Pulsar detection project report
MLPR a.y. 2022/2023 - 31 January 2023

Andrea Taurino
Politecnico di Torino
s281605@studenti.polito.it

1 Introduction

HTRU2 is a dataset which describes a sample of pulsar candidates collected during the High Time Resolution Universe Survey (South).

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.

In this project we're going to use and compare the performance of different Machine Learning algorithms in order to facilitate a rapid analysis of data by labeling the pulsar candidates through classification systems.

1.1 Dataset features description

The dataset contains 16,259 spurious samples caused by RFI/noise, and 1,639 real pulsar samples. These samples are binary labeled with 0 (negative, spurious non-pulsar) and 1 (positive, real pulsar).
In total, the dataset stores 17,898 samples and is divided in two parts: the training set contains 8929 samples (8108 negative, 821 positive samples) and the test set 8969 samples (8151 negative, 818 positive samples)
In our analysis we'll keep the train/evaluation split described above.

Each sample is described by 8 continuous features, the first four are statistics obtained from the integrated pulse profile while the last four are statistics computed on the DM-SNR curve.
Each feature is described 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.

2 Dataset analysis

Before analyzing the dataset, a z-normalization process was applied. This process allows to center and scale the features to unit variance. Reducing the features' variance allows better handling from a numeric point of view by simplifying computing and reducing the risks of overflow.

$$Z_i = \frac{X_i - \mu}{\sigma} \tag{1}$$

The equation 1 explains the z-normalization process: the sample X_i is centered by subtracting from its features the mean of each feature computed on all the samples μ and then it's normalized by dividing it by the standard deviation of the features of the dataset σ .

In Figure 1, we can see the histograms of all the z-normalized features of our dataset. It's also immediate to see that there's no need for further preprocessing - such as Gaussianization - since there are no evident outliers in the features shown in the plot.

The next step is analyzing the correlation between features in order to understand whether dimensionality reduction tecniques - such as PCA - can be applied to the dataset.
The **heatmap** in Figure 2 shows the correlations between features plotted by exploiting the absolute value of the Pearson correlation coefficient:

$$correlation = \left| \frac{Cov(X,Y)}{\sqrt{Var(X)}\sqrt{Var(X)}} \right| \tag{2}$$

A correlation coefficient close to 1 - shown here in darker colour - means strong correlation between features. From the data gathered in Figure 2, considering the features from 0 to 7, we can clearly see that features 2-3, 6-7 and 0-2 are strongly correlated. This suggests we should try to map the data up to 5 uncorrelated features in order to reduce the number of parameters to estimate.

In order to achieve this result we'll use PCA, a dimensionality reduction tecnique that reduces the dimensional feature space of the dataset from n-dimensional to m-dimensional ($m << n$) by focusing on the directions with the highest variance. As said before, we'll use PCA to reduce dimensions up to $m = 5$ and, because of the strong correlation between some of the features, we don't expect a huge loss in the minDCF metric that we'll use to evaluate the classifier.

The minDCF metric measures the cost we would pay if we made optimal decisions for the validation set (in this case, it's extracted from the training set) using the scores computed by the classifier.
We'll consider three applications: our main application, the uniform

prior one [3], and two applications with the prior biased towards negative [4] and positive class [5].

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

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

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

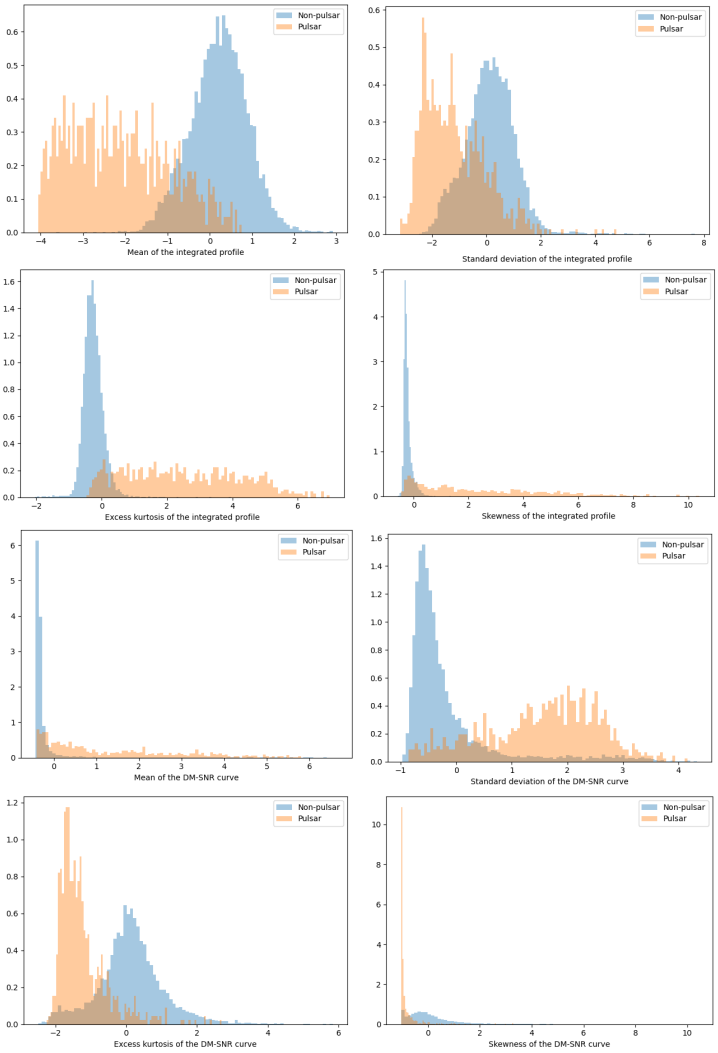

Figure 1: Histogram of the features of the z-normalized training set.

