Machine Learning: Analysis of HTRU2 Dataset

294815 Marco Prattico'

July 2022

Contents

Abstract

The main goal of this work is to make a comparison between the most common techniques in Machine Learning in the classification of the HTRU2 dataset. In the first part, after an introduction to the dataset, we analyze the features to understand their distribution and their correlation and how to manipulate them to obtain better results. Then, we discuss the training of the models, and we choose the best ones. In the end, we do the score calibration of these and see how they actually work on the testing dataset to see if the results are what we expected.

1 Introduction

The HTRU2 dataset is a dataset that describes a sample of pulsar candidates collected during the High Time Resolution Universe Survey (South) [1]. 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 interstellar 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, the 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 information, 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 techniques are widely adopted to automatically label each sample. Classification systems, in particular, treat candidate datasets as binary classification problems, composed by true pulsar signals and false pulsar signals.

Each sample is described by eight features made up of continuous variables. The original data set is made up of 17,898 total examples, of which 1,639 are positive examples and 16,259 are negative examples. These samples are split as follows: 8929 samples are used for training and 8969 samples are used for testing. During this project, we will use a *K-Fold cross-validation* approach in order to get more data for training. We will use mainly a *K-Fold* with K = 5 and we will shuffle the data to get homogeneous folds.

2 Features

As we previously said, the dataset is made up of almost 18,000 samples described by eight continuous variables. In this project, the features are

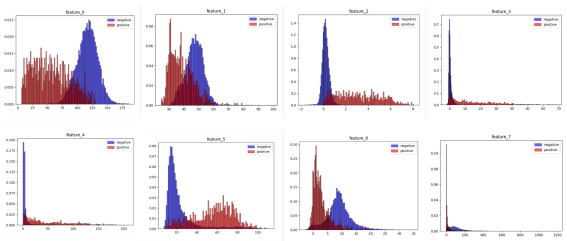


Figure 1: *Raw features*

pre-processed by means of z-normalization (centering and scaling to unit variance):

$$z_i = \frac{x_i - \mu}{\sigma}, \; \forall \, i \in 0, ..., N-1 \qquad (1)$$

In this case, x_i is the i-th sample, while μ and σ are the mean and covariance of the feature. The use of z-normalization helps to reduce the computational costs due to the non-normalized features. The results of the normalization are reported below.

2.1 Gaussianization

As reported in fig 2, the plot of the features seems well distributed with no relevant outliers. Therefore, the Gaussianization of the features probably

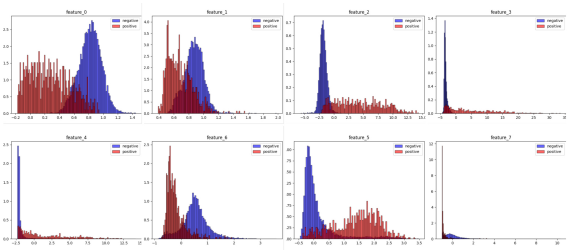


Figure 2: *Z-normalized features*

does not improve the classifiers' performances. We test anyway the effectiveness of the feature Gaussianization on the Gaussian models. The results of this kind of pre-processing is shown in fig. 3

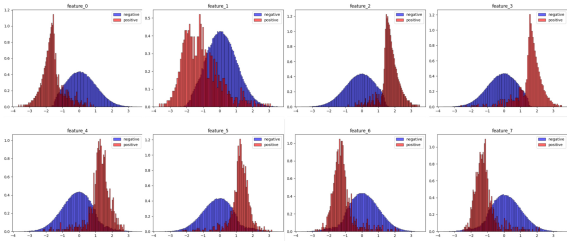


Figure 3: *Features Gaussianization*

2.2 Correlation

In this part, we analyze how the features are correlated to each other. We can compute the correlation using the following:

$$corr(X,Y) = \frac{Cov(X,Y)}{\sqrt{Var(X)\sqrt{Var(Y}} \qquad (2)$$

The heat map (fig. 4) shows that, probably, we can benefit from using the *PCA* to map the data to 7 or 6 features, in order to remove the highly correlated ones. In fact, in fig. 4 it is shown that features 3 and 4 and features 6 and 7 are highly correlated. Actually, *PCA (Principal Component Analysis)* is a preprocessing technique that consists of reducing the dimensionality of samples, preserving the features with the highest variance, which means finding a subspace containing the most "useful" information. In particular, it computes a mapping from an *m-dimensional* feature space to a *n-dimensional* feature space where $n << m$. The use of *PCA* may simplify classification reducing the effects due to high dimensionality and reducing the risk of overfitting. We see later the effect of reducing the dimensions on this dataset.

Figure 4: *Gray: whole Z-normalized training set. Red: True pulsar signal samples. Blue: False pulsar signal samples.*

3 Classifying the features

In this project, we will consider both cases, with a uniform prior, where $(\tilde{\pi}, C_{fp}, C_{fn}) = (0.5, 1, 1)$ and with a bias towards one of the two classes using $(\tilde{\pi}, C_{fp}, C_{fn}) = (0.1, 1, 1)$ and $(\tilde{\pi}, C_{fp}, C_{fn}) = (0.9, 1, 1)$.

In the first part, we are interested in choosing the most encouraging approach by detecting the minimum detection cost. In fact, we measure performance in terms of normalized minimum detection costs. However, *minDCF* is used to measure the cost to pay if we made the optimal decisions for the test set (in this case the validation set) through the use of the classifier scores.

In the second part, we will analyze how to choose an optimal threshold for classification.

