POLITECNICO DI TORINO

Classification of HTRU2 dataset

Machine learning and pattern recognition

Giacomo Vitali and Marco La Gala

January 26, 2022

Abstract

In this paper, we analyze the HTRU2 dataset provided by Rob Lyon, and we report our results and considerations. HTRU2 is a dataset that describes a sample of pulsar candidates collected during the High Time Resolution Universe Survey (South).

We start by analyzing the dataset to understand its characteristics, then we apply some pre-processing like Z-normalization to avoid overflow errors with large numbers.

Then we analyze the performance of different classifiers like *Multivariate Gaussian Classifier*, *Logistic Regression*, *Support Vector Machine*, and *Gaussian Mixture Model* in terms of *min DCF* values. Then thanks to the comparison between *min DCF* and *actual DCF* and Bayes Error plot, we find the need for re-calibration.

Finally, we check if the choice of our hyper-parameters and the results obtained with the analysis are consistent with the test data, and we conclude that the best models are the linear ones because they provide a better classification on both training and test data.

Keywords: HTRU2 dataset, Classification algorithms, Multivariate Gaussian Classifier, Logistic Regression, Support Vector Machine, Gaussian Mixture Model.

1 Introduction

HTRU2 [1] is a dataset that describes a sample of pulsar candidates collected during the High Time Resolution Universe Survey (South) [2].

The pulsar is a highly magnetized rotating neutron star that produces radio emission detectable on Earth and provides the first indirect evidence for the existence of gravitational waves and the possibility to reveal extreme phenomena in neutron stars astrophysics.

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.

Therefore, the identification of pulsars in the universe is a prerequisite for pulsars and gravitational waves study. At present, a large number of pulsar searches have produced millions of pulsar candidates.

Without 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.

In the face of these large-scale data, if only relying on manual visual classification by experts in related fields, it will be a huge project. Since the emergence of machine learning, its theory and technology have become increasingly mature and have been successfully applied to astronomical research fields such as pulsar candidate screening [3].

Classification systems, in particular, are being widely adopted, which treat the candidate dataset as a binary classification problem.

Here the legitimate pulsar examples are a minority positive class, and spurious examples are the majority negative class.

The dataset contains 16,259 spurious examples caused by RFI/noise and 1,639 real pulsar examples. These examples were all checked by human annotators.

2 HRTU2 features

We have $17,898$ samples divided into two datasets (train and test) with $1,639$ positive (identified with class 1) and $16,259$ negative ex-

amples (identified with class 0).

Each candidate is described by 8 continuous variables and a single class variable. These variables 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.

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. The remaining four variables are similarly obtained from the DM-SNR curve.

We decide to apply *Z-normalization* to the dataset to prevent overflow errors during mathematical operations. It normalizes every value in the dataset such that the mean of all of the values is 0 and the standard deviation is 1 (centering and scaling to unit variance):

$$z_i = \frac{x_i - \mu}{\sigma} \tag{1}$$

where x_i is the observed value, μ is the mean of the sample, and σ is the standard deviation of the sample.

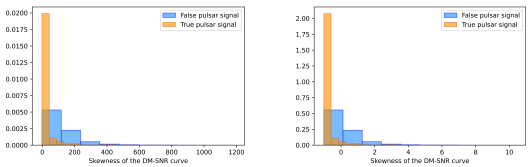
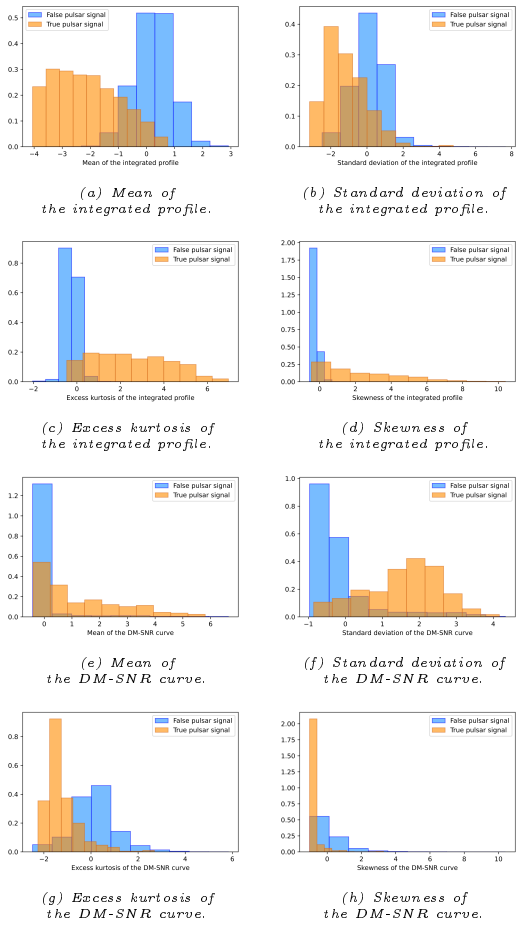


Figure 1: Histograms before and after Z-normalization of the Skewness of the DM-SNR curve feature.

In Figure 1 we show the distribution before and after Z-normalization, and we can notice that the distribution remains similar.

While in Figure 2 we show the histograms for each feature where the orange ones refers to *true pulsar signals* while the blue ones to *false pulsar signals*. We can see the presence of very few outliers, so we do not need other pre-processing.



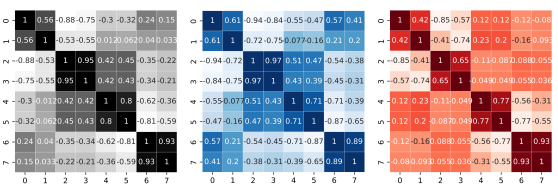
(a) Mean of the integrated profile. (b) Standard deviation of the integrated profile. (c) Excess kurtosis of the integrated profile. (d) Skewness of the integrated profile. (e) Mean of the DM-SNR curve. (f) Standard deviation of the DM-SNR curve. (g) Excess kurtosis of the DM-SNR curve. (h) Skewness of the DM-SNR curve.

Figure 2: Histograms of the HTRU2 dataset features (training set) after Z-normalization.

We employ the heat map to find the correlation between different features (Figure 3).

A heat map (or heatmap) is a representation of data in the form of a map or diagram in which data values are represented as colors which provides a visual summary of information and helps to understand complex datasets. A heat map shows the *Pearson correlation coefficient*:

$$\frac{Cov(X,Y)}{\sqrt{Var(X)}\sqrt{Var(Y)}} \tag{2}$$



(a) Whole training dataset (b) True pulsar signal class (c) False pulsar signal class

Figure 3: Heat maps with Pearson correlation coefficent of the whole dataset and the two classes.

We can see in Figure 3 that the features are

correlated in the same way on the two different classes and the whole dataset, so the correlation doesn't depend on the class. Here darker color implies a larger value for the *Pearson correlation coefficient*, and the correlation on the diagonal is always 1 because each feature it's fully correlated with itself.