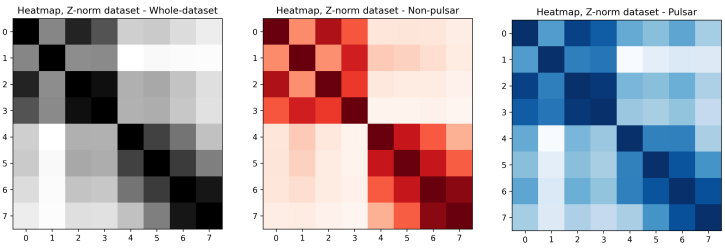

Figure 2: Heatmap of the z-normalized dataset and among the negative and positive class

3 HTRU2 classification

In order to perform the performance analysis of the different classification models, the K-fold cross validation method was chosen. The reason of this choice is the reasonable size of the dataset that allowed computations in reasonable time while, at the same time, guaranteeing a more solid analysis if compared to a single split approach because of the dataset unbalancing caused by the $\frac{\text{numPositiveSamples}}{\text{numNegativeSamples}} = \frac{1}{10}$ ratio.
The K-fold cross validation method consists in splitting the training set in K smaller sets and using, at each time, K - 1 sets for training and the remaining set for validation. This algorithm is executed K times and the training and validation sets that are generated are always different. In this analysis, before computing the K-folds, the dataset has been shuffled randomly. Nevertheless, when approaching the model evaluation step, the K-fold cross validation allows us to train the *final* model on the entire dataset before testing it on a separate testing set.
In the following performance analysis we'll use K-fold with $k = 3$

3.1 Gaussian classifier

We now consider the gaussian classifiers, in specific we'll consider the MVG (Multivariate Gaussian Classifier).
These family of classifiers lean on the assumption that the data, given the class c, is gaussian-distributed:

$$\boldsymbol{X}|C = c \sim \mathcal{N}(\mu_c, \Sigma_c) \tag{6}$$

We consider four kinds of gaussian classifiers:

- Full covariance MVG, where each class has its own covariance matrix Σ_c
- Tied covariance MVG, where there's a unique covariance matrix for all the classes $\Sigma_c = \Sigma$ computed on the whole dataset
- Naive Bayes MVG, where the Naive Bayes assumption implies diagonal covariance matrix Σ_c
- Naive Bayes tied covariance MVG classifier, where the covariance matrix $\Sigma_c = \Sigma$ is also diagonal

Given the considerations done given the heatmap in Figure 2, it's unlikely that the Naive Bayes diagonal covariance MVG models will perform better or equal than the others, because a diagonal Σ implies zero correlation among all the features and this is untrue given the considerations done in Section 2 and Figure 2.
Nevertheless, the full and tied covariance MVG models should perform better given their ability to take into account the covariance between features and the dataset's correlations shown in the heatmap.

As we can see from the results in Table 3.1, in all the scenarios the Naive Bayes diagonal MVG models tend to perform worse than their full and tied counterparts, this proves the point expressed before. Dimensionality reduction using PCA performs quite well when $m = 7$ and it's possible to see that the best models, Full and Tied-Full covariance, perform overall equal or better then when they are applied to the full data.
Nevertheless, all models' performance drops significantly when $m < 7$ and, for this reason, we'll drop the models reduced with PCA $m < 7$ for the following analysis.

Going back to the analysis of the best model we can clearly see that the best performances are offered by the Tied Full Covariance MVG. This probably means that the data we're classifing might return better results with linear classifiers such as the Tied MVG.

We now turn our attention to discriminative approaches.

Table 3.1: MVG classifier - minDCF on the validation set

	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$
		no PCA	
Full-Cov	0.142	0.289	0.659
Diag-Cov	0.192	0.316	0.734
Tied-Full-Cov	0.112	0.226	0.570
Tied-Diag-Cov	0.160	0.266	0.579
		PCA m = 5	
Full-Cov	0.149	0.256	0.651
Diag-Cov	0.220	0.462	0.749
Tied-Full-Cov	0.150	0.264	0.565
Tied-Diag-Cov	0.171	0.313	0.593
		PCA m = 6	
Full-Cov	0.152	0.292	0.636
Diag-Cov	0.222	0.532	0.728
Tied-Full-Cov	0.138	0.255	0.572
Tied-Diag-Cov	0.163	0.302	0.600
		PCA m = 7	
Full-Cov	0.139	0.307	0.627
Diag-Cov	0.214	0.509	0.718
Tied-Full-Cov	0.112	0.224	0.567
Tied-Diag-Cov	0.140	0.274	0.597

3.2 Linear logistic regression

Logistic regression is a discriminative model for classification where instead of modeling the distribution of the observed samples $\mathbf{X}|C$ it directly models the class posterior distribution $\mathbf{C}|X$
In the following analysis we'll use a *balanced* version of the regularized logistic regression where the cost of the different classes have been re-balanced by minimizing the objective function:

$$J(\boldsymbol{w}, b) = \frac{\lambda}{2}\|\boldsymbol{w}\|^2 + \frac{\pi_T}{n_T}\sum_{i=1|c_i=1}^{n} \log\left(1 + e^{-z_i(\boldsymbol{w}^T \boldsymbol{x_i} + b)}\right) + \frac{1-\pi_T}{n_F}\sum_{i=1|c_i=0}^{n} \log\left(1 + e^{-z_i(\boldsymbol{w}^T \boldsymbol{x_i} + b)}\right) \tag{7}$$

We'll consider $\pi_T = 0.5$ balanced Logistic Regression as our main application target, even though this corresponds, to an appropriate degree of approximation, to the unbalanced Logistic Regression. By using the K-fold cross-validation and the minDCF metric we're going to estimate the *regularization coefficient* hyperparameter λ by looking at the resulting plots in Figure 3. The model behaves quite well in the region with λ between 10^{-5} and 10^{-4}.