3.1 Multivariate Gaussian Classifier

At the beginning, we consider the Gaussian classifiers that are based on a generative approach, in which they assume that the data have a normal distribution. These classifiers model the distribution of the observed samples given the class through the following relation:

$$\mathbf{X}|C \sim \mathcal{N}(\mu_c, \Sigma_c) \qquad (3)$$

	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$
	Z-normalized features - no PCA					
	Single Fold			5-Fold		
Full-Cov	0.141	0.261	0.739	0.140	0.286	0.672
Diag-Cov	0.201	0.304	0.704	0.213	0.476	0.732
Tied Full-Cov	0.104	0.230	0.489	0.112	0.224	0.573
Tied Diag-Cov	0.165	0.263	0.580	0.138	0.273	0.597
	Z-normalized features - PCA = 7					
Full-Cov	0.134	0.271	0.697	0.139	0.304	0.640
Diag-Cov	0.209	0.478	0.697	0.214	0.506	0.720
Tied Full-Cov	0.104	0.237	0.484	0.112	0.223	0.576
Tied Diag-Cov	0.137	0.270	0.550	0.138	0.273	0.597
	Z-normalized features - PCA = 6					
Full-Cov	0.156	0.261	0.662	0.152	0.288	0.652
Diag-Cov	0.217	0.480	0.680	0.223	0.531	0.727
Tied Full-Cov	0.148	0.253	0.508	0.140	0.259	0.580
Tied Diag-Cov	0.170	0.283	0.586	0.162	0.299	0.585
	Gaussianized features - no PCA					
Full-Cov	0.177	0.244	0.916	0.154	0.247	0.710
Diag-Cov	0.160	0.265	0.591	0.170	0.246	0.683
Tied Full-Cov	0.143	0.231	0.528	0.132	0.235	0.538
Tied Diag-Cov	0.160	0.280	0.624	0.136	0.250	0.557
	Gaussianized features - PCA = 7					
Full-Cov	0.177	0.244	0.867	0.153	0.245	0.697
Diag-Cov	0.194	0.275	0.863	0.162	0.248	0.660
Tied Full-Cov	0.141	0.234	0.509	0.134	0.248	0.660
Tied Diag-Cov	0.134	0.256	0.573	0.137	0.255	0.562
	Gaussianized features - PCA = 6					
Full-Cov	0.163	0.230	0.783	0.154	0.242	0.692
Diag-Cov	0.183	0.264	0.734	0.156	0.238	0.644
Tied Full-Cov	0.141	0.253	0.504	0.136	0.249	0.547
Tied Diag-Cov	0.143	0.263	0.586	0.141	0.258	0.588

Table 1: *Results of MVG classifier.*

3.1.1 Multivariate Gaussian Classifier

The first model used is the *Multivariate Gaussian Classifiers*. In table 1 , we can see the results using *MVG* classifier, with all kinds of covariance matrices that are: *Full Covariance matrix*, *Diagonal Covariance*, *Tied Full Covariance*, and *Tied Diagonal Covariance*. We also considered different techniques of preprocessing, such as *Z-normalization*, *PCA* and *Gaussianization* and we considered both cases, with a single split and with k-fold cross-validation with $k = 5$.

In this case, the best results are reached by *MVG* with a tied full covariance matrix trained on a single fold. The results with training on a 5-fold cross-validation are slightly worse. In fact, even though the results of a single-fold are slightly better, and since the results between k-fold and single-fold are consistent, we can use anyway the 5-fold cross-validation, because it guarantees a more robust training. Indeed, we

train our models on a larger amount of data by using 5 folds.

However, the use of *PCA* does not make the results worse significantly, in particular when we select 7 components. So, we can use *PCA* with 7 components without having worse performance and, at the same time, we reduce computational costs.

As we expected, the use of *"Gaussianization"* has no effect, in contrast, it makes the performances worse. So, in this case, the only kind of pre-processing that is useful is *PCA* (in addition to z-normalization).

However, it is useful to reduce the features to 7, through *PCA*, and to consider a tied covariance matrix. The use of a diagonal covariance matrix is advantageous because it decreases the number of parameters considered, but, in general, the performances are worse.

In the end, the best candidate is the *MVG* model, with a tied covariance matrix.

3.1.2 Naive Bayes Classifier

Here, we face the *Naive Bayes* classifier which is an MVG classifier with Naive Bayes assumptions. These assumptions suppose that features are independently distributed and uncorrelated to each other. This means that the covariance matrix elements off-diagonal should be equal to 0. Thus, the covariance matrix is a diagonal matrix. Since the heat map in fig 4 shows that the features are not uncorrelated, we do not expect optimal performances.

For the sake of simplicity, we report only the *Naive Bayes* classifier results applied on z-normalized data without using PCA. From table 2, we can observe that the results confirm our analysis and these are equal to the ones reported for the *MVG* classifier with a diagonal matrix. Here, in fact, the performances are non-optimal. So, we will not consider this classifier in future analyses.

3.2 Logistic Regression

Now, we focus on discriminative approaches and we compare the results with and without the PCA.