There are 3 features hightly correlated (2-3, 4-5, 6-7), so we can try to reduce the original space with n dimension into a m subspace applying PCA with $n << m$. We will try PCA with $m = 7$, $m = 6$ and $m = 5$. We try PCA to understand if the mapping data to m uncorrelated features reduce the number of parameters to estimate without losing important information.

3 Classify HTRU2 features

To understand which model is most promising we can adopt two methodologies: Single-fold or K-fold.

For the first one, we can split the training dataset into development (for model training) and validation subsets. In this way, the final classifier will be the same that we evaluate on the validation set: at least for the validation set model selection and hyper-parameters will be optimal. We need to train fewer models, so training is faster, but we have less data for validation and model training.

For the K-fold method, 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, but we have 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.

For all the classifiers, we consider both approaches. Single-fold consists of 66% training data and 33% validation data. The K-fold is implemented with $K = 5$. In each step, we will take four folds for development (80% training data) and one fold for validation (20% validation data). In both cases, we shuffle our data before splitting.

We consider a uniform prior application that will be our main application:

$$(\widetilde{\pi}, C_{fp}, C_{fn}) = (0.5, 1, 1) \tag{3}$$

Also, we consider other two unbalanced applications:

$$(\widetilde{\pi}, C_{fp}, C_{fn}) = (0.1, 1, 1) \tag{4}$$
$$(\widetilde{\pi}, C_{fp}, C_{fn}) = (0.9, 1, 1) \tag{5}$$

We are interested in choosing the most promising approach. We, therefore, measure performance in terms of *normalized minimum detection costs* (*min DCF*). It measures the cost we would pay if we made optimal decisions for the test set (in our case validation set) using the recognizer scores. We will analyze how to choose an optimal threshold in a second stage.

3.1 Multivariate Gaussian

The first family of classifiers we take into consideration is the *Gaussian Classifier* that assumes Gaussian distribution of the data:

$$X \sim \mathcal{N}(\mu, \Sigma) \tag{6}$$

where μ is the mean and Σ is the covariance matrix.

We start from the *Multivariate Gaussian Classifier* that makes no other assumption of the data, and we proceed with *Gaussian Classifier with Naive Bayes assumption* and *Gaussian Classifier with Tied Covariances*.

	Single Fold			5-fold		
	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$
Z-normalized Features - no PCA						
Full-Cov	0.116	0.257	0.559	0.142	0.283	0.656
Diag-Cov	0.171	0.264	0.621	0.193	0.312	0.747
Tied Full-Cov	0.091	0.209	0.474	0.112	0.223	0.575
Z-normalized Features - PCA (m = 7)						
Full-Cov	0.115	0.264	0.523	0.139	0.298	0.629
Diag-Cov	0.182	0.485	0.563	0.214	0.505	0.724
Tied Full-Cov	0.091	0.209	0.474	0.112	0.222	0.574
Z-normalized Features - PCA (m = 6)						
Full-Cov	0.120	0.261	0.544	0.152	0.285	0.622
Diag-Cov	0.195	0.506	0.568	0.223	0.525	0.722
Tied Full-Cov	0.109	0.236	0.507	0.140	0.258	0.582
Z-normalized Features - PCA (m = 5)						
Full-Cov	0.129	0.240	0.630	0.148	0.245	0.644
Diag-Cov	0.197	0.425	0.595	0.220	0.453	0.734
Tied Full-Cov	0.124	0.237	0.517	0.150	0.261	0.575

Table 1: MVG Classifiers - min-DCF on the validation set.

In Table 1, we can observe the *min DCF* we obtained on the validation set using different priors and different dimensions for the PCA.

First, we can notice that PCA is not effective, even though PCA $m = 7$ does not degrade performance either. Instead, further reductions (PCA $m = 6$ and PCA $m = 5$) decrease accuracy.

The Tied Full-Cov model achieves the best performance without PCA and with PCA $m = 7$. We see that the hypothesis of interpreting the covariance matrix as representing noise independent of the class is correct, and the covariance matrices probably are similar for the two classes.

The Diag-Cov, instead, is the worst model and it makes us believe that the assumption of inde-

pendent components may be wrong. The Full-Cov classifier obtains discrete results, but it is not good as the Tied Full-Cov. Maybe, this means that we don't have enough data to estimate all the parameters.

It is also clear that all the models have worse performances for the unbalanced tasks with respect to the balanced one. The considerations are valid for both Single-fold and K-fold.

From this analysis, we understand that linear models perform better than quadratic ones and PCA $m = 7$ preserves most of the useful information to perform a decision. However, we keep considering PCA $m = 6$ in the following classifier to see if that consideration remains valid. Also, from now on, we will not examine anymore PCA $m = 5$.

We now turn our attention to discriminative approaches.

3.2 Linear Logistic Regression

We start considering regularized linear Logistic Regression. Since classes are not balanced, we re-balance the costs of the different classes, minimizing:

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

$$+\frac{1-\pi_T}{n_F}\sum_{i=1|c_i=0}^{n} \log\left(1+e^{-z_i(w^T x_i+b)}\right) \quad (7)$$

The model parameters are w and b. It assumes that the decision rules are linear hyperplanes orthogonal to w:

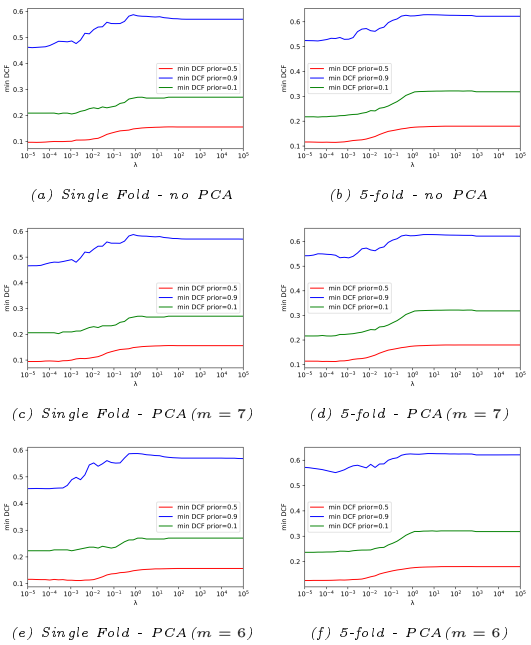
$$w^T x + b \quad (8)$$

Logistic Regression will find the hyperplane that maximizes class probabilities.

We decide to use cross-validation to find the value of the hyper-parameter λ that most minimize the *min DCF*. λ is a regularization coefficient that tells us how much we should weigh the cost due to having a high norm with respect to the classification cost.