In order to choose the optimal hyperparameter λ we have to consider the implications of this choice in the Linear Logistic Regression model:

- $\lambda \simeq \mathbf{0}$ could return good predictions during training but worse results when predicting labels on unseen data (bad generalization, overfitting)
- $\lambda >> \mathbf{0}$ we obtain a small norm $||w||^2$, but poor ability to separate the classes

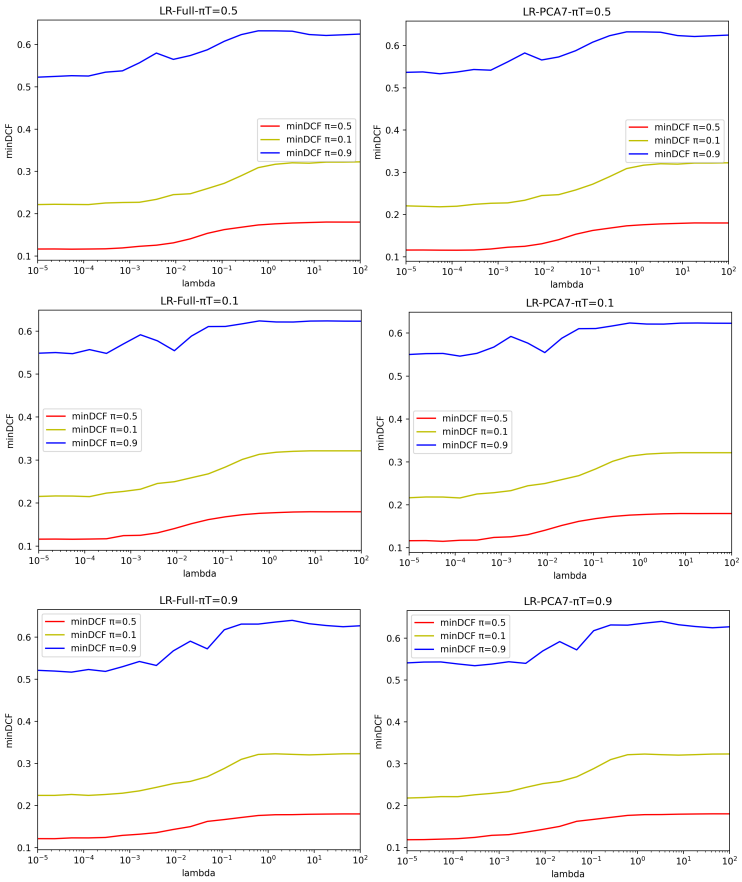

Figure 3: Plots of the minDCF for the Logistic Regression model. Full dataset on the left, PCA = 7 on the right. From top to bottom $\pi_T = 0.5$, $\pi_T = 0.1$, $\pi_T = 0.9$

Given the considerations done above on λ and the minDCF results in Figure 3, choosing $\lambda = 10^{-4}$ for our linear logistic regression model might be the right decision for both the PCA m = 7 data and the full data.
In Table 3.2 we can see the results for the linear logistic regression with $\lambda = 10^{-4}$ and the results are coherent with the hypothesis formulated for the MVG classifier, where we hypothesized good performances for linear classifiers.
In the next sections concerning Quadratic SVM we'll see if this hypothesis still holds.
The results here show a good performance for both the balanced applications $\pi_T = 0.5$ and $\pi_T = 0.1$ in both the scenarios with PCA m = 7 and without PCA. Yet, considering our main application $\tilde{\pi} = 0.5$, the best results are obtained using PCA with m = 7 with balancing $\pi_T = 0.5$

Table 3.2: Linear logistic regression classifier minDCF metrics

	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$
	no PCA	$\lambda = 10^{-4}$	
Linear LogReg ($\pi_T = 0.5$)	0.116	0.222	0.527
Linear LogReg ($\pi_T = 0.1$)	0.116	0.215	0.546
Linear LogReg ($\pi_T = 0.9$)	0.123	0.225	0.522
	PCA m = 7	$\lambda = 10^{-4}$	
Linear LogReg ($\pi_T = 0.5$)	0.115	0.219	0.537
Linear LogReg ($\pi_T = 0.1$)	0.117	0.216	0.555
Linear LogReg ($\pi_T = 0.9$)	0.120	0.221	0.538

3.3 Support Vector Machine

Up to now we've seen that linear classifiers return good prediction when used on this dataset, we'll see if the analysis of the Support Vector Machine model will either confirm or disprove this hypothesis.

In order to perform the analysis on Support Vector Machine models we'll employ three kinds of SVM classifiers:

- linear
- quadratic
- RBF - Radial Basis Function

We'll also perform the analysis on both unbalanced and balanced SVM. We start by considering the linear model.

3.3.1 Linear SVM

In order to solve the linear SVM problem we'll consider the dual formulation:

$$J^D(\alpha) = -\frac{1}{2}\alpha^T H\alpha + \alpha^T 1 \tag{8}$$

where α_i is constrained between values $0 \leq \alpha_i \leq C$ and C is an hyper-parameter to be selected using cross-validation and $\boldsymbol{H}$ is the following matrix

$$\boldsymbol{H}_{i,j} = z_i z_j \boldsymbol{x}_i^T \boldsymbol{x}_j \tag{9}$$

where $z_i = +1$ is the class label for the positive samples and $z_i = -1$ is the label for the negative samples

In order to implement the class balancing in SVM we tweak the hyperparameter C in the following way: $C_T = C\frac{\pi_T}{\pi_T^{emp}}$ for the positive samples and $C_F = C\frac{\pi_F}{\pi_F^{emp}}$ for the negative samples

By looking at the plots in Figure 4 we can see that, for all the models, a good choice for hyperparameter C might be between 10^{-1} and 0; for this reason we selected $C = 5 * 10^{-1}$.