3.2.1 Linear Logistic Regression

Here, we turn our attention to regularized linear Logistic Regression. Since the classes are not balanced, we can re-balance the costs of different classes by minimizing:

$$J(w,b) = \frac{\lambda}{2}\|w\|^2 + \frac{\pi_T}{n_T}\sum_{i=1|c_i=1}^{n}\log(1+e^{-z_i(w^T x_i+b)}) + \frac{1-\pi_T}{n_F}\sum_{i=1|c_i=0}^{n}\log(1+e^{-z_i(w^T x_i+b)}) \tag{4}$$

So, we aim also to minimize the average cross-entropy between the distribution of observed and predicted labels. In our main application, we start by considering $\pi_T = 0.5$. At first, we have to tune the parameter λ, so we can plot *minDCF* with respect to different values of λ. According to what we saw in the previous section, we avoid the analysis with the use of *Gaussianization*. In the plots in fig 5, we consider again both cases with a single-fold and with five-folds. Here, it is clear that the minimum values of *minDCF* are reached when using a lower value of λ. Having a lower value of the hyper-parameter of λ means having a low relative weight of the regularization

	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$
	5-Folds		
MVG (Tied Full-Cov)	0.112	0.223	0.573
	Z-normalized features - No PCA		
Naive Bayes MVG	0.213	0.476	0.732

Table 2: *minDCFs results of Naive Bayes Classifier*

	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$
	5-Folds		
MVG (Tied Full-Cov)	0.112	0.223	0.573
	Z-normalized features - no PCA		
Log Reg ($\lambda = 10^{-4}$, $\pi_T = 0.5$)	0.115	0.219	0.528
Log Reg ($\lambda = 10^{-4}$, $\pi_T = 0.1$)	0.114	0.211	0.554
Log Reg ($\lambda = 10^{-4}$, $\pi_T = 0.9$)	0.118	0.222	0.521
	Z-normalized features - PCA = 7		
Log Reg ($\lambda = 10^{-4}$, $\pi_T = 0.5$)	0.114	0.219	0.549
Log Reg ($\lambda = 10^{-4}$, $\pi_T = 0.1$)	0.114	0.211	0.555
Log Reg ($\lambda = 10^{-4}$, $\pi_T = 0.9$)	0.119	0.219	0.526

Table 3: *Results of Linear Logistic Regression*

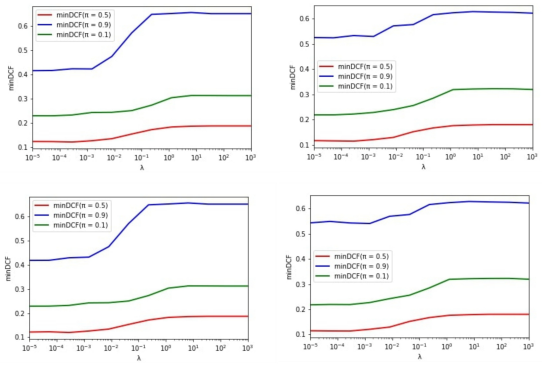


Figure 5: *minDCFs of Linear Logistic Regression model compared to the changing of λ. Top: only z-normalization. Bottom: PCA, with m = 7. Left: single fold. Right: 5-folds*

term $\frac{\lambda}{2}\|w\|^2$. This term allows for reducing the risk of over-fitting the training data. The choosing of the hyper-parameter is a trade-off between two cases:

- if λ is too large, we will obtain a solution that has a small norm, but is not able to well separate the classes
- if λ is too small, we will get a solution that has a good separation on the training set, but may have a poor classification accuracy for unseen data.

Then, we can choose $\lambda = 10^{-4}$ to train our model. We consider a training with different values of the prior $\tilde{\pi}$, to analyze the performance on other applications. In the end, from fig. 5 we can observe that the results between the single-fold and the k-folds are consistent, so we can consider only the case with 5 folds.

In table 3, we observe that we obtain similar results using $\pi_T = 0.5$ and $\pi_T = 0.1$. Then, doing the score balancing does not significantly improve performance. We also see that the *MVG* model with a tied full covariance matrix still works better than *Linear Logistic Regression*. Since *MVG* with tied full covariance and *Linear Logistic Regression* correspond to linear separation rules, we may assume that linear models work better than quadratic ones. In the end, we can say that balancing the scores is not necessary and *PCA* with $m = 7$, does not decrease the performances at all, consistent with what we saw before.

3.2.2 Quadratic Logistic Regression

In this case, we analyze what happens when using the *Quadratic Logistic Regression* model, which corresponds to quadratic boundaries. Since we cannot obtain a quadratic separation rule in the original feature space, we train the model in the expanded feature space. We define the feature vector $\phi(x) = [vec(xx^T)\ x]$. Then, since the posterior log-likelihood can be expressed as

$$s(x, w, c) = w^T \phi(x) + c \tag{5}$$

corresponding to quadratic forms in the original feature space, we are actually estimat-

	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$
	5-Folds		
MVG (Tied Full-Cov)	0.112	0.223	0.573
Log Reg ($\lambda = 10^{-4}$, $\pi_T = 0.5$)	0.114	0.219	0.549
	Z-normalized features - no PCA		
Quad Log Reg ($\lambda = 10^{-4}$, $\pi_T = 0.5$)	0.114	0.219	0.464
Quad Log Reg ($\lambda = 10^{-4}$, $\pi_T = 0.1$)	0.112	0.209	0.494
Quad Log Reg ($\lambda = 10^{-4}$, $\pi_T = 0.9$)	0.123	0.240	0.512
	Z-normalized features - PCA = 7		
Quad Log Reg ($\lambda = 10^{-4}\, \pi_T = 0.5$)	0.114	0.217	0.480
Quad Log Reg ($\lambda = 10^{-4}\, \pi_T = 0.1$)	0.111	0.205	0.487
Quad Log Reg ($\lambda = 10^{-4}\, \pi_T = 0.9$)	0.124	0.230	0.500

Table 4: *minDCFs results of Quadratic Logistic Regression*

ing quadratic separation surfaces in the original space. As for the linear case, we have to tune the hyper-parameter λ. The results reported in fig. 6 are similar to what we obtained in the previous case, so, similar to what we did before, we can choose $\lambda = 10^{-4}$. Also here, we restrict our analysis to the five-folds case.

