Machine Learning & Pattern Recognition

Gender Identification 2022/2023

Roberto Pulvirenti s317704

Gabriele Greco s303435

Table of Contents

1. Abstract

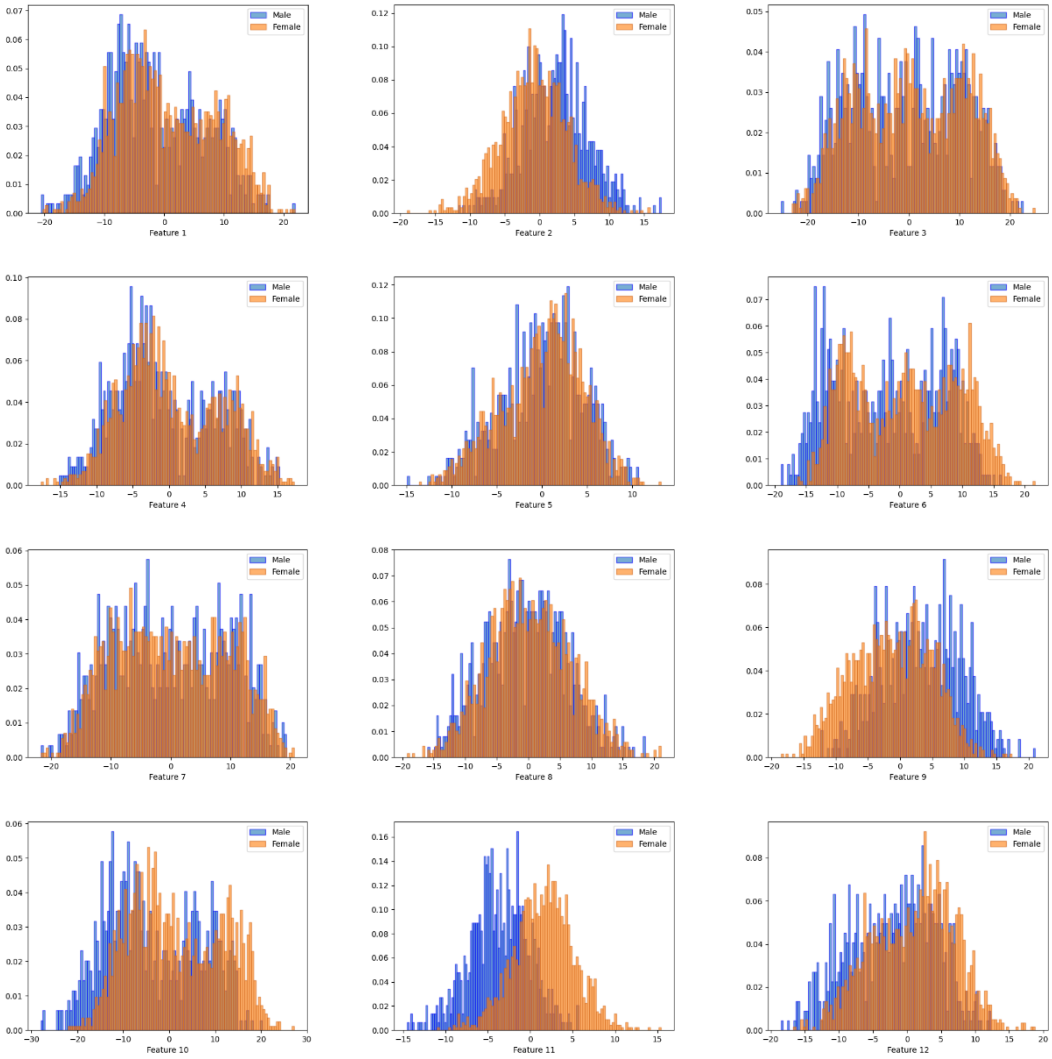
This report aims to analyse a dataset composed of various low-level face images of males and females using different Machine Learning models. First, we will explore the structure of the dataset and try to understand how it is distributed. Then, we will evaluate various classifiers and try to find the best system that can accurately classify our samples with the lowest cost.