From the results showed in Table 3.3 we can see that the PCA reduction, again, doesn't reduce the quality of the classification.
Nevertheless, if we focus on our main application $\tilde{\pi} = 0.5$ the class balancing doesn't deeply affect the results for $\pi_T = 0.1$ and $\pi_T = 0.5$ while the same doesn't apply for $\pi_T = 0.9$.
We'll consider as best model the linear SVM with $\pi_T = 0.5$ and no PCA because of its slightly better behaviour in the application $\tilde{\pi} = 0.9$ if compared to the other linear SVM results.

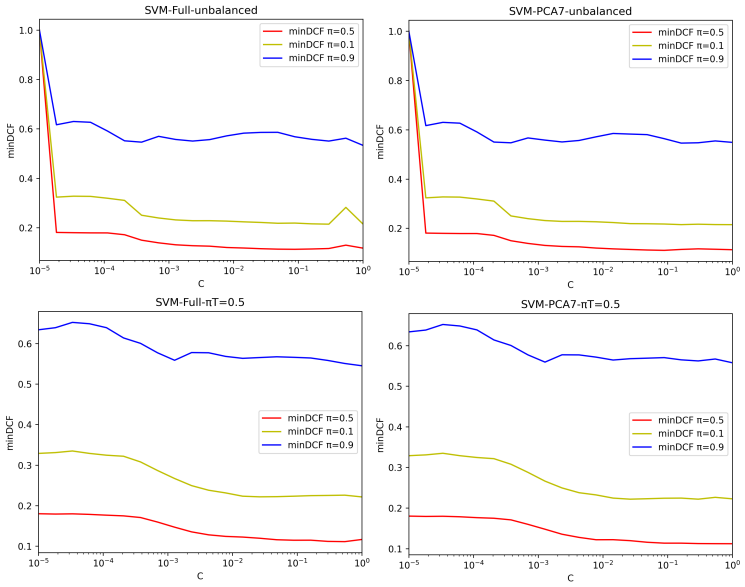

Figure 4: Plots of the minDCF for the linear SVM performed on the full dataset (left) and PCA m = 7 (right). Unbalanced results on top and balanced with $\pi_T = 0.5$ at the bottom

Table 3.3: Linear SVM classifier minDCF with parameter $C = 5 * 10^{-1}$

	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$
	no PCA		
LinSVM (unbalanced)	0.117	0.217	0.554
LinSVM ($\pi_T = 0.5$)	0.113	0.224	0.548
LinSVM ($\pi_T = 0.1$)	0.115	0.217	0.556
LinSVM ($\pi_T = 0.9$)	0.127	0.220	0.526
	PCA m = 7		
LinSVM (unbalanced)	0.118	0.217	0.575
LinSVM ($\pi_T = 0.5$)	0.114	0.229	0.562
LinSVM ($\pi_T = 0.1$)	0.113	0.216	0.559
LinSVM ($\pi_T = 0.9$)	0.124	0.223	0.545

3.3.2 Quadratic and RBF SVM

In order to prove the hypothesis concerning the better performances offered by the linear classifiers we employ two non-linear SVM formulations.
In order to obtain non-linearity we expand, implicitly, the features in a higher dimensional space through two different kernel functions:

- $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\boldsymbol{x}_i^T \boldsymbol{x}_j + c)^d$ with $d = 2$ for the quadratic SVM
- $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = e^{-\gamma \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}$ for the RBF (Radial Basis Function) SVM

We'll now start analyzing the quadratic SVM model with the selection of the hyperparameters c and C.

If we look at the plots in Figure 5 it's quite hard to decide which couple of hyperparameters should be selected, but we'll try

to focus on the ones that employ the lowest minDCF and are more stable.

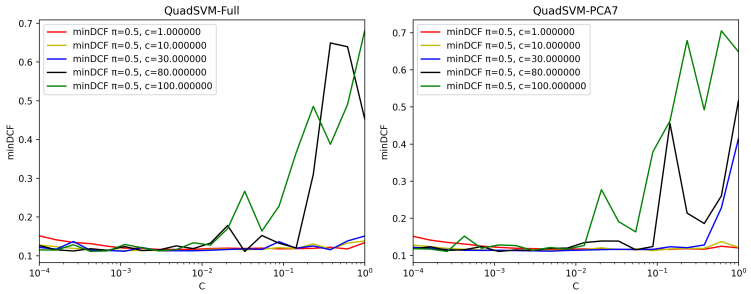

Figure 5: Plots of the minDCF for the quadratic kernel SVM with $\pi_T = 0.5$ performed on the full dataset (right) and PCA with m = 7 (left)

In the end the hyperparameters C = 10^{-3} and c = 10 are selected and the relative results are analyzed in the following table.

Table 3.4: Quadratic SVM classifier minDCF with parameters $C = 10^{-3}$, $c = 10$

	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$
		no PCA	
Quad-SVM (unbalanced)	0.114	0.213	0.602
Quad-SVM ($\pi_T = 0.5$)	0.114	0.229	0.540
Quad-SVM ($\pi_T = 0.1$)	0.112	0.216	0.582
Quad-SVM ($\pi_T = 0.9$)	0.148	0.272	0.576
		PCA m = 7	
Quad-SVM (unbalanced)	0.113	0.216	0.615
Quad-SVM ($\pi_T = 0.5$)	0.114	0.229	0.549
Quad-SVM ($\pi_T = 0.1$)	0.113	0.217	0.585
Quad-SVM ($\pi_T = 0.9$)	0.150	0.272	0.576

Surprisingly, the formulated hypothesis doesn't hold and the quadratic SVM performances are equal and in some cases even better than the linear models discussed until now.
Similarly to what happened in the case of the linear SVM, balancing provides small improvements for the main application $\tilde{\pi} = 0.5$ when $\pi_T = 0.5$ and $\pi_T = 0.1$, while performance slightly decreases for $\pi_T = 0.9$.
The same considerations apply to both no PCA and PCA with m = 7 scenarios, nevertheless PCA with m = 7 provides similar results to the full dimensional data.
We'll consider the best result with no PCA applied when balanced with $\pi_T = 0.1$ because provides the best performance on the main application $\tilde{\pi} = 0.5$, but it's also good when considering the other non-main applications.