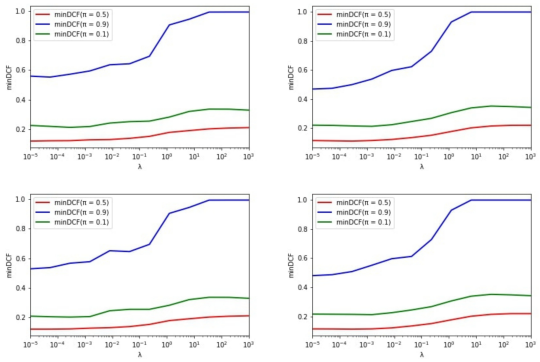


Figure 6: *minDCFs of Quadratic Logistic Regression model compared to the changing of λ. Top: only z-normalization. Bottom: z-normalization and PCA, with $m = 7$. Left: single fold. Right: 5-folds*

The table 4 shows that the Logistic Regression with a quadratic kernel has similar performances to the other models. In addition, the performances using $\pi_T = 0.1$ are not remarkably improved. Then, the results up to now show that both linear and quadratic boundaries work well for classification in this case. For this reason, we will analyze both linear and quadratic *SVM*.

3.3 Support Vector Machines

Therefore, we turn our attention to *SVMs*. First, we will analyze the *linear SVM* and we will address the problem regarding the tuning of the parameter C. Next, we will focus on non-linear SVMs: *Polynomial* and *Radial Basis Function (RBF)* SVM.

3.3.1 Linear SVM

Support Vector Machines are linear classifiers that look for maximum margin separation hyperplanes. The (primal) SVM objective consists of minimizing:

$$\mathcal{J}(w,b) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} max(0, 1 - z_i(w^T x_i + b)) \tag{6}$$

where n is the number of training samples, C is an hyper-parameter and z_i the class label for the i-th sample, encoded as:

$$z_i = \begin{cases} +1 \text{ if } x_i \text{ belongs to } \mathcal{H}_T \\ -1 \text{ if } x_i \text{ belongs to } \mathcal{H}_F \end{cases} \tag{7}$$

To solve the SVM problem, we can also consider the dual formulation:

$$\mathcal{J}^D(\alpha) = -\frac{1}{2}\alpha^T H\alpha + \alpha^T \mathbf{1}$$
$$s.t.\ 0 \leq \alpha_i \leq C,\ \forall i \in \{1,...,n\},\ \sum_{i=1}^{n} \alpha_i z_i = 0 \tag{8}$$

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

$$H_{ij} = z_i z_j x_i^T x_j \tag{9}$$

The dual formula is differentiable, however it contains the box-costrained and the additional constraint $\sum_{i=1}^{n} \alpha_i z_i = 0$. Due to the use of the L-BFGS algorithm that is capable of handling only box constraints, we have to reformulate the problem in order to remove the last constraint related to the presence of the bias term in the SVM primal formulation. We can reformulate the problem by using a mapping:

$$\hat{x}_i = \begin{bmatrix} x_i \\ K \end{bmatrix} \tag{10}$$

where $K = 1$ and the matrix $\hat{H}$ computed from the extended features $\hat{x}_i$ rather than from the original features x_i becomes:

$$\hat{H}_{ij} = z_i z_j \hat{x}_i^T \hat{x}_j \tag{11}$$

For linear SVM, we have to tune the hyper-parameter C. Again, we use the K-fold cross-validation to choose the best value of C. Accord-

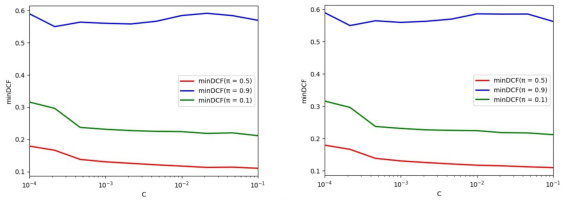


Figure 7: *minDCFs of Linear SVM model compared to the changing of C. Left: only z-normalization. Right: z-normalization and PCA, with $m = 7$.*

ing to fig. 7, we choose $C = 10^{-1}$ to obtain a low *minDCF*.

We can also deal with a re-balanced version of linear SVM. To re-balance the classes we use a different value of C in the box-constraint for different classes:

$$0 \leq \alpha_i \leq C_i,\ \forall i \in \{1,...,n\} \tag{12}$$

where $C_i = C_T$ for samples from classes $\mathcal{H}_T$ and $C_i = C_F$ for samples from class $\mathcal{H}_F$. We select $C_T = C\frac{\pi_T}{\pi_T^{emp}}$ and $C_F = C\frac{\pi_F}{\pi_F^{emp}}$, where π_T^{emp} and π_F^{emp} are the empirical priors for the two classes computed over the training set. According to fig