2. Introduction

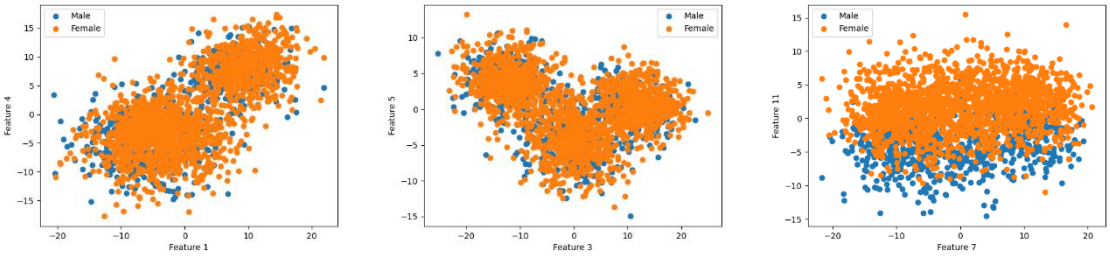
The dataset consists of image embeddings, i.e. low-dimensional representations of images obtained by mapping face images to a common, low-dimensional manifold (typically few hundred dimensions), for example by means of suitable neural networks. To keep the model tractable, the dataset consists of synthetic data, and embeddings have significantly lower dimension than in real use-cases. The embeddings are 12-dimensional, continuous-valued vectors, belonging to either the male (label 0) or the female (label 1) class. The embedding components do not have a physical interpretation. The samples belong to 3 different age groups, each one characterized by a different distribution; however, this information is not available. The target application assumes that the prior probabilities for the two classes are equal, but this does not hold for the misclassification costs. Moreover, there's another issue to take in consideration: the training dataset is composed of 2400 samples, 720 males and 1680 females, this means that it is very biased towards female class, while the test dataset is made up of 6000 samples of which 4200 males and 1800 females therefore does not correspond to the class proportion of the training set.

2.1. Feature analysis

We start analyzing the dataset through histograms plot of the single features in order to get some information about their distribution (data were centered before plot):

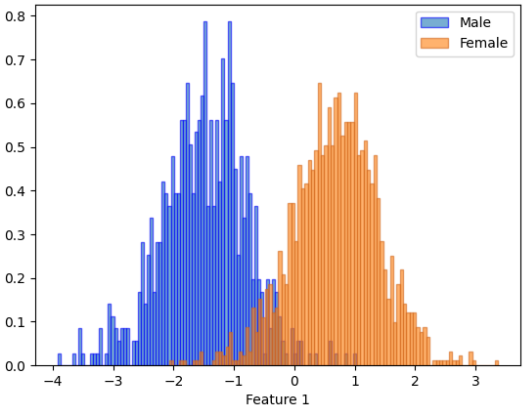


Taking a look to these plots permit us to make the first considerations, in fact histograms 2-5-8-11 remember a gaussian distribution and some others reflect the fact that in our dataset are represented 3 different group ages, such as histograms of feature 3 and 6. Feature 11 looks like the one that most easily divides the samples of the two classes. To confirm what has just been deduced, here are some of the most representative cross-feature plots. It's evident that the distribution of the two classes is very similar in most cases and that, for some features, there are 3 different gaussian form representative of the 3 different group ages:



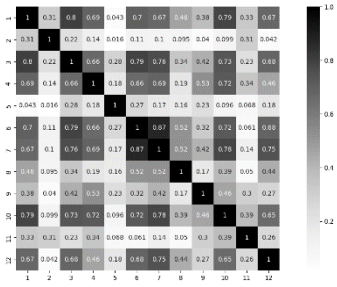
This means that gaussian models could perform well.

