \end{cases}$$

### 3.1 Gaussian Classifiers

We can now start our analysis on Machine Learning models by considering a Gaussian classifiers such as MVG (Full covariance) and Naive Bayes (Diagonal covariance), alongside with their Tied assumption (each class has its own mean $\boldsymbol{\mu}_c$ but the covariance matrix $\boldsymbol{\Sigma}$ is computed over the whole dataset).
Gaussian Classifiers works under the assumption that data follows a gaussian distribution:

$$(\boldsymbol{X}_i|\boldsymbol{C}_i = c, \boldsymbol{\Theta}) \sim (\boldsymbol{X}|\boldsymbol{C} = c, \boldsymbol{\Theta}) \sim N(\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)$$

Worst results are expected from the Naive Bayes version of the classifier given its assumption on features being independently distributed. In fact, under this hypothesis, features should be poorly correlated with zero covariance, leading to the off-diagonal elements of the covariance matrix being zero. Since some of our features are strongly correlated we expect non-optimal results from this classifier.
Instead the Tied version of the MVG, should provide better results given its ability to capture correlations and the large number of samples at our disposal. Below the results obtained.

Table 1: *Gaussian Classifiers with Z-Normalization*

| | 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 | 0.575 |
| Tied Diag-Cov | 0.171 | 0.288 | 0.627 | 0.161 | 0.264 | 0.573 |
| | | | 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 | 0.113 | 0.224 | 0.570 |
| Tied Diag-Cov | 0.135 | 0.267 | 0.544 | 0.138 | 0.268 | 0.598 |
| | | | 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 | 0.571 |
| Tied Diag-Cov | 0.167 | 0.276 | 0.577 | 0.163 | 0.297 | 0.601 |
| | | | 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 | 0.149 | 0.263 | 0.571 |
| Tied Diag-Cov | 0.171 | 0.288 | 0.627 | 0.169 | 0.311 | 0.610 |

From Table 1 we can derive that our assumptions on Naive Bayes and Tied MVG where correct. Values of the minDCF are consistent between both 5-Folds and Single Split, despite this result we continue our analysis employing only the K-Fold approach, from which we expect more robust results. By applying PCA with $m = 7$ results are consistent with the ones obtained on Z-Normalized only data, however starting from PCA with $m = 6$ performance begin to deteriorate, for this reason we drop both $m = 6$ and $m = 5$. Finally we observe some other interesting results such as the fact that a linear separation rule, Tied Full-Cov, outperforms the quadratic one provided by the Full-Cov model, this leads us to think that linear classifiers could behave better than quadratic ones. We will see if this hypothesis holds.

### 3.2 Logistic Regression

We shift now our attention to Logistic Regression models.
Since classes are unbalanced we employ a regularized version of the objective function:

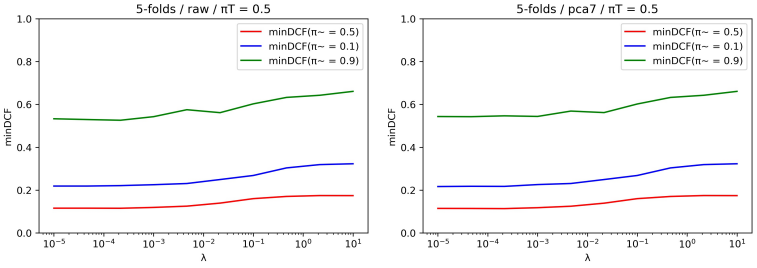
$$J(\boldsymbol{\omega}, b) = \frac{\lambda}{2}||\boldsymbol{\omega}||^2 + \frac{\pi_T}{n_T}\sum_{i=1|_{c_i=1}}^{n} \log(1 + e^{-z_i s_i}) + \frac{1-\pi_T}{n_F}\sum_{i=1|_{c_i=0}}^{n} \log(1 + e^{-z_i s_i})$$

with $s_i = (\boldsymbol{\omega}^T \boldsymbol{x}_i + b)$, where $(\boldsymbol{\omega}, b)$ are the model parameters, $\frac{\lambda}{2}||\boldsymbol{\omega}||^2$ a regularization term that helps obtaining a $\boldsymbol{\omega}$ with lower norm (reducing the risk of over-fitting the training data) and $\lambda$, which is an hyperparameter called regularization coefficient.