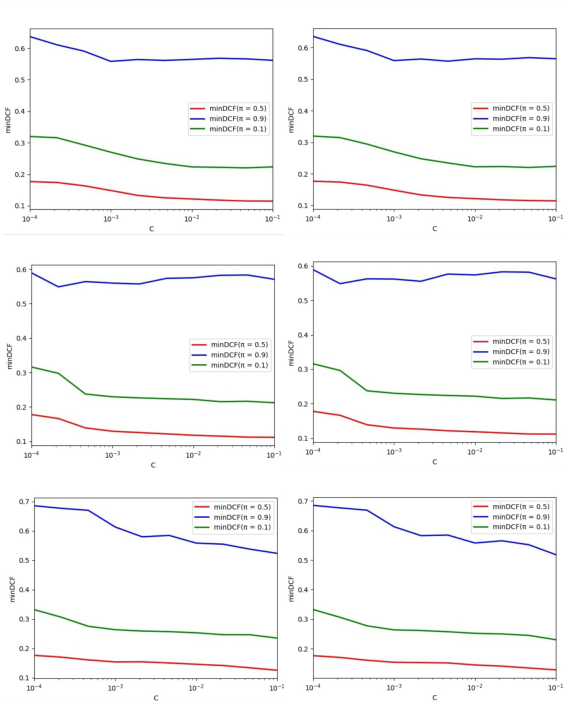


Figure 8: *minDCFs of Linear SVM models balanced compared to the changing of C. Left: only z-normalization. Right: z-normalization and PCA, with $m = 7$. First Row: balancing with $\pi_T = 0.5$. Second Row: balancing with $\pi_T = 0.1$. Third Row: balancing with $\pi_T = 0.9$*

8, $C = 10^{-1}$ is still a good value.

As expected, linear SVM performs similarly to the other linear models (table 5). In the end, class re-balancing is unnecessary so that we can use the default SVM formulation.

	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$
	5-Folds		
MVG (Tied Full-Cov)	0.112	0.223	0.573
Log Reg ($\lambda = 10^{-4}$, $\pi_T = 0.5$)	0.114	0.219	0.549
Quad Log Reg ($\lambda = 10^{-4}$, $\pi_T = 0.5$)	0.114	0.217	0.480
	Z-normalized features - PCA = 7		
Linear SVM ($C = 0.1$)	0.110	0.211	0.563
Linear SVM ($C = 0.1$, $\pi_T = 0.5$)	0.114	0.224	0.565
Linear SVM ($C = 0.1$, $\pi_T = 0.1$)	0.112	0.211	0.562
Linear SVM ($C = 0.1$, $\pi_T = 0.9$)	0.128	0.230	0.518

Table 5: *minDCFs results of LinearSVM*

3.3.2 Polynomial SVM

SVMs allow for non-linear classification through an implicit expansion of the features in a higher-dimensional space. The SVM dual objective depends on the training samples only through dot-products (as in (10)), and we can compute a classification score through scalar products between training and evaluation samples. Therefore, SVM does not require that we explicitly com-

	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$
	5-Folds		
MVG (Tied Full-Cov)	0.112	0.223	0.573
Log Reg ($\lambda = 10^{-4}$, $\pi_T = 0.5$)	0.114	0.219	0.549
Quad Log Reg ($\lambda = 10^{-4}$, $\pi_T = 0.5$)	0.114	0.217	0.480
Linear SVM ($C = 0.1$)	0.110	0.211	0.563
	Z-normalized features - no PCA		
Poly SVM ($C = 0.01$)	0.112	0.212	0.483
Poly SVM ($C = 0.01$, $\pi_T = 0.5$)	0.112	0.218	0.488
Poly SVM ($C = 0.01$, $\pi_T = 0.1$)	0.112	0.208	0.480
Poly SVM ($C = 0.01$, $\pi_T = 0.9$)	0.145	0.302	0.487
RBF SVM ($C = 0.1$)	0.158	0.259	0.579
RBF SVM ($C = 0.1$, $\pi_T = 0.5$)	0.167	0.303	0.596
RBF SVM ($C = 0.1$, $\pi_T = 0.1$)	0.155	0.259	0.578
RBF SVM ($C = 0.1$, $\pi_T = 0.9$)	0.232	0.393	0.578
	Z-normalized features - PCA = 7		
Poly SVM ($C = 0.01$)	0.113	0.211	0.511
Poly SVM ($C = 0.01$, $\pi_T = 0.5$)	0.114	0.216	0.499
Poly SVM ($C = 0.01$, $\pi_T = 0.1$)	0.112	0.207	0.535
Poly SVM ($C = 0.01$, $\pi_T = 0.9$)	0.170	0.328	0.573
RBF SVM ($C = 0.1$)	0.158	0.260	0.587
RBF SVM ($C = 0.1$, $\pi_T = 0.5$)	0.166	0.303	0.614
RBF SVM ($C = 0.1$, $\pi_T = 0.1$)	0.157	0.260	0.595
RBF SVM ($C = 0.1$, $\pi_T = 0.9$)	0.177	0.297	0.668

Table 6: *minDCFs results of kernel SVMs*

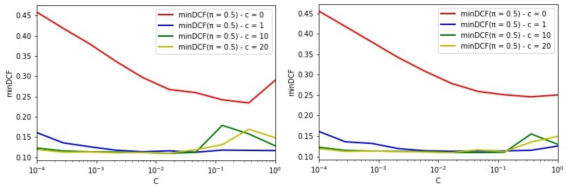


Figure 9: *minDCFs of Polynomial SVM model compared to the changing of C. Left: only z-normalization. Right: z-normalization and PCA, with $m = 7$.*