Regularization allows reducing the risk of over-fitting the training data. Of course, if we select a large value of λ without any other consideration, we will obtain a solution that has a small norm, without any guarantee that it is able to separate well the classes. On the other hand, if λ is too small, we will get a solution with a higher norm that has good separation on the training set but may have poor classification accuracy for unseen data (the model is over-confident).

From Figure 4, we can see that the results with Single-fold and K-fold are similar. However, we consider mostly the K-fold version because the evaluation results are more reliable and the final model should be more robust due to an increased number of training samples. The best value for the hyper-parameter λ is 10^{-4}.



(a) Single Fold - no PCA (b) 5-fold - no PCA

(c) Single Fold - PCA ($m = 7$) (d) 5-fold - PCA ($m = 7$)

(e) Single Fold - PCA ($m = 6$) (f) 5-fold - PCA ($m = 6$)

Figure 4: Linear Logistic Regression - min DCF for different values of λ.

	Single Fold			5-fold		
	$\tilde{\pi}=0.5$	$\tilde{\pi}=0.1$	$\tilde{\pi}=0.9$	$\tilde{\pi}=0.5$	$\tilde{\pi}=0.1$	$\tilde{\pi}=0.9$
Z-Normalized Features - no PCA						
Log Reg($\lambda=10^{-4}$, $\pi_T=0.5$)	0.099	0.209	0.468	0.115	0.218	0.528
Log Reg($\lambda=10^{-4}$, $\pi_T=0.1$)	0.095	0.199	0.486	0.114	0.211	0.556
Log Reg($\lambda=10^{-4}$, $\pi_T=0.9$)	0.102	0.203	0.464	0.118	0.221	0.520
Z-Normalized Features - PCA (m = 7)						
Log Reg($\lambda=10^{-4}$, $\pi_T=0.5$)	0.096	0.206	0.477	0.113	0.216	0.548
Log Reg($\lambda=10^{-4}$, $\pi_T=0.1$)	0.098	0.199	0.482	0.114	0.211	0.555
Log Reg($\lambda=10^{-4}$, $\pi_T=0.9$)	0.101	0.206	0.477	0.119	0.218	0.526
Z-Normalized Features - PCA (m = 6)						
Log Reg($\lambda=10^{-4}$, $\pi_T=0.5$)	0.114	0.223	0.456	0.126	0.238	0.559
Log Reg($\lambda=10^{-4}$, $\pi_T=0.1$)	0.116	0.230	0.465	0.126	0.242	0.580
Log Reg($\lambda=10^{-4}$, $\pi_T=0.9$)	0.122	0.223	0.494	0.135	0.242	0.524

Table 2: Logistic Regression - min-DCF on the validation set.

From Table 2, we can see that re-balancing the cost of different classes through π_T does not improve the model for the other two applications. The model that performs better is the one with PCA $m = 7$ and $\pi_T = 0.5$. However, we get pretty similar *min DCF* values for all the three π_T.

Also, we can observe that the results from PCA $m = 6$ are the worst ones as we see on the MVG classifiers.

As we can see from Table 3, the two classifiers seen so far are pretty similar, and we can take into account that Logistic Regression obtains little better performances also with unbalanced applications.

	Single Fold			5-fold		
	$\tilde{\pi}=0.5$	$\tilde{\pi}=0.1$	$\tilde{\pi}=0.9$	$\tilde{\pi}=0.5$	$\tilde{\pi}=0.1$	$\tilde{\pi}=0.9$
Z-Normalized Features - PCA (m = 7)						
Tied Full-Cov	0.091	0.209	0.474	0.112	0.222	0.574
Log Reg($\lambda=10^{-4}$, $\pi_T=0.5$)	0.096	0.206	0.477	0.113	0.216	0.548

Table 3: Comparison between most promising model.

We consider only the linear version of the Logistic Regression and not the quadratic one because we expect to obtain worse results in the latter. However, we will see some quadratic classifiers in the SVM section.

3.3 Support Vector Machine

In this section, we will see *Linear SVM* with and without class balancing and *Quadratic SVM* with *Polynomial* and *RBF kernel*.

Support Vector Machine is similar to Logistic Regression but it considers a different interpretation. Instead of finding the hyperplane that maximizes the class probabilities, SVM will find the hyperplane that separates the classes with the largest margin. Here the regularization term has a physical interpretation.

Intuitively, we want to find the model parameters w and b that maximize the distance of the closest point. So, for each w and b, we can compute the distances from all the points with respect to the separation boundary and select the minimum one (minimum distance). Then, we want to find the solution that has the maximum minimum distance.

$$w^*, b^* = \arg\max_{w,b} \min_{i\in\{1...n\}} d(x_i) \quad \text{subject to} \quad z_i(w^T x + b) > 0 \tag{9}$$

where d is the distance and x_i is the training point.

Direct optimization of Equation 9 is not-trivial, so we further transform the problem to arrive at an easier-to-solve equivalent problem.

$$J(w,b) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \max\left[0, 1 - z_i(w^T x_i + b)\right] \tag{10}$$

To solve the *Primal SVM problem* we consider a Lagrangian formulation of the problem. Since the problem is convex, we can equivalently maximize the Lagrangian formulation w.r.t. α_i while requiring that the derivatives w.r.t. w and b vanish subject to $\alpha_i > 0$.

We found $w = \sum_{i=1}^{n} \alpha_i z_i x_i$ and $\sum_{i=1}^{n} \alpha_i z_i = 0$ and when we substitute them inside the Lagrangian formulation we found the *Dual SVM formulation*:

$$J^D(\alpha) = -\frac{1}{2}\alpha^T \mathbf{H}\alpha + \alpha^T \mathbf{1} \tag{11}$$

subject to

$$0 \le \alpha_i \le C, \quad \forall i \in \{1...n\}, \quad \sum_{i=1}^{n} \alpha_i z_i = 0$$

where $\mathbf{1}$ is a n-dimensional vector of ones, and $\mathbf{H}$ is a matrix whose elements are

$$\mathbf{H}_{i,j} = z_i z_j x_i^T x_j \tag{12}$$

The model parameter b^* can be computed by considering a sample x_i that lies on the margin.

$$z_i({w^*}^T x_i + b^*) = 1 \tag{13}$$

The Dual formulation is differentiable, however it contains both box constraints of the form $a \le \alpha_i \le b$ and the additional constraint $\sum_{i=1}^{n} \alpha_i z_i = 0$. A version of the L-BFGS algorithm called L-BFGS-B is able to handle box constraints, however, it cannot incorporate the latter. We therefore slightly modify the SVM problem to make the constraint disappear. The dual formulation becomes:

$$\widehat{J}^D(\alpha) = -\frac{1}{2}\alpha^T \widehat{\mathbf{H}}\alpha + \alpha^T \mathbf{1} \quad \text{subject to} \quad 0 \le \alpha_i \le C, \quad \forall i \in \{1...n\} \tag{14}$$

and, since we use the mapping

$$\widehat{x}_i = \begin{bmatrix} x_i \\ K \end{bmatrix} \tag{15}$$

the matrix $\widehat{\mathbf{H}}$ becomes:

$$\widehat{\mathbf{H}}_{i,j} = z_i z_j \widehat{x}_i^T \widehat{x}_j \tag{16}$$

For linear SVM we need to tune the hyperparameters C and K. We decide to try only $K = 1.0$ and $K = 10.0$. Again, we use both Single-fold and K-fold cross-validation.