Let's now consider the case of the RBF kernel SVM classifier.
The first step is, again, selecting the appropriate hyperparameters by looking at the plots in Figure 6
In this case, we'll select $C = 5 * 10^{-1}$ and $\gamma = 10^{-2}$.

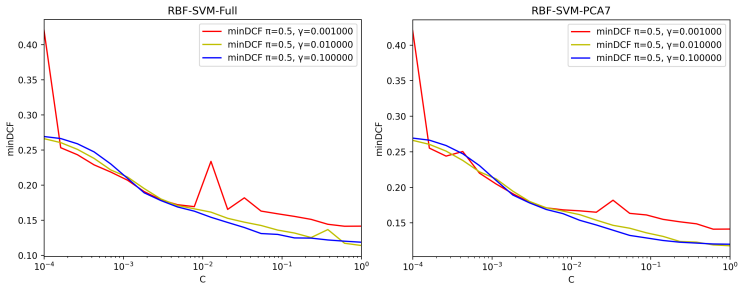

Figure 6: Plots of the minDCF for RBF kernel SVM performed on the full dataset and PCA with m = 7 and $\pi_T = 0.5$

Table 3.5: RBF kernel SVM classifier minDCF with parameters $C = 5 * 10^{-1}$ and $\gamma = 10^{-2}$

	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$
		no PCA	
RBF-SVM (unbalanced)	0.121	0.225	0.583
RBF-SVM ($\pi_T = 0.5$)	0.118	0.232	0.563
RBF-SVM ($\pi_T = 0.1$)	0.119	0.225	0.585
RBF-SVM ($\pi_T = 0.9$)	0.156	0.287	0.604
		PCA m = 7	
RBF-SVM (unbalanced)	0.121	0.227	0.585
RBF-SVM ($\pi_T = 0.5$)	0.118	0.229	0.605
RBF-SVM ($\pi_T = 0.1$)	0.119	0.226	0.586
RBF-SVM ($\pi_T = 0.9$)	0.157	0.294	0.612

The following results in Table 3.5 show quite good results but yet worse performance if compared to the quadratic and linear SVM models, yet we can still see that there are again no significant differences between the no PCA case and the data with PCA m = 7. The performance seems to be overall worse than the models seen until now and we will not consider this model in the next steps of our analysis.

3.4 Gaussian Mixture Models

We now approach the final model in our analysis: the Gaussian Mixture Model.
This generative model works by approximating generic distributions using a specific number of multiple gaussian distributions. This number will be the hyperparameter to estimate. We will use three different versions of GMM: full covariance, tied covariance and diagonal covariance matrix.

Using the hyperparameters gathered from the plots in Figure 7, we can see in Table 3.6 that this model performs far worse than the ones analyzed before, yet the performance is still approximately the same between the full data and the PCA with m = 7 data.

For these reasons we discard this model for our following analysis.

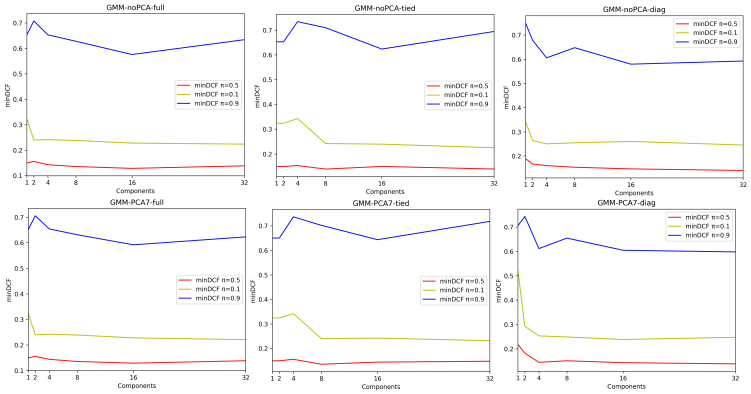

Figure 7: Plots of the minDCF for GMM performed on the full dataset (top) and PCA with m = 7 (bottom) for full covariance model (left), tied model (centre), diagonal model (right)

Table 3.6: GMM classifier minDCF with parameters EM-algorithm-stop = 10^{-6}, $\psi = 0.1$, $\alpha = 0.1$

	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$
		no PCA	
fullCov-GMM (16 comp.)	0.129	0.228	0.576
diagCov-GMM (16 comp.)	0.147	0.260	0.580
tiedCov-GMM (8 comp.)	0.139	0.242	0.709
		PCA m = 7	
fullCov-GMM (16 comp.)	0.129	0.228	0.592
diagCov-GMM (4 comp.)	0.144	0.253	0.612
tiedCov-GMM (8 comp.)	0.135	0.240	0.702

4 Score calibration

Up to now we have only considered the minimum DCF metrics.
The minDCF measures the cost we would pay if we made optimal decisions for the evaluation set using the recognizer scores. The cost we will actually pay, however, depends on the goodness of the decisions we make using those scores. We therefore turn our attention to the actual DFC.
If scores are well calibrated, the optimal threshold that optimizes the Bayes risk is $t = -log\frac{\tilde{\pi}}{1-\tilde{\pi}}$, so computing the actDCF will show how good the model is when using the theoretical threshold.