- $\boldsymbol{\lambda \gg 0}$: poor separation of classes and a small $||\boldsymbol{\omega}||$.
- $\boldsymbol{\lambda \simeq 0}$: good separation of classes but poor generalization on unseen data.

Let's proceed with the choice of $\lambda$ by plotting minDCF graphs for a range of values.

Figure 3: *minDCF plots of Logistic Regression with $\pi_T = 0.5$ for different priors $\tilde{\pi}$ on Z-Normalized data.*




We can see from the plots that a good value of $\lambda$ could be $10^{-5}$ and that results do not differ much between No PCA and PCA $m = 7$. Below the table containing the results for differently balanced models with $\lambda = 10^{-5}$.

Table 2: *Logistic Regression Classifiers with Z-Normalization*

| | 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 |

Logistic Regression overall performs well, this is a result that we hypothesized during the analysis of Gaussian classifiers, where linear models performed better than quadratic ones this strengthens our hypothesis on linear decision rules being the best candidates for this specific dataset. We will finalize this assumption later, when we will discuss about quadratic SVMs. PCA $m = 7$ in this case performs better than No PCA, however we will keep considering both approaches. We can observe that classifier with $\pi_T = 0.1$ performs better than the others for the main application $\tilde{\pi} = 0.5$, this is due to the imbalance towards samples of $Class_0$ of the HTRU2 dataset.

### 3.3 Support Vector Machines

The risk minimization problem seen in Logistic Regression can be cast to a more general problem, allowing the separation of the samples up to a margin, this strategy is called Support Vector Machine. To solve the SVM problem we can consider the dual formulation, which is easier to optimize (its complexity depends only on the number of samples), and it allows us to compute non-linear hyperplanes without the need to explicitly expand the features:

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

$$with\ 0 \leq \alpha_i \leq C, \forall i \in \{1, ..., n\}\ and\ \sum_{i=1}^{n} \alpha_i z_i = 0$$

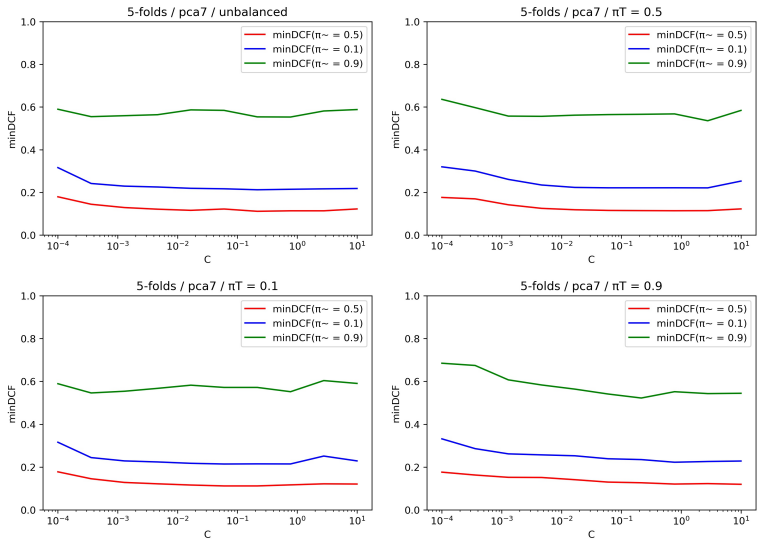
We can also consider a balanced version of the SVM where balancing is done by considering different values of C for each class in the box constraint of the dual formulation:

$$0 \leq \alpha_i \leq C_i, \forall i \in \{1, ..., n\}$$

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

We now plot the minDCF graphs for both balanced and unbalanced SVM in order to select an adequate value for C.

Figure 4: *minDCF plots of linear SVM. Unbalanced / Balanced with $\pi_T = 0.5$ / Balanced with $\pi_T = 0.1$ / Balanced with $\pi_T = 0.9$. Z-Normalized data with PCA $m = 7$.*




From the graphs we can observe that the two types of models seems to perform almost identical in terms of minDCF. For this reason we choose $C = 10^{-2}$ as our hyperparameter for both applications.