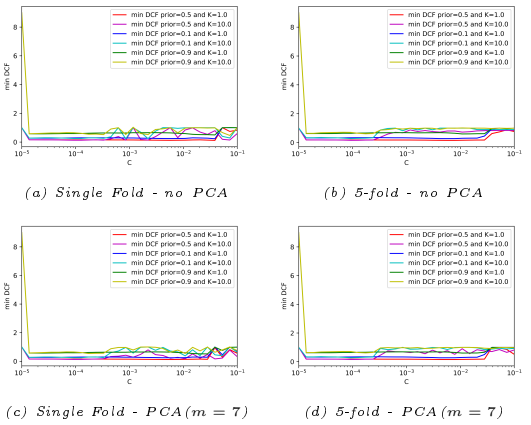
From the unbalanced SVM on Figure 5, we can see that the best value for K is 1.0, and the best value of C is 10^{-2}. Again, the K-fold results are consistent with the Single-fold results, suggesting that our model behaves correctly.

We can also consider the balanced version. Balancing is done by considering a different value of C for the different classes in the box constraints of the dual formulation:

$$0 \le \alpha_i \le C_i \tag{17}$$

where $C_i = C_T$ for samples of class $\mathcal{H}_T$ and $C_i = C_F$ for samples of class $\mathcal{H}_F$. We select C_T and C_F as:

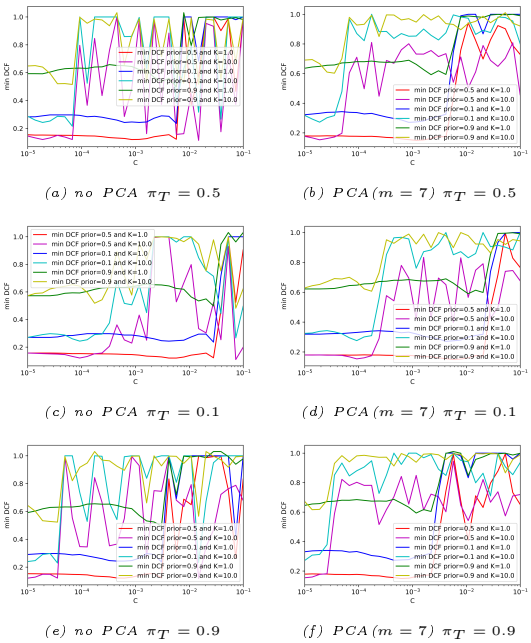
$$C_T = C\frac{\pi_T}{\pi_T^{emp}} \qquad C_F = C\frac{\pi_F}{\pi_F^{emp}} \tag{18}$$



(a) Single Fold - no PCA (b) 5-fold - no PCA

(c) Single Fold - PCA ($m = 7$) (d) 5-fold - PCA ($m = 7$)

Figure 5: Linear SVM (without class balancing) - min DCF for different values of C.



(a) no PCA $\pi_T = 0.5$ (b) PCA ($m = 7$) $\pi_T = 0.5$

(c) no PCA $\pi_T = 0.1$ (d) PCA ($m = 7$) $\pi_T = 0.1$

(e) no PCA $\pi_T = 0.9$ (f) PCA ($m = 7$) $\pi_T = 0.9$

Figure 6: Linear SVM (with class balancing) - min DCF for different values of C - no PCA.

where π_T^{emp} and π_F^{emp} are the empirical priors for the two classes computed over the training set.

From Figure 6, we can see that $K = 1$ provides a better and more stable *min DCF*. Even if prior $\widetilde{\pi} = 0.5$ and $K = 10$ provide a better *min DCF* with respect to prior $\widetilde{\pi} = 0.5$ and $K = 1$, we decide to take the latter because it remains with pretty much the same values for a wide range of C. We decide to choose a different value of the hyper-parameter C for each prior.

	Single Fold			5-fold		
	$\widetilde{\pi} = 0.5$	$\widetilde{\pi} = 0.1$	$\widetilde{\pi} = 0.9$	$\widetilde{\pi} = 0.5$	$\widetilde{\pi} = 0.1$	$\widetilde{\pi} = 0.9$
Z-Normalized Features - no PCA						
Linear SVM($C = 10^{-2}$, *unbalanced*)	0.130	0.247	0.600	0.158	0.287	0.612
Linear SVM($C = 10^{-3}, \pi_T = 0.5$)	0.121	0.243	0.640	0.151	0.274	0.677
Linear SVM($C = 6 * 10^{-3}, \pi_T = 0.1$)	0.122	0.247	0.622	0.151	0.276	0.656
Linear SVM($C = 7 * 10^{-4}, \pi_T = 0.9$)	0.122	0.247	0.620	0.152	0.272	0.643
Z-Normalized Features - PCA (m = 7)						
Linear SVM($C = 10^{-2}$, *unbalanced*)	0.130	0.250	0.605	0.159	0.285	0.618
Linear SVM($C = 10^{-3}, \pi_T = 0.5$)	0.121	0.243	0.641	0.152	0.275	0.678
Linear SVM($C = 6 * 10^{-3}, \pi_T = 0.1$)	0.122	0.247	0.623	0.150	0.276	0.658
Linear SVM($C = 7 * 10^{-4}, \pi_T = 0.9$)	0.122	0.247	0.621	0.152	0.272	0.651
Z-Normalized Features - PCA (m = 6)						
Linear SVM($C = 10^{-2}$, *unbalanced*)	0.136	0.260	0.611	0.165	0.293	0.614
Linear SVM($C = 10^{-3}, \pi_T = 0.5$)	0.123	0.253	0.636	0.153	0.282	0.679
Linear SVM($C = 6 * 10^{-3}, \pi_T = 0.1$)	0.123	0.250	0.622	0.154	0.286	0.662
Linear SVM($C = 7 * 10^{-4}, \pi_T = 0.9$)	0.125	0.253	0.625	0.154	0.289	0.659

Table 4: Linear SVM (with and without class balancing) - min-DCF on the validation set.

From Table 4, we can see that class re-balancing improves the model, so we will mainly consider the linear model with $\pi_T = 0.5$ and $C = 10^{-3}$.

We will now analyze the non-linear formulation. SVM allows non-linear classification through an implicit expansion of the features in a higher-dimensional space. The *Dual SVM formulation* depends on the training samples only through dot-products, and we can compute a classification score through scalar products between training and evaluation samples.

$$s(x_t) = \sum_{i=1}^{n} \alpha_i z_i x_i^T x_t + b \tag{19}$$

SVM does not require that we explicitly compute the feature expansion, because it's sufficient that we are able to compute the scalar product between the expanded features:

$$k(x_1, x_2) = \phi(x_1)^T \phi(x_2) \tag{20}$$

where function k is called *kernel function*.