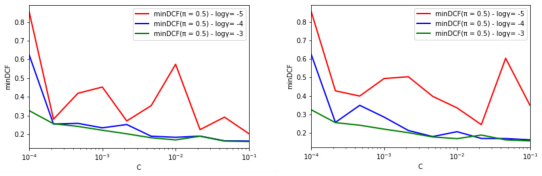


Figure 10: *minDCFs of RBF SVM model compared to the changing of C. Left: only z-normalization. Right: z-normalization and PCA, with $m = 7$.*

pute the feature expansion: it's sufficient that we are able to compute the scalar product between the expanded features $k(x_1, x_2) = \phi(x_1)^T \phi(x_2)$ where k is the *kernel function*. Then, we will compute the score of a test sample as follows:

$$s(x_t) = \sum_{i=1}^{n} \alpha_i^* z_i k(x_i, x_t) \tag{13}$$

Here, we use a *polynomial kernel* of degree $d = 2$, which corresponds to $k(x_1, x_2) = (x_1^T x_2 + c)^d$. Again, we tune the parameters C and c using a k-fold cross-validation approach. To obtain low values of *minDCF* we can choose $C = 0.01$ and $c = 10$. Since the plots look similar in both cases, with and without *PCA*, in particular in the neighborhood of the chosen value of C, we expect similar behavior in both cases. In fact, we can observe in table 7 that the results with and without *PCA* are really similar. Thus, we can use the model with *PCA* without significantly degrading performance. Since we used a polynomial quadratic kernel, the results of this model are similar to the ones obtained with the quadratic Logistic Regression.

In this case, we have similar performances to the best models analyzed before. We can see that the use of a balance of $\pi_T = 0.5$ does not significantly improve the performance of the classifier.

In the end, the hyper-parameter C allows selecting a trade-off between margin and errors on the training set, adding a penalty for each misclassified sample. If we choose a small value of C, the penalty for the misclassification is small and this brings us to choose decision boundaries with a large margin. In contrast, choosing a large value of C results in selecting decision boundaries with a small margin, in order to reduce misclassified samples. So, in terms of polynomial SVM, since we chose a lower value of C with respect to the linear case, here we have a larger margin corresponding to a less complex separation rule.

3.3.3 Radial Basis Function SVM

Now, we turn our attention to the *Radial Basis Function (RBF)* SVM, which is based on a quadratic approach. In this case, the kernel is expressed as:

$$k(x_1, x_2) = e^{-\gamma \|x_1 - x_2\|^2} \tag{14}$$

Here, we need to choose the best value of the hyper-parameter γ together with the C parameter. According to fig. 10, to get lower *minDCFs*, we can choose the kernel width $\gamma = 10^{-3}$ and $C = 0.1$.

As reported in table 7, the results of this classifier are worse than the other classifiers. Again, we underline that the reduction to 7 dimensions does not affect the performance of the classifier. Here, best performances are reached when using a balancing of $\pi_T = 0.1$.

3.4 Generative Mixture Models

The last model on which we focus is a generative approach based on training a *Generative Mixture Model (GMM)* over the data of each class. We

consider both full covariance and diagonal models, with and without tied covariance, using a five-fold cross-validation approach to select the number of Gaussians.

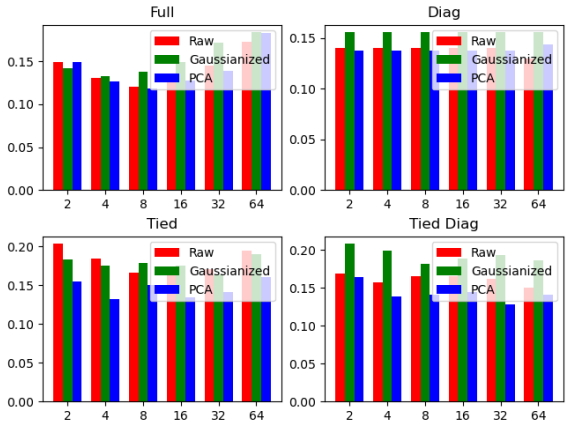


Figure 11: *Bar Plot describes how the minDCFs change compared to the changing of the number of components. The bars show the minDCFs value for different kinds of pre-processing.*

According to fig. 11 and 12, we can choose a number of components equal to 4 for our target application when using tied covariances and 8 components otherwise. We can observe that the performances decrease when we increase the number of components, in particular for $\tilde{\pi} = 0.9$, this happens because of the over-fitting problem. In this case, the best model is the GMM with a full covariance matrix. However, GMMs with other kinds of covariance perform slightly worse. Finally, the performances of the GMM model are not as good as those of the other models we analyzed in the previous sections.

3.5 Score calibration

In the end, we select the most promising candidate from different type of classifiers. In summary, we selected:

1. MVG with tied covariance matrix applied on z-normalized features with *PCA* where $m = 7$

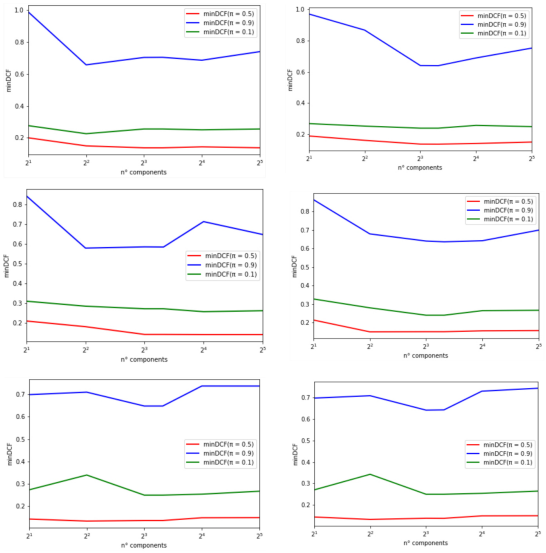
2. Linear Logistic Regression with $\lambda = 10^{-4}$ and $\pi_T = 0.5$ applied on z-normalized features with *PCA* where $m = 7$