Table 3: *Linear Support Vector Machines with Z-Normalization*

| | 5-Folds | | |
|---|---|---|---|
| $\tilde{\pi}$ | 0.5 | 0.1 | 0.9 |
| | No PCA | | |
| Linear SVM($C = 10^{-2}$, *unbalanced*) | 0.118 | 0.224 | 0.581 |
| Linear SVM($C = 10^{-2}$, $\pi_T = 0.5$) | 0.121 | 0.223 | 0.560 |
| Linear SVM($C = 10^{-2}$, $\pi_T = 0.1$) | 0.117 | 0.222 | 0.574 |
| 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 | 0.581 |
| Linear SVM($C = 10^{-2}$, $\pi_T = 0.5$) | 0.121 | 0.223 | 0.560 |
| Linear SVM($C = 10^{-2}$, $\pi_T = 0.1$) | 0.117 | 0.222 | 0.574 |
| Linear SVM($C = 10^{-2}$, $\pi_T = 0.9$) | 0.145 | 0.253 | 0.557 |

We can again denote that PCA $m = 7$ does not worsen the results, on the contrary, they are practically identical. Rebalancing doesn't make much of a difference so from now on we will only consider the unbalanced application. We can denote again that, as seen for Logistic Regression, the model with $\pi_T = 0.1$ provides slightly better result.

### 3.4 Quadratic SVMs

The dual SVM formulation depends on the samples through dot products:

$$\boldsymbol{H_{ij}} = z_i z_j \boldsymbol{x}_i^T \boldsymbol{x}_j$$

Thanks to this property no explicit feature expansion is required to compute the scores, it's enough to be able to compute scalar products between training and test samples. If we have a function that efficiently computes dot-products in the expanded space, then both training and scoring can be performed by using k, a *kernel function*:

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{\Phi}(\boldsymbol{x}_i)^T \boldsymbol{\Phi}(\boldsymbol{x}_j^T)$$

that will allow us to compute a linear separation surface in the expanded space, which corresponds to a non-linear separation surface in the original feature space. We can employ two different types of kernel functions:

- **Radial Basis Function (RBF)**: $e^{-\gamma||\boldsymbol{x}_i - \boldsymbol{x}_j||^2}$
- **Polynomial**: $(\boldsymbol{x}_i^T \boldsymbol{x}_j + c)^d$

Below the minDCF graphs, plotted in order to select the value of $C$, $\gamma$ and $c$, alongside with the table containing the results.

Figure 5: *minDCF plots of quadratic SVM. RBF / Polynomial. priors $\tilde{\pi} = 0.5$, Z-Normalized data with PCA $m = 7$.*

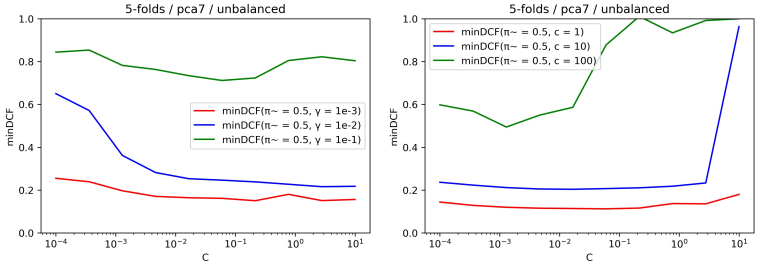



Table 4: *Quadratic SVMs with Z-Normalization*

| | 5-Folds | | |
|---|---|---|---|
| $\tilde{\pi}$ | 0.5 | 0.1 | 0.9 |
| | No PCA | | |
| RBF SVM($C = 10^{-1}$, $\gamma = 10^{-3}$) | 0.156 | 0.260 | 0.582 |
| Poly SVM($C = 10^{-3}$, $c = 1$, $d = 2$) | 0.130 | 0.232 | 0.705 |
| | PCA $m = 7$ | | |
| RBF SVM($C = 10^{-1}$, $\gamma = 10^{-3}$) | 0.156 | 0.260 | 0.582 |
| Poly SVM($C = 10^{-3}$, $c = 1$, $d = 2$) | 0.130 | 0.232 | 0.705 |

As expected quadratic classifiers perform worse than linear ones, for this reason we wont consider RBF and Poly SMVs as possible candidates for the calibration and evaluation phase of this analysis.