We will compute the actDCF scores for the following best performing models:

- no PCA:
  - quadratic SVM with hyperparameters $C = 10^{-3}$, $c = 10$ and $\pi_T = 0.1$ balancing
  - linear SVM with hyperparameter $C = 5 * 10^{-1}$ and $\pi_T = 0.5$ balancing
- PCA with m = 7:
  - tied full covariance matrix MVG
  - linear logistic regression with hyperparameter $\lambda = 10^{-4}$ and $\pi_T = 0.5$ balancing

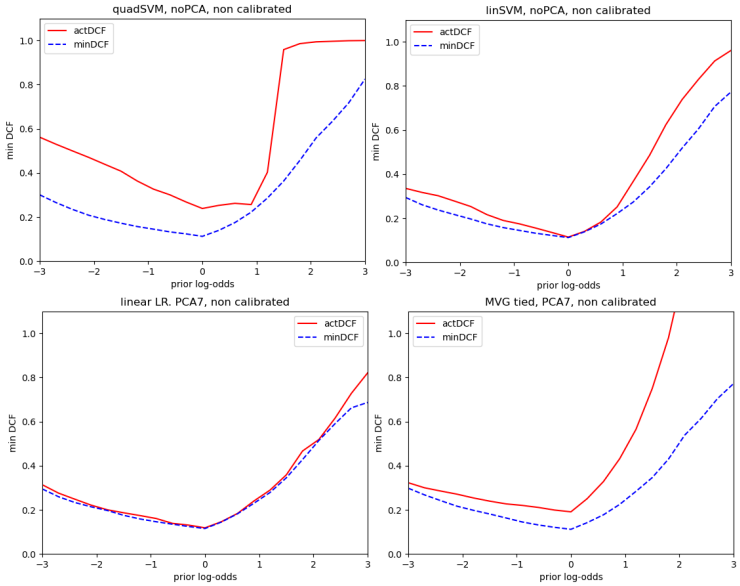

Figure 8: Bayes error plots of the best performing models, non calibrated

As we can see from the Bayes error plots represented in Figure 8, the linear logistic regression is the only model that is already well calibrated on all the different applications and doesn't seem to require score recalibration, while all the other models' minDCF is quite far from the corresponding actDCF curve and thus require score recalibration.
In Table 4.1 we can observe that, for our main application $\tilde{\pi} = 0.5$, the actDCF is $\approx 110\%$ higher than the minDCF for the quadratic SVM and the tied full covariance MVG has $\approx 70\%$ higher actDCF.
The linear SVM actDCF is acqually quite close the minDCF in the $\tilde{\pi} = 0.5$ application, while starts getting higher in the other applications, with $\approx 60\%$ higher actDCF for the $\tilde{\pi} = 0.9$ application.
The values in Table 4.1 also confirm the plot clearly showing that the linear logistic regression model is already well calibrated and doesn't need to be recalibrated.

Table 4.1: Uncalibrated actDCF for the best performing models

	$\tilde{\pi} = 0.5$		$\tilde{\pi} = 0.1$		$\tilde{\pi} = 0.9$	
no PCA	minDCF	actDCF	minDCF	actDCF	minDCF	actDCF
quadratic SVM	0.112	0.239	0.216	0.482	0.582	0.996
linear SVM	0.113	0.115	0.224	0.285	0.548	0.781
PCA m = 7	minDCF	actDCF	minDCF	actDCF	minDCF	actDCF
tied full-cov MVG	0.112	0.191	0.224	0.274	0.567	1.422
linear LR	0.115	0.118	0.219	0.229	0.537	0.547

We now proceed to perform the score recalibration on all these models, including the linear logistic regression.

In order to achieve the calibrated scores, we'll use an approach which consistis in computing a linear transformation function $f(s) = \alpha s + \beta$ that maps the scores s of each model to a well calibrated score $s_{calibrated} = f(s)$.
Since $f(s)$ should produce well-calibrated scores, it can be interpreted as the log-likelihood ratio for the two class hypotheses:

$$f(s) = log\frac{f_{S|C}(s|H_T)}{f_{S|C}(s|H_F)} = \alpha s + \beta \tag{10}$$

So the class posterior probability for a prior $\tilde{\pi}$ corresponds to:

$$log\frac{P(C = H_T|s)}{P(C = H_F|s)} = \alpha s + \beta + log\frac{\tilde{\pi}}{1-\tilde{\pi}} \tag{11}$$

We can interpret the scores s as features, so that they have a similar expression as the log posterior ratio of the logistic regression model.
If we let

$$\beta' = \beta + log\frac{\tilde{\pi}}{1-\tilde{\pi}} \tag{12}$$

then we have exactly the same model, thus we can employ the prior-weighed logistic regression model to learn the parameters α and β' over our training scores.
Then, to recover the calibrated scores, we'll need to compute

$$f(s) = \alpha s + \beta = \alpha s + \beta' - log\frac{\tilde{\pi}}{1-\tilde{\pi}} \tag{13}$$

In our case, we'll employ the main application prior $\tilde{\pi} = 0.5$ for the score recalibration and we'll use the hyperparameter $\lambda = 10^{-4}$ for the logistic regression that computes α and β'.