We will try two different kernels:

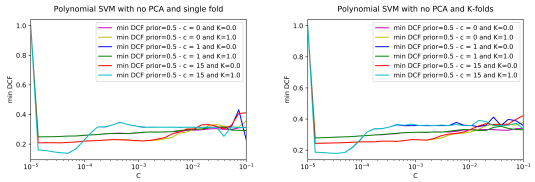
1. Polynomial kernel of degree d: $k(x_1, x_2) = (x_1^T x_2 + c)^d$
2. Radial Basis Function kernel: $k(x_1, x_2) = e^{-\gamma \|x_1 - x_2\|^2}$

So, for the Polynomial SVM, we need to estimate two more hyper-parameters c and d. We decided to fix $d = 2$ and perform a cross-validation to find c and C.

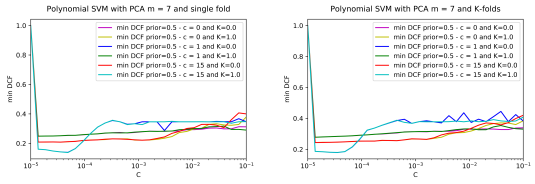
From Figure 7, we can see that different values of c and C provide different values of *min DCF* and the best values are $c = 15$ and $C = 5 * 10^{-5}$.

Also, in the RBF SVM, we need to estimate two hyper-parameters called γ and C. Using again cross-validation we find that the best values are $\gamma = 10^{-3}$ and $C = 10^{-1}$, as we can see in Figure 8.

From Table 5, we can see that linear approaches perform better than the non-linear

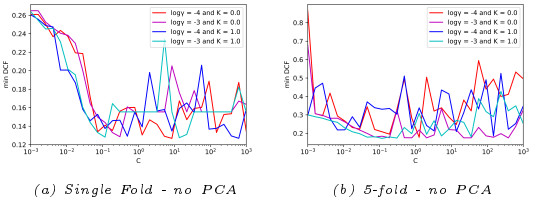


(a) Single Fold - no PCA *(b) 5-fold - no PCA*

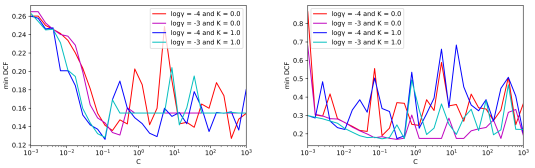


(c) Single Fold - PCA(m = 7) *(d) 5-fold - PCA(m = 7)*

Figure 7: Polynomial SVM - min DCF for different values of C - no PCA and PCA $m = 7$ with $\pi_T = 0.5$.



(a) Single Fold - no PCA *(b) 5-fold - no PCA*



(c) Single Fold - PCA(m = 7) *(d) 5-fold - PCA(m = 7)*

Figure 8: RBF SVM - min DCF for different values of C - no PCA and PCA $m = 7$ with $\pi_T = 0.5$.

ones. So, we decided to keep the linear SVM with $C = 10^{-3}$ and $\pi_T = 0.5$. Also, as we have seen before, PCA doesn't provide improvements with respect to the no PCA case.

We can see in Table 6 that the linear SVM is not very efficient when we compare it to the other models seen up to now.

Now we turn our attention again to generative approaches.

3.4 Gaussian Mixture Model

The last model we consider is a generative approach based on training a GMM over the data of each class. GMM can approximate generic distributions, so we expect to obtain better results than the Gaussian model.

We consider GMM with full covariance, with full diagonal and tied covariance. For tied covariance models, tying takes place at the class level, so different classes have different covariance matrices.

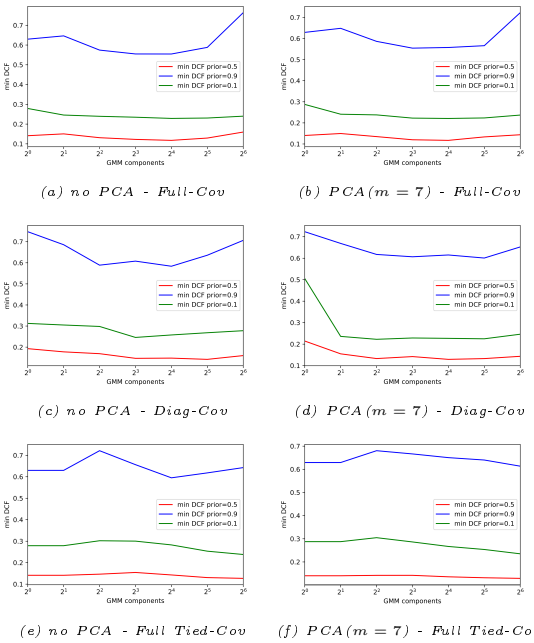
	Single Fold			5-fold		
	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$
Z-Normalized Features - no PCA						
Linear SVM($C = 10^{-3}$, $\pi_T = 0.5$)	0.121	0.243	0.640	0.151	0.274	0.677
Polynomial SVM($C = 5 * 10^{-5}$, $c = 15, d = 2$)	0.139	0.301	0.537	0.185	0.389	0.630
RBF SVM($C = 10^{-1}$, $\gamma = 10^{-3}$)	0.131	0.240	0.617	0.211	0.330	0.743
Z-Normalized Features - PCA (m = 7)						
Linear SVM($C = 10^{-3}$, $\pi_T = 0.5$)	0.121	0.243	0.641	0.152	0.275	0.678
Polynomial SVM($C = 5 * 10^{-5}$, $c = 15, d = 2$)	0.139	0.301	0.539	0.186	0.389	0.630
RBF SVM($C = 10^{-1}$, $\gamma = 10^{-3}$)	0.131	0.240	0.612	0.175	0.257	0.680
Z-Normalized Features - PCA (m = 6)						
Linear SVM($C = 10^{-3}$, $\pi_T = 0.5$)	0.123	0.253	0.636	0.153	0.282	0.679
Polynomial SVM($C = 5 * 10^{-5}$, $c = 15, d = 2$)	0.145	0.304	0.538	0.191	0.395	0.641
RBF SVM($C = 10^{-1}$, $\gamma = 10^{-3}$)	0.129	0.236	0.609	0.374	0.667	0.910

Table 5: Linear and non-linear SVM - min-DCF on the validation set.

	Single Fold			5-fold		
	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$
Z-Normalized Features - PCA (m = 7)						
Tied Full-Cov	0.091	0.209	0.474	0.112	0.222	0.574
Log Reg($\lambda = 10^{-4}$, $\pi_T = 0.5$)	0.096	0.206	0.477	0.113	0.216	0.548
Linear SVM($C = 10^{-3}$, $\pi_T = 0.5$)	0.121	0.243	0.641	0.152	0.275	0.678

Table 6: Comparison between most promising model.