### 3.5 Gaussian Mixture Model

We end our classification task by talking about Gaussian Mixture Models. GMM works under the assumption that each sample is generated from a mixture of a finite number of Gaussian distributions with unknown parameters. The actual number of this distributions is an hyperparameter, below we plot different types of GMM (Full Covariance, Diagonal Covariance and Tied Covariance) in order to estimate the correct number of components.

Figure 6: *minDCF plots of GMM. Full Covariance / Diagonal Covariance / Tied Covariance. Z-Normalized data with PCA m = 7.*

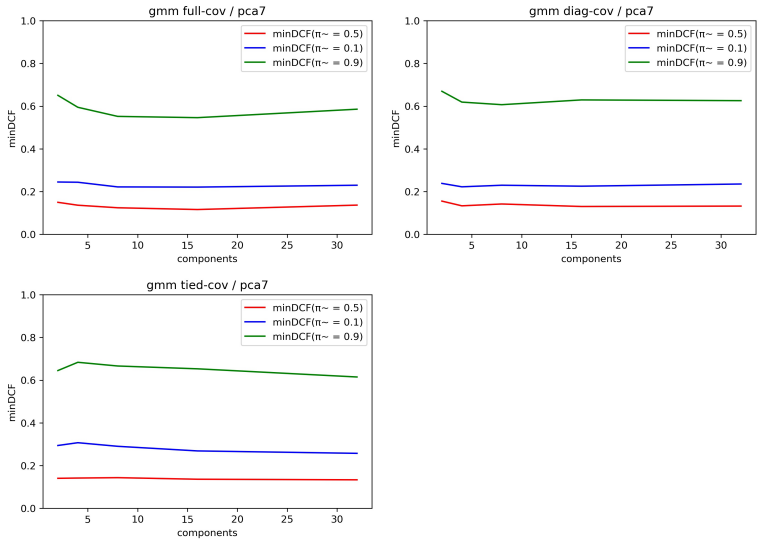



Table 5: *Gaussian Mixture Models with Z-Normalization*

| | 5-Folds | | |
|---|---|---|---|
| $\tilde{\pi}$ | 0.5 | 0.1 | 0.9 |
| | No PCA | | |
| GMM Full Cov (8 components) | 0.124 | 0.222 | 0.552 |
| GMM Diagonal Cov (16 components) | 0.130 | 0.225 | 0.629 |
| GMM Tied Cov (32 components) | 0.133 | 0.257 | 0.615 |
| | PCA $m = 7$ | | |
| GMM Full Cov (8 components) | 0.124 | 0.222 | 0.552 |
| GMM Diagonal Cov (16 components) | 0.130 | 0.225 | 0.629 |
| GMM Tied Cov (32 components) | 0.133 | 0.257 | 0.615 |

Overall GMM Full Cov with 8 components performs well, for this reason we will consider it as one of the possible candidates together with Tied Full-Cov, Logistic Regression and Linear SVM.

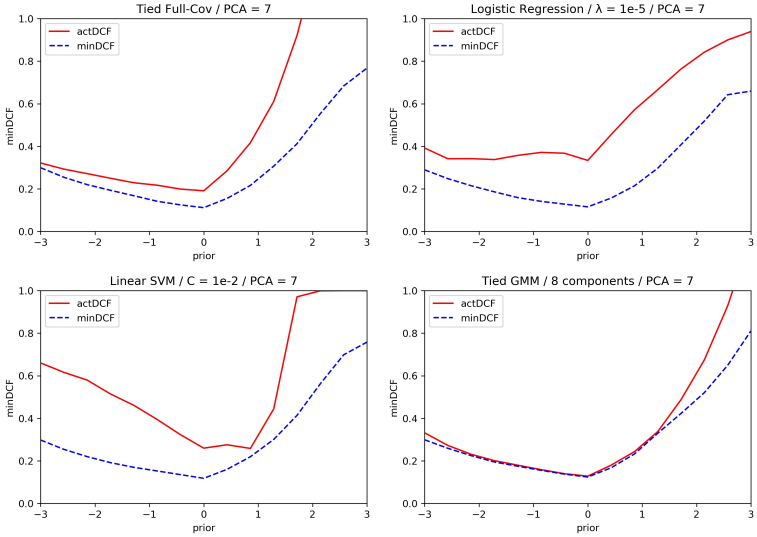
# 4 Scores calibration

We focus now on the calibration of the best performing models:

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

Up to now we considered only minDCF as a metric, however the actual cost that we would pay depends on the goodness of the threshold used to perform class assignment, for this reason we introduce now actual DCF. This differs from minDCF because the classification is now performed using the threshold corresponding to $\tilde{\pi}$ instead of testing all the possible thresholds and taking the lowest DCF. The Bayes Error Plots below show the differences between minDCF and actDCF for the selected models, a larger gap means a greater loss due to score mis-calibration.

Figure 7: *minDCF vs actDCF for uncalibrated classifiers. Z-Normalized data with PCA $m = 7$.*




As we can see the minDCF differs a lot from the value of the actDCF, for this reason we can try to employ a score calibration technique that consists in computing a transformation function that will map uncalibrated scores $s$ to calibrated ones $s_c$. If we assume that this function is linear in $s$ we can write it as:

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

$f(s)$ can be interpreted as the log-likelihood ratio for the two class (since it should provide well-calibrated scores), and the class posterior for a prior $\tilde{\pi}$ can be written as:

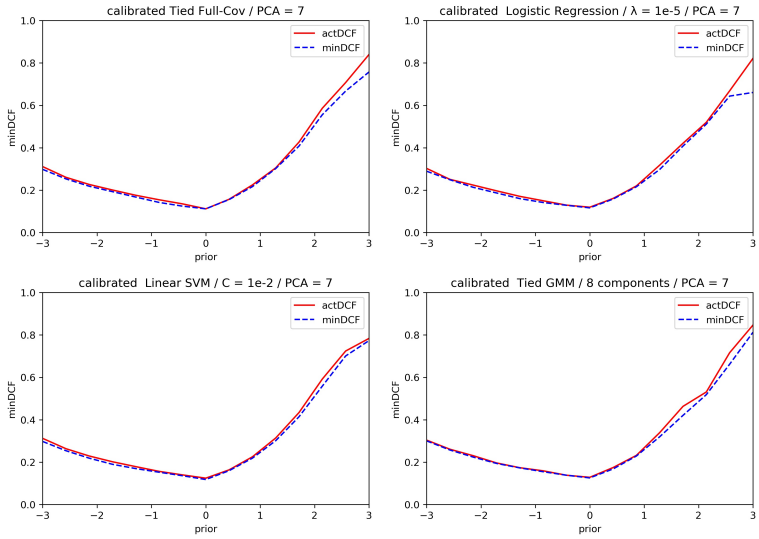
$$\log\left(\frac{P(C = Class_1|s)}{P(C = Class_0|s)}\right) = \alpha s + \beta + \log\frac{\tilde{\pi}}{1-\tilde{\pi}}$$

Interpreting $s$ as features and rewriting the equation we can obtain the same expression for the log posterior ratio of the Logistic Regression. if we let $\beta' = \beta + \log\frac{\tilde{\pi}}{1-\tilde{\pi}}$, then all we need to do is train the model to learn parameters $\alpha$ and $\beta'$. The final equation for recovering calibrated scores will be:

$$f(s) = \alpha s + \beta' - \log\frac{\tilde{\pi}}{1-\tilde{\pi}}$$

As a value for $\tilde{\pi}$ we select 0.5 since we want to optimize our application for this specific case, instead for hyperparameter $\lambda$ (of the calibration LogReg) we try $10^{-4}$.

Figure 8: *minDCF vs actDCF for calibrated classifiers. Z-Normalized data with PCA $m = 7$.*




By employing score calibration we successfully mapped miscalibrated scores to well-calibrated ones over a wide range of applications. Below the results in detail.