3. Linear SVM with $C = 0.5$ applied on z-normalized features with *PCA* where $m = 7$

	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$
	5-Folds		
MVG (Tied Full-Cov)	0.112	0.223	0.573
Log Reg ($\lambda = 10^{-4}$, $\pi_T = 0.5$)	0.114	0.219	0.549
Quad Log Reg ($\lambda = 10^{-4}$, $\pi_T = 0.5$)	0.114	0.217	0.480
Linear SVM ($C = 0.1$)	0.110	0.211	0.563
Poly SVM ($C = 0.01$)	0.113	0.206	0.511
	Z-normalized features - no PCA		
GMM Full-Cov, 8 Gau	0.124	0.226	0.529
GMM Tied Full-Cov, 4 Gau	0.141	0.282	0.644
GMM Diag-Cov, 8 Gau	0.148	0.277	0.576
GMM Tied Diag-Cov, 4 Gau	0.163	0.305	0.734
	Z-normalized features - PCA = 7		
GMM Full-Cov, 8 Gau	0.123	0.224	0.560
GMM Tied Full-Cov, 4 Gau	0.140	0.294	0.645
GMM Diag-Cov, 8 Gau	0.141	0.221	0.613
GMM Tied Diag-Cov, 4 Gau	0.140	0.260	0.603

Table 7: *minDCFs results of GMM*

Figure 12: *minDCFs of GMM models balanced compared to the changing of $n-components$. Left: only z-normalization. Right: z-normalization and PCA, with $m = 7$. First Row: Full-Covariance. Second Row: Diagonal-Covariance. Third Row: Tied-Covariance*



Up to now, we have considered only the *minimum DCF*, that is the cost we would pay if we made optimal decisions for the evaluation set using the recognizer scores. The cost we would actually pay, however, depends on the goodness of the decisions we make using those scores, thus, in a binary case like this, depends on the goodness of the threshold we use to perform the class assignment. Therefore, we turn our attention to the *actual DCFs*.

If the scores are well calibrated, the optimal thresold that optimizes the Bayes risk is $t = -log(\frac{\tilde{\pi}}{1-\tilde{\pi}})$. We can now evaluate the *actual DCF* to assess how good the models would be if we were using the theoretical threshold for each application.

From table 8, we can observe that only linear Logistic Regression is almost well-calibrated in the main application. Regarding MVG and linear SVM, we see that there are losses, respectively, of $\approx 40\%$ and $\approx 50\%$ for the primary application and the calibration losses increase for other applications. Then, we can confirm this analysis by the *Beyesan Error Plot* of the models. However, we can re-calibrate the scores, transforming those so that the theoretical threshold $t = -log\frac{\tilde{\pi}}{1-\tilde{\pi}}$ provides close to optimal values over a wide range of effective priors $\tilde{\pi}$. We want to compute a transformation function f that maps the classifier scores s to well-calibrated scores $s_{cal} = f(s)$. We assume that the function f has a simple form:

	$\tilde{\pi} = 0.5$		$\tilde{\pi} = 0.1$		$\tilde{\pi} = 0.5$	
	min DCF	act DCF	min DCF	act DCF	min DCF	act DCF
	Z-normalized features - PCA = 7					
MVG	0.112	0.191	0.223	0.276	0.576	1.422
Linear Log Reg	0.114	0.116	0.219	0.227	0.549	0.568
Linear SVM	0.110	0.214	0.212	0.478	0.563	0.964

Table 8: *minDCFs vs actDCF of the chosen models before the calibration.*

	$\tilde{\pi} = 0.5$		$\tilde{\pi} = 0.1$		$\tilde{\pi} = 0.5$	
	min DCF	act DCF	min DCF	act DCF	min DCF	act DCF
MVG	0.111	0.117	0.228	0.231	0.521	0.556
Linear Log Reg	0.117	0.125	0.222	0.234	0.487	0.530
Linear SVM	0.111	0.129	0.213	0.233	0.498	0.542

Table 9: *minDCFs vs actDCF of the chosen models after the calibration.*

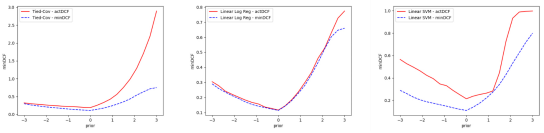


Figure 13: *minDCF vs actDCF computed with a 5-folds approach. Left: MVG model. Center: Quadratic Log Reg. Right: Polynomial SVM*

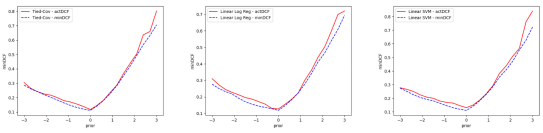


Figure 14: *minDCF vs actDCF after the calibration. Left: MVG model. Center: Quadratic Log Reg. Right: Polynomial SVM*

$$f(s) = \alpha s + \beta \tag{15}$$

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

$$f(s) = log\frac{f_{S|C}(s|\mathcal{H}_\mathcal{T})}{f_{S|C}(s|\mathcal{H}_\mathcal{F})} = \alpha s + \beta \tag{16}$$

The class posterior probability for prior $\tilde{\pi}$ corresponds to:

$$log\frac{f_{S|C}(s|\mathcal{H}_\mathcal{T})}{f_{S|C}(s|\mathcal{H}_\mathcal{F})} = \alpha s + \beta + log\frac{\tilde{\pi}}{1-\tilde{\pi}} \tag{17}$$

We can interpret the scores as features and this has a similar expression to the log posterior ratio of the Logistic Regression model. In fact, we can rewrite $\beta' = \beta + log\frac{\tilde{\pi}}{1-\tilde{\pi}}$ and we have exactly the same model. Now, we can employ the prior-weighted Logistic Regression model to learn the parameter α and β', over our training scores. To recover the calibrated scores $f(s)$, we need to compute:

$$f(s) = \alpha s + \beta = \alpha s + \beta' - log\frac{\tilde{\pi}}{1-\tilde{\pi}} \tag{18}$$

In this case, we have to specify a prior $\tilde{\pi}$ to compute the calibration. If we choose a prior $\tilde{\pi} = 0.5$, other applications will also benefit from the calibration. This is shown in fig. 14. For the calibration, we used a value of $\lambda = 10^{-4}$ and the results are reported in table 9. We observe that the differences between minDCF and actDCF are reduced as expected, except for linear Logistic Regression, which has already had well-calibrated scores, but now the results are slightly worse than before.