The number of components (GMM components) cannot be estimated from the likelihood because they will tend to be infinite. This is because the likelihood will increase if we increase the number of components. There are several criteria to find the "right" number of Gaussians and we choose to employ the cross-validation to evaluate how good the model is on the validation set, and based on that we can find the best number of components. We try the cross-validation both on single-fold and K-fold, but we report in Figure 9 only the latter.



(a) no PCA - Full-Cov *(b) PCA(m = 7) - Full-Cov*

(c) no PCA - Diag-Cov *(d) PCA(m = 7) - Diag-Cov*

(e) no PCA - Full Tied-Cov *(f) PCA(m = 7) - Full Tied-Cov*

Figure 9: GMM - min DCF for different values of GMM components.

From Figure 9, we can see that the application with $\widetilde{\pi} = 0.9$ is worse than the others. The best application is the one with $\widetilde{\pi} = 0.5$.

Taking into consideration the main application (red line) we can show that Full-Cov and Diag-Cov have similar performance and they show some degree of over-fitting when the number of components becomes large. With Tied Full-Cov GMM we can employ a large number of components because we assume that all components have the same covariance matrix differently from non-Tied models where we need to estimate one covariance matrix for each cluster.

We can observe that we have different values for the best number of components for each model. We select 16 components for the Full-Cov model, 32 for the Diag-Cov, and 64 for the Tied Full-Cov model.

	Single Fold			5-fold		
	$\widetilde{\pi} = 0.5$	$\widetilde{\pi} = 0.1$	$\widetilde{\pi} = 0.9$	$\widetilde{\pi} = 0.5$	$\widetilde{\pi} = 0.1$	$\widetilde{\pi} = 0.9$
Z-Normalized Features - no PCA						
Full-Cov, 16-G	0.104	0.202	0.520	0.118	0.229	0.555
Diag-Cov, 32-G	0.122	0.236	0.477	0.142	0.268	0.636
Tied Full-Cov, 64-G	0.110	0.226	0.560	0.127	0.238	0.643
Z-Normalized Features - PCA (m = 7)						
Full-Cov, 16-G	0.110	0.206	0.548	0.117	0.221	0.558
Diag-Cov, 32-G	0.102	0.199	0.481	0.133	0.225	0.601
Tied Full-Cov, 64-G	0.126	0.253	0.639	0.129	0.235	0.613
Z-Normalized Features - PCA (m = 6)						
Full-Cov, 16-G	0.117	0.209	0.583	0.127	0.217	0.600
Diag-Cov, 32-G	0.118	0.247	0.608	0.148	0.260	0.647
Tied Full-Cov, 64-G	0.128	0.233	0.692	0.135	0.259	0.671

Table 7: GMM - min DCF on the validation set.

From Table 7, we can see that the Full-Cov model with 16 components is the better one followed by the Tied Full-Cov with 64 components.

As always, PCA doesn't provide any improvement.

	Single Fold			5-fold		
	$\widetilde{\pi} = 0.5$	$\widetilde{\pi} = 0.1$	$\widetilde{\pi} = 0.9$	$\widetilde{\pi} = 0.5$	$\widetilde{\pi} = 0.1$	$\widetilde{\pi} = 0.9$
Z-Normalized Features - PCA (m = 7)						
Tied Full-Cov	0.091	0.209	0.474	0.112	0.222	0.574
Log Reg ($\lambda = 10^{-4}$, $\pi_T = 0.5$)	0.096	0.206	0.477	0.113	0.216	0.548
Linear SVM($C = 10^{-3}$, $\pi_T = 0.5$)	0.121	0.243	0.641	0.152	0.275	0.678
Full-Cov, 16-G	0.110	0.206	0.548	0.117	0.221	0.558

Table 8: Comparison between most promising model.

From Table 8 that summarizes all the best models we have seen in the report, we can observe that the Tied Full-Cov and the Logistic Regression are the best ones. We decide to also keep the GMM Full-Cov model and discard the Linear SVM because the former performs better than the latter.

4 Score Calibration

Up to now, we consider only *min DCF* metrics that allow us to compare different models. However, it measures the cost we would pay if we made optimal decisions for the evaluation set using the recognizer scores.

In the binary case, the cost we would pay depends on the goodness of the threshold we choose to perform the class assignment. For this reason, we now turn our attention to *actual DCF*.

If the scores are well-calibrated, the optimal threshold that optimizes the *Bayes risk* is:

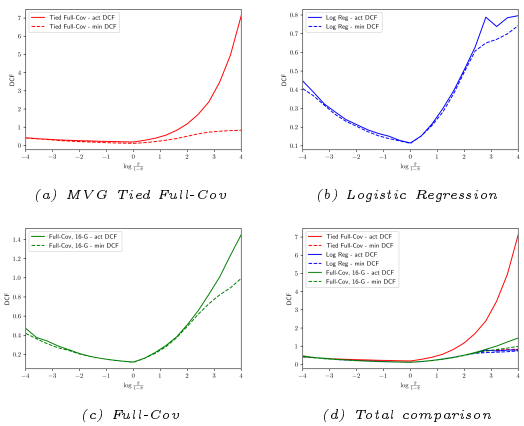
$$t = -\log \frac{\widetilde{\pi}}{1 - \widetilde{\pi}} \tag{21}$$

We can evaluate the *actual DCF* to assess how good the model would be if we were using the theoretical threshold for each application instead of the optimal one.

	$\widetilde{\pi} = 0.5$		$\widetilde{\pi} = 0.1$		$\widetilde{\pi} = 0.9$	
	min DCF	act DCF	min DCF	act DCF	min DCF	act DCF
Z-Normalized Features - PCA (m = 7)						
Tied Full-Cov	0.112	0.190	0.222	0.274	0.574	1.422
Log Reg($\lambda = 10^{-4}$, $\pi_T = 0.5$)	0.113	0.115	0.216	0.226	0.548	0.567
Full-Cov, 16-G	0.117	0.121	0.221	0.228	0.558	0.596

Table 9: Comparison between min DCF and actual DCF for the most promising models.

From Table 9, we can observe a comparison between *min DCF* and *actual DCF* for each model and each application. We can see that for the Tied Full-Cov model, the scores are not well calibrated and for the $\widetilde{\pi} = 0.5$ we have a loss of $\approx 70\%$ that becomes of $\approx 147\%$ with $\widetilde{\pi} = 0.9$. The other two models provide scores already well-calibrated. This is confirmed by the *Bayes error plots*, which show the DCF for different applications.



(a) MVG Tied Full-Cov (b) Logistic Regression

(c) Full-Cov (d) Total comparison

Figure 10: Bayes error plot for each model before calibration.