Table 6: *Selected models with score calibration and Z-Normalization. (names have been shorted for space reasons)*

| | minDCF | | | actDCF | | |
|---|---|---|---|---|---|---|
| $\tilde{\pi}$ | 0.5 | 0.1 | 0.9 | 0.5 | 0.1 | 0.9 |
| | | | No PCA | | | |
| Tied Full-Cov | 0.112 | 0.224 | 0.573 | 0.113 | 0.228 | 0.616 |
| LogReg($10^{-5}$, 0.5) | 0.115 | 0.219 | 0.533 | 0.121 | 0.225 | 0.541 |
| SVM($C = 10^{-2}$) | 0.118 | 0.224 | 0.584 | 0.124 | 0.232 | 0.589 |
| GMM Full Cov(8) | 0.124 | 0.228 | 0.536 | 0.128 | 0.235 | 0.579 |
| | | | PCA $m = 7$ | | | |
| Tied Full-Cov | 0.112 | 0.223 | 0.572 | 0.114 | 0.228 | 0.616 |
| LogReg($10^{-5}$, 0.5) | 0.114 | 0.217 | 0.543 | 0.115 | 0.225 | 0.553 |
| SVM($C = 10^{-2}$) | 0.118 | 0.224 | 0.581 | 0.126 | 0.233 | 0.588 |
| GMM Full Cov(8) | 0.124 | 0.222 | 0.552 | 0.127 | 0.231 | 0.583 |

# 5 HTRU2 evaluation

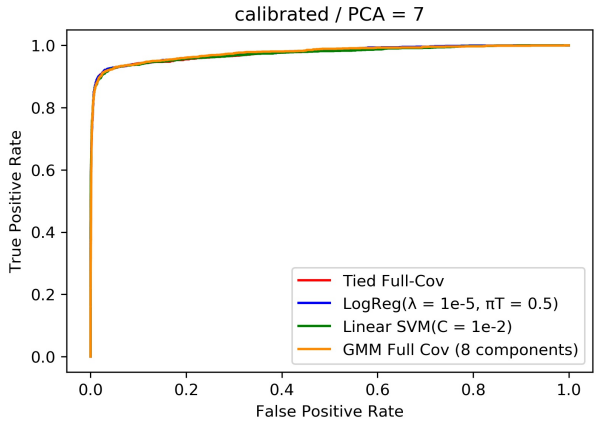
We now proceed to analyze the selected models' performances over the test set, since PCA $m = 7$ proved to be a good preprocessing strategy we will only consider this approach. We again consider performances in terms of minDCFs and actDCFs.

Table 7: *Selected classifiers with score calibration on evaluation set, Z-Normalization and PCA $m = 7$. (names have been shorted for space reasons)*

| | minDCF | | | actDCF | | |
|---|---|---|---|---|---|---|
| $\tilde{\pi}$ | 0.5 | 0.1 | 0.9 | 0.5 | 0.1 | 0.9 |
| | | | PCA $m = 7$ | | | |
| Tied Full-Cov | 0.111 | 0.216 | 0.580 | 0.114 | 0.224 | 0.600 |
| LogReg($10^{-5}$, 0.5) | 0.109 | 0.210 | 0.551 | 0.113 | 0.223 | 0.564 |
| SVM($C = 10^{-2}$) | 0.116 | 0.217 | 0.580 | 0.120 | 0.228 | 0.588 |
| GMM Full Cov(8) | 0.113 | 0.218 | 0.518 | 0.115 | 0.220 | 0.539 |

The results are consistent with the ones obtained during the previous analysis over the training set, with the Logistic Regression being, in this case, the best model (even if the differences between the models are very low). We can also utilize ROC (Receiver Operating Characteristic) curve for comparing the models, the best ones will have the highest Area Under Curve.

Figure 9: *ROC plot for the selected classifiers. Z-Normalized data with PCA $m = 7$.*




In our case the curve is pretty much the same for all the classifiers, the high slope in the left part means that the models are able to correctly classify $Class_1$ (True Pulsar) while keeping $Class_0$ (False Pulsar) to a minimum.

## 5.1 Conclusions

During our analysis we discovered that, for the HTRU2 dataset, linear classifiers performs better than quadratic ones. Performances evaluated over the evaluation set proved to be in line with the ones obtained during the Training step. This means that the choices we previously made were correct.

# References


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