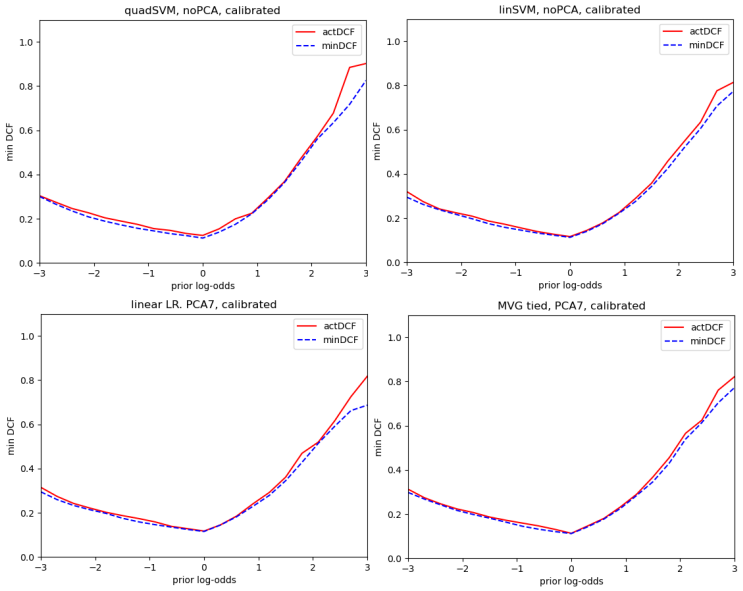

Figure 9: Bayes error plots of the best performing models, calibrated

As we can see in Figure 9 the score recalibration has proven to be quite effective for all the models except, as we expected, the linear logistic regression.
We can also see that, despite choosing $\tilde{\pi} = 0.5$, the other applications with different priors $\tilde{\pi} = 0.1$ and $\tilde{\pi} = 0.9$ have also benefited from the score recalibration.

Table 4.2: Calibrated actDCF for the best performing models

	$\tilde{\pi} = 0.5$		$\tilde{\pi} = 0.1$		$\tilde{\pi} = 0.9$	
no PCA	minDCF - actDCF					
quadratic SVM	0.112	0.124	0.216	0.233	0.582	0.586
linear SVM	0.113	0.116	0.224	0.228	0.548	0.567
PCA m = 7	minDCF - actDCF					
tied full-cov MVG	0.112	0.113	0.224	0.220	0.567	0.591
linear LR	0.115	0.117	0.219	0.227	0.537	0.547

After computing the calibrated actDCF values, we can see in Table 4.2 that indeed all the models have benefited from the score recalibration, including the minimal but negligible improvements in the linear regression model and especially the tied full covariance MVG, where the calibrated actDCF regarding $\tilde{\pi} = 0.5$ is almost equal to the minDCF.

5 Experimental results

In this section we will provide the results obtained by the different models on the test dataset.
Since we used k-fold cross validation with k = 3 during our previous analysis, we are allowed to train again the same models on the entire training set.
As we've done in the previous experiments, we'll apply Z-normalization and PCA on the training set and also on the test set using the parameters learned during training.

Table 5.1: Non calibrated actDCF on test set for the best performing models

	$\tilde{\pi} = 0.5$		$\tilde{\pi} = 0.1$		$\tilde{\pi} = 0.9$	
no PCA	minDCF - actDCF					
quadratic SVM	0.108	0.235	0.201	0.502	0.497	0.995
linear SVM	0.106	0.112	0.201	0.291	0.564	0.806
PCA m = 7	minDCF - actDCF					
tied full-cov MVG	0.110	0.202	0.208	0.277	0.587	1.338
linear LR	0.109	0.113	0.202	0.224	0.538	0.566

The following results in Table 5.1 refer to the models discussed before:

- no PCA:
  - quadratic SVM with hyperparameters $C = 10^{-3}$, $c = 10$ and $\pi_T = 0.1$ balancing
  - linear SVM with hyperparameter $C = 5 * 10^{-1}$ and $\pi_T = 0.5$ balancing
- PCA with m = 7:
  - tied full covariance matrix MVG

– linear logistic regression with hyperparameter $\lambda = 10^{-4}$ and $\pi_T = 0.5$ balancing

and the values are consistent with the minDCF values obtained during the validation step.
The best performance is reached with the linear SVM model while also linear logistic regression, quadratic SVM and tied full covariance MVG show better performance if compared to the validation step.

Again, regarding the actDCF scores, we are in the same exact situation as before during the validation step, LR is once again the only model that doesn't need recalibration, while the others do.
Therefore we go on applying score recalibration (Figure 5.2) on the obtained scores on the test set.

Table 5.2: Calibrated actDCF on test set for the best performing models

	$\tilde{\pi} = 0.5$		$\tilde{\pi} = 0.1$		$\tilde{\pi} = 0.9$	
no PCA	minDCF - actDCF					
quadratic SVM	0.108	0.119	0.201	0.215	0.497	0.515
linear SVM	0.106	0.111	0.201	0.222	0.564	0.574
PCA m = 7	minDCF - actDCF					
tied full-cov MVG	0.110	0.115	0.208	0.221	0.587	0.611
linear LR	0.109	0.112	0.202	0.221	0.538	0.566

As we can see all the classifiers benefit from the score recalibration on all the applications with, once again, the exception of the linear logistic regression that obtains slightly, but negligible, worse actDCF scores on $\tilde{\pi} = 0.5$ application.

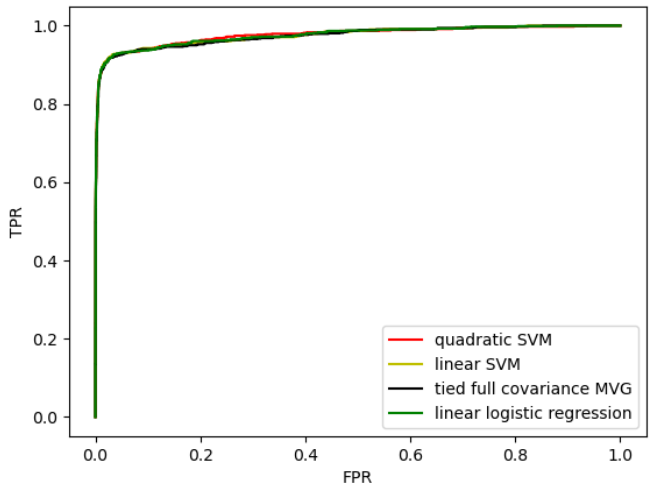

Figure 10: ROC plot of the best performing models

In the end, in Figure 10, in order to further prove the goodness of our selected models, we compute the ROC plot and we obtain good and similar results for all the selected classifiers, showing almost identical area under curve.