From Figure 10, we can see that Logistic Regression (Figure 10b) and GMM full covariance with 16 components (Figure 10c) are already well-calibrated for prior log-odds lower than 2. Instead, the MVG Tied Full-Cov performs worse than the others, especially when the prior log-odds is greater than 0. This is also evident in the plot with all the models (Figure 10d).

For this reason, we can consider an approach that consists of re-calibrating the scores. It means transforming the scores so that the theoretical threshold provides close to optimal values over a wide range of effective prior $\widetilde{\pi}$.

We want to compute a transformation function f that maps the classifiers scores s to well-calibrated scores $s_{cal} = f(s)$. We assume that function f is a linear function:

$$f(s) = \alpha s + \beta \tag{22}$$

Since $f(s)$ should produce well-calibrated scores, it can be interpreted as the log-likelihood ratio for the two class hypotheses:

$$f(s) = \log \frac{f_{S|C}(s|\mathcal{H}_T)}{f_{S|C}(s|\mathcal{H}_F)} = \alpha s + \beta \tag{23}$$

The class posterior probability for prior $\widetilde{\pi}$ correspond to:

$$\log \frac{P(C = \mathcal{H}_T|s)}{P(C = \mathcal{H}_F|s)} = \alpha s + \beta + \log \frac{\widetilde{\pi}}{1 - \widetilde{\pi}} \tag{24}$$

If we interpret the scores as features, this has a very similar expression as the log posterior ratio of the Logistic Regression model. If we let $\beta' = \beta + \log \frac{\widetilde{\pi}}{1-\widetilde{\pi}}$, then we have the same model. We can employ the prior-weighted Logistic Regression model to learn the model parameters α , β' over our training scores. To recover the calibrated score $f(s)$ we will need to compute:

$$f(s) = \alpha s + \beta = \alpha s + \beta' - \log \frac{\widetilde{\pi}}{1 - \widetilde{\pi}} \tag{25}$$

Note that we have to specify a prior $\widetilde{\pi}$, and we are still effectively optimizing the calibration for a specific application $\widetilde{\pi}$. However, as we will see, the model will still provide good calibration for different applications. We consider $\widetilde{\pi} = 0.5$.

	$\widetilde{\pi} = 0.5$		$\widetilde{\pi} = 0.1$		$\widetilde{\pi} = 0.9$	
	min DCF	act DCF	min DCF	act DCF	min DCF	act DCF
Z-Normalized Features - PCA (m = 7)						
Tied Full-Cov (cal.)	0.112	0.114	0.222	0.227	0.574	0.612
Log Reg($\lambda = 10^{-4}$, $\pi_T = 0.5$) (cal.)	0.113	0.114	0.216	0.226	0.548	0.562
Full-Cov, 16-G (cal.)	0.117	0.122	0.221	0.250	0.558	0.600

Table 10: Comparison between min DCF and actual DCF for the most promising models after calibration.

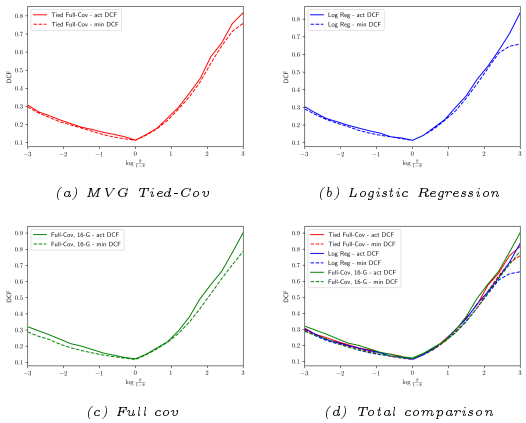


Figure 11: Bayes error plot for each model after calibration.

From Table 10, we can see that the re-calibration works very well, and the values of the *actual DCF* becomes very similar to the *min DCF* ones, especially for the $\widetilde{\pi} = 0.5$.

We can also notice that the *actual DCF* Tied Full-Cov model with $\widetilde{\pi} = 0.9$ improves a lot.

As evident from Figure 11, now the application is well calibrated for all three models. From the total comparison (Figure 11d), we can see that the three models provide the pretty same value of *min DCF*.

5 Experimental Results

In this section we analyze how these models will behave on the test set in terms of *min DCF*. We train the model with the training set and use the test set as evaluation data.

From Table 11, we can see that the results are consistent with those obtained on the validation set. The best model is the Logistic Regression with $\pi_T = 0.1$ and PCA $m = 7$, but also MVG Tied Full-Cov (no PCA), and Logistic Regression with both $\pi_T = 0.5$ and $\pi_T = 0.1$ (no PCA) are pretty good.

We can notice that for Logistic Regression the best results for our primary application ($\widetilde{\pi} = 0.5$) are obtained with $\pi_T = 0.1$, however the difference with respect to models using $\pi_T = 0.5$ is not significant.

The similarity between validation and evaluation results suggests that there are no relevant differences between the evaluation population and the training population.

Also, from the results, we can notice that PCA is not effective, but it's not detrimental, so we decide to compare our best classifiers through a *ROC plot* considering the models with PCA $m = 7$. In this way, we can reduce the dimensionality of the feature space without

	$\tilde{\pi}=0.5$	$\tilde{\pi}=0.1$	$\tilde{\pi}=0.9$
Z-Normalized Features - no PCA			
MVG Full-Cov	0.140	0.284	0.646
MVG Diag-Cov	0.185	0.331	0.621
MVG Tied Full-Cov	0.110	0.207	0.591
Log Reg($\lambda=10^{-4}, \pi_T=0.5$)	0.110	0.199	0.534
Log Reg($\lambda=10^{-4}, \pi_T=0.1$)	0.110	0.197	0.534
Log Reg($\lambda=10^{-4}, \pi_T=0.9$)	0.115	0.206	0.510
Linear SVM($C=10^{-2}$, unbalanced)	0.148	0.292	0.603
Linear SVM($C=10^{-3}, \pi_T=0.5$)	0.145	0.285	0.599
Linear SVM($C=6*10^{-3}, \pi_T=0.1$)	0.144	0.281	0.613
Linear SVM($C=7*10^{-4}, \pi_T=0.9$)	0.145	0.280	0.605
Polynomial SVM($C=5*10^{-5}, c=15, d=2$)	0.204	0.481	0.642
RBF SVM($C=10^{-1}, \gamma=10^{-3}$)	0.169	0.249	0.709
GMM Full-Cov, 16-G	0.123	0.230	0.525
GMM Diag-Cov, 32-G	0.147	0.279	0.592
GMM Tied Full-Cov, 64-G	0.130	0.231	0.590
Z-Normalized Features - PCA (m = 7)			
MVG Full-Cov	0.139	0.295	0.574
MVG Diag-Cov	0.200	0.528	0.749
MVG Tied Full-Cov	0.112	0.212	0.570
Log Reg($\lambda=10^{-4}, \pi_T=0.5$)	0.108	0.201	0.546
Log Reg($\lambda=10^{-4}, \pi_T=0.1$)	0.107	0.199	0.533
Log Reg($\lambda=10^{-4}, \pi_T=0.9$)	0.111	0.205	0.555
Linear SVM($C=10^{-2}$, unbalanced)	0.157	0.319	0.601
Linear SVM($C=10^{-3}, \pi_T=0.5$)	0.150	0.290	0.598
Linear SVM($C=6*10^{-3}, \pi_T=0.1$)	0.144	0.289	0.604
Linear SVM($C=7*10^{-4}, \pi_T=0.9$)	0.146	0.289	0.610
Polynomial SVM($C=5*10^{-5}, c=15, d=2$)	0.226	0.594	0.643
RBF SVM($C=10^{-1}, \gamma=10^{-3}$)	0.159	0.250	0.660
GMM Full-Cov, 16-G	0.134	0.293	0.600
GMM Diag-Cov, 32-G	0.121	0.226	0.588
GMM Tied Full-Cov, 64-G	0.131	0.231	0.603