4 Experimental results

Now, we analyze the results of the classifiers over the test dataset, evaluating the systems in terms of *minDCFs*. The previous analysis was based mainly on the k-fold approach, so we train here our models using the whole training-set. The results reported in table 10 are consistent with the analysis done using the training and validation set. In fact, also in this case, the models chosen previously work as well as in the previous analysis. Therefore, due to the similarity between the results obtained when using the validation and evaluation sets, we can say that the training and evaluation sets are not very different.

The next step is to calibrate the scores of the chosen classifiers, as before, and then to compare the models using a *Receiver Operating Characteristics (ROC)* curve.

Now, we can compare the classifiers through the *ROC*, in which the best classifier has the highest *Area Under Curve (AUC)*. In fig. 15, we can see that *linear Logistic Regression* and *linear SVM* have the highest *AUC*, while the tied MVG model have a slightly lower *AUC*. We can conclude that the two best classifiers, from those chosen previously, are: *linear SVM* and *linear Logistic Regression*, both using *PCA* with $m = 7$.

5 Conclusions

In conclusion, by our analysis, we can say that the models we chose during validation also perform well in the evaluation set. Since the chosen classifiers deal with linear boundaries, we can conclude that these kinds of boundaries work well with this dataset. In particular, we found that

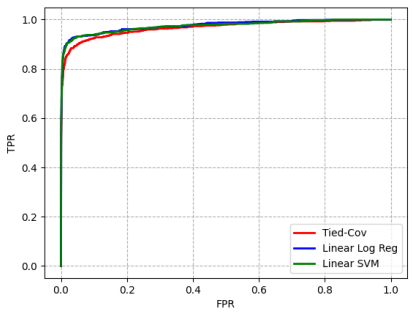


Figure 15: *ROC curve of the chosen models*

linear Logistic Regression and *linear SVM* have very good performances on the evaluation set, while the MVG performance is slightly lower. In general, we can reach $minDCF \approx 0.1$ for balanced applications. Although, for unbalanced applications, the performances are not bad. In fact, we can reach $minDCF \approx 0.2$ if the prior $\tilde{\pi} = 0.1$ and $minDCF \approx 0.5$ if the prior $\tilde{\pi} = 0.9$. Thus, the analysis made on the training/validation set appears effective also on the evaluation set.

	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$
	5-Folds		
	z-normalized features - no PCA		
MVG (Full-Cov)	0.140	0.282	0.645
MVG (Diag-Cov)	0.185	0.329	0.621
MVG (Tied Full-Cov)	0.110	0.207	0.591
MVG (Tied Diag-Cov)	0.152	0.262	0.544
Log Reg ($\lambda = 10^{-4}$, $\pi_T = 0.5$)	0.109	0.199	0.538
Quad Log Reg ($\lambda = 10^{-4}\, \pi_T = 0.5$)	0.104	0.202	0.420
Linear SVM ($C = 0.1$)	0.114	0.199	0.548
Poly SVM ($C = 0.01$, $\pi_T = 0.5$)	0.101	0.203	0.459
RBF SVM ($C = 0.1$, $\pi_T = 0.5$)	0.115	0.223	0.498
GMM Full-Cov, 8 Gau	0.111	0.223	0.590
GMM Diag-Cov, 8 Gau	0.145	0.289	0.548
GMM Tied Full-Cov, 8 Gau	0.138	0.283	0.580
	z-normalized features - PCA = 7		
MVG (Full-Cov)	0.139	0.293	0.574
MVG (Diag-Cov)	0.201	0.514	0.755
MVG (Tied Full-Cov)	0.110	0.208	0.587
MVG (Tied Diag-Cov)	0.141	0.257	0.556
Log Reg ($\lambda = 10^{-4}$, $\pi_T = 0.5$)	0.109	0.202	0.538
Quad Log Reg ($\lambda = 10^{-4}\, \pi_T = 0.5$)	0.104	0.201	0.442
Linear SVM ($C = 0.1$)	0.115	0.202	0.557
Poly SVM ($C = 0.01$, $\pi_T = 0.5$)	0.102	0.203	0.469
RBF SVM ($C = 0.1$, $\pi_T = 0.5$)	0.116	0.222	0.504
GMM Full-Cov, 8 Gau	0.119	0.229	0.520
GMM Diag-Cov, 8 Gau	0.140	0.227	0.580
GMM Tied Full-Cov, 8 Gau	0.138	0.278	0.567

Table 10: *minDCFs results when using the evaluation-set*

References

[1] A D Cameron et al. “The High Time Resolution Universe Pulsar Survey – XVI. Discovery and timing of 40 pulsars from the southern Galactic plane”. In: *Monthly Notices of the Royal Astronomical Society* 493.1 (Jan. 2020), pp. 1063–1087. DOI: 10.1093/mnras/staa039. URL: https://doi.org/10.1093%2Fmnras%2Fstaa039.