All the models have a high slope at the left of the curve, meaning a good ability to correctly classify positive samples (pulsar signals) while keeping the false positive samples at a minimum.

5.1 Experimental results on all the classifiers

In the following section we show the evaluation results on all the classifiers tested thoughout this analysis. As we can see, the models we selected before, perform better than the others, accordingly to our analysis.
However, some results might slightly change between no PCA and PCA with m = 7, and be slightly better than the results obtained on the best classifiers in Table 5.1.
The only surprise, in Tables 6.1 and 6.2, is the good performance of the RBF Support Vector Machine classifier.

6 Conclusions

During our analysis we hypothesized that linear classifiers could return better results than the others, this hypothesis has been disproved because of the very good results reached by the quadratic Support Vector Machine.
However, the similarity between validation and evaluation results suggest that the training and testing set might have similar distribution.
Nevertheless we were able to reach a actDCF of ≈ 1 for the target application $\tilde{\pi} = 0.5$ with also good actDCFs for $\tilde{\pi} = 0.1$ and $\tilde{\pi} = 0.9$.
Overall, given the performance metrics obtained on the test set regarding the selected best classifier we can be confident of the good quality of the choices made during our analysis.

Table 6.1: Experimental results with minDCF metric on all the models - no PCA

	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$
no PCA			
Full-Cov-MVG	0.140	0.283	0.646
Diag-Cov-MVG	0.185	0.330	0.621
Tied-Full-Cov-MVG	0.110	0.207	0.591
Tied-Diag-Cov-MVG	0.152	0.262	0.544
Linear LR ($\pi_T = 0.5, \lambda = 10^{-4}$)	0.110	0.199	0.534
Linear LR ($\pi_T = 0.1, \lambda = 10^{-4}$)	0.110	0.197	0.534
Linear LR ($\pi_T = 0.9, \lambda = 10^{-4}$)	0.115	0.206	0.510
LinSVM (unbalanced)	0.113	0.201	0.552
LinSVM ($\pi_T = 0.5, C = 5 * 10^{-1}$)	0.106	0.201	0.561
LinSVM ($\pi_T = 0.1, C = 5 * 10^{-1}$)	0.111	0.201	0.554
LinSVM ($\pi_T = 0.9, C = 5 * 10^{-1}$)	0.116	0.206	0.499
Quad-SVM (unbalanced)	0.108	0.199	0.497
Quad-SVM ($\pi_T = 0.5, 10^{-3}, 10$)	0.107	0.210	0.465
Quad-SVM ($\pi_T = 0.1, 10^{-3}, 10$)	0.108	0.201	0.497
Quad-SVM ($\pi_T = 0.9, 10^{-3}, 10$)	0.127	0.239	0.546
RBF-SVM (unbalanced)	0.113	0.212	0.531
RBF-SVM ($\pi_T = 0.5, 5 * 10^{-1}, 10^{-2}$)	0.111	0.214	0.544
RBF-SVM ($\pi_T = 0.1, 5 * 10^{-1}, 10^{-2}$)	0.113	0.211	0.529
RBF-SVM ($\pi_T = 0.9, 5 * 10^{-1}, 10^{-2}$)	0.144	0.267	0.546
fullCov-GMM (16 comp.)	0.128	0.216	0.617
diagCov-GMM (16 comp.)	0.137	0.240	0.592
tiedCov-GMM (8 comp.)	0.134	0.249	0.659

Table 6.2: Experimental results with minDCF metric on all the models - PCA = 7

	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$
PCA m = 7			
Full-Cov-MVG	0.139	0.293	0.574
Diag-Cov-MVG	0.201	0.514	0.755
Tied-Full-Cov-MVG	0.110	0.208	0.587
Tied-Diag-Cov-MVG	0.141	0.257	0.556
Linear LR ($\pi_T = 0.5, \lambda = 10^{-4}$)	0.109	0.199	0.538
Linear LR ($\pi_T = 0.1, \lambda = 10^{-4}$)	0.109	0.197	0.534
Linear LR ($\pi_T = 0.9, \lambda = 10^{-4}$)	0.115	0.205	0.527
LinSVM (unbalanced)	0.114	0.203	0.552
LinSVM ($\pi_T = 0.5, C = 5 * 10^{-1}$)	0.107	0.204	0.574
LinSVM ($\pi_T = 0.1, C = 5 * 10^{-1}$)	0.113	0.202	0.553
LinSVM ($\pi_T = 0.9, C = 5 * 10^{-1}$)	0.123	0.208	0.492
Quad-SVM (unbalanced)	0.108	0.201	0.507
Quad-SVM ($\pi_T = 0.5, 10^{-3}, 10$)	0.107	0.211	0.464
Quad-SVM ($\pi_T = 0.1, 10^{-3}, 10$)	0.107	0.201	0.516
Quad-SVM ($\pi_T = 0.9, 10^{-3}, 10$)	0.128	0.250	0.555
RBF-SVM (unbalanced)	0.114	0.211	0.526
RBF-SVM ($\pi_T = 0.5, 5 * 10^{-1}, 10^{-2}$)	0.112	0.214	0.547
RBF-SVM ($\pi_T = 0.1, 5 * 10^{-1}, 10^{-2}$)	0.112	0.211	0.530
RBF-SVM ($\pi_T = 0.9, 5 * 10^{-1}, 10^{-2}$)	0.143	0.262	0.555
fullCov-GMM (16 comp.)	0.127	0.223	0.618
diagCov-GMM (4 comp.)	0.140	0.275	0.287
tiedCov-GMM (8 comp.)	0.135	0.248	0.660