Table 11: Min DCF over test set for all models trained over all training set.

making the performance worse.

We can also verify if the chosen hyper-parameters provide close to optimal results on the evaluation set. In this way, we can understand if we take the correct values for the hyper-parameters.

From Figure 12 to Figure 14, we can see the *min DCF* for different values of the hyper-parameters on the evaluation set and validation set.

As we can see, the value we choose for the hyper-parameter λ and C were indeed effective (Figure 12 and Figure 13).

From Figure 14, we can see that the optimal number of components is similar between evaluation and validation set:

1. For Full-Cov model, the 8-G component model is slightly better, but the result of the value we choose 16-G component is very close (*min DCF* of 0.123 vs 0.110)

2. For Diag-Cov model, the 8-G component model is slightly better, but the result of the value we choose 32-G component is very close (*min DCF* of 0.147 vs 0.131)

3. For Tied Full-Cov model, the 32-G component model is slightly better, but the result of the value we choose 64-G component is very close (*min DCF* of 0.130 vs 0.129)

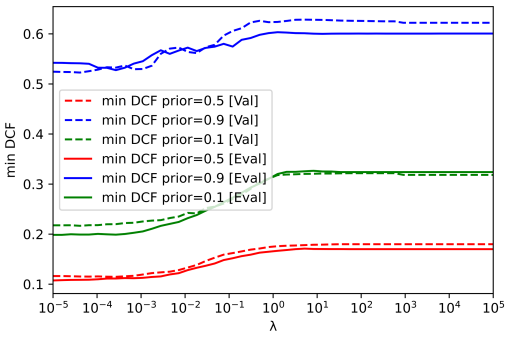
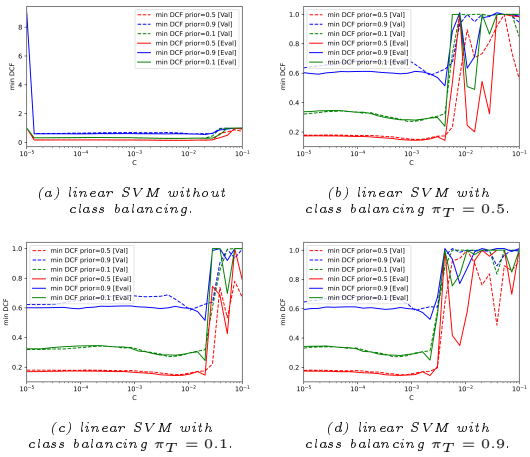


Figure 12: Min DCF for different values of λ on the evaluation and validation set.



(a) linear SVM without class balancing. *(b) linear SVM with class balancing $\pi_T = 0.5$.*

(c) linear SVM with class balancing $\pi_T = 0.1$. *(d) linear SVM with class balancing $\pi_T = 0.9$.*

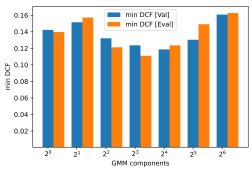
Figure 13: Min DCF for different values of C on the evaluation and validation set.

From Figure 15, we can see that the three models have similar performances and their curves show a high slope in the left part. This means that we can have a high number of true positives (correctly classified *true pulsar signals*) keeping lower the number of false positives (misclassified *false pulsar signals*).

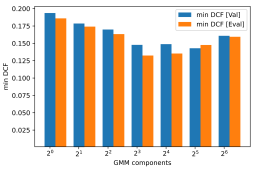
6 Conclusions

We have seen that linear models perform better than quadratic ones on the HTRU2 dataset. We notice that the MVG Tied Full-Cov (PCA $m = 7$) and the Logistic Regression (PCA $m = 7$) with $\lambda = 10^{-4}$ and $\pi_T = 0.5$ are the models that classify better both training and evaluation set.

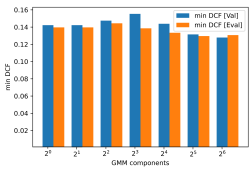
Thanks to the choices of the different hyper-parameters, we result with a *min DCF* ≈ 0.1



(a) GMM Full-Cov



(b) GMM Diag-Cov



(c) GMM Tied Full-Cov

Figure 14: Min DCF for different values of components on the evaluation and validation set.

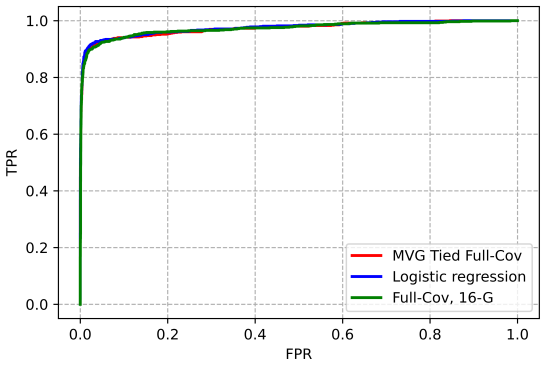


Figure 15: ROC between most promising model.

for the target application ($\widetilde{\pi} = 0.5$). We achieve even discrete *min DCF* of ≈ 0.2 with $\widetilde{\pi} = 0.1$ and ≈ 0.5 for $\widetilde{\pi} = 0.9$ applications.

The choices we made on our training/validation sets proved to be effective also for the evaluation set.

References

[1] R. J. Lyon, HTRU2, https://figshare.com/articles/dataset/HTRU2/3080389/1

[2] M. J. Keith et al., The high time resolution universe pulsar survey - i. system configuration and initial discoveries. Monthly Notices of the Royal Astronomical Society, vol. 409, pp. 619-627, 2010 https://doi.org/10.1111/j.1365-2966.2010.17325.x

[3] Cheng Jun Zhang et al., A Review of Research on Pulsar Candidate Recognition Based on Machine Learning