A transformation that we can use is the Linear Discriminant Analysis (LDA), this preprocessing step allows us to understand if the features are linearly discriminable, so this means that if a linear and/or gaussian model may perform better than no-gaussian and/or no-linear model. LDA founds C-1 directions over to plot our features where C is the number of classes (in our case we have only 1 dimension because we trait a binary classification problem):



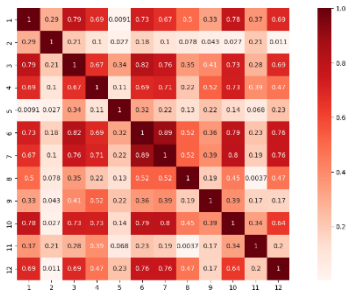
As we can see from this plot, classes are discriminable but there is a small region where we can make some mistakes.

We focus now on the correlation between the features using the Pearson correlations and plotting the results on a heatmap:



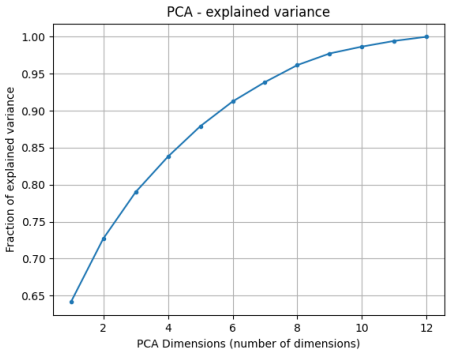
Heatmap of the entire dataset

Heatmap of male class



Heatmap of female class

From the heatmap of the entire dataset we can see that in general a lot of features are slightly correlated (from 0.4 to 0.7) and some others are strongly correlated (≈0.8), such us features 1-3, 1-10, 7-10 and 6-7 (the latter has a correlation coefficient of 0.87), so this suggest that we may have some benefits if we map data from a 12-dimensional space to 9 or 10-dimensional space in order to reduce the number of parameters to estimate for a model. So, we now look at the plot of the cumulative variance over PCA dimensions to understand how much information we lose in applying PCA:



The plot explains us that if we remove 2/3 dimensions, we will maintain around the 97% of the variance, so we will try our model applying PCA.

The last observation is made on the 2 heatmaps of male and female class, they are very similar so this means that the MVG model and tied Gaussian model (MVGT) will probably have similar performance (because of the assumption of MVGT, which says that the covariance matrices of the 2 classes are the same) and we can also say that the Naïve Bayes model will perform worst because there's a moderate correlation between the various features.

3. Classification and validation

This report is based on the results obtained from the following list of classification models:

- Generative models - Gaussian classifiers:
  - Multivariate Gaussian classifier (MVG)
  - MVG + Diagonal Covariance
  - MVG + Tied Covariance
  - MVG + Diagonal and Tied Covariance
- Logistic Regression:
  - Prior weighted Logistic Regression
  - Quadratic Logistic Regression
- SVM - Support Vector Machine:
  - Linear SVM
  - SVM with Polynomial kernel function with degree=2
  - SVM with RBF kernel function
- GMM - Gaussian Mixture Model:
  - Gaussian Mixture Model
  - GMM + Diagonal Covariance
  - GMM + Tied Covariance
  - GMM + Diagonal and Tied Covariance

For the validation phase, to determine the most promising and suitable model and evaluate the impact of using PCA, it has been decided to employ a K-Fold cross validation with K = 5. The reason behind this decision is that through this methodology we can use a larger amount of data for both training and validation sets.

To measure the performance of our models, we will utilize minDCF. This metric considers the costs associated with different types of errors and provides a comprehensive evaluation of the model's effectiveness in detecting the target class.

Since our dataset is highly unbalanced we will focus not only on the primary application denoted as $(\tilde{\pi}, C_{fn}, C_{fp}) = (0.5, 1, 1)$ but also on other types of applications, specifically $(\tilde{\pi}, C_{fn}, C_{fp}) = (0.1, 1, 1)$ and $(\tilde{\pi}, C_{fn}, C_{fp}) = (0.9, 1, 1)$.

3.1. Generative models – Gaussian classifiers

We start the examination with Gaussian classifiers. They all work with the assumption that the data follows a gaussian distribution, so our goal is to retrieve the density distribution from the training set and use it to get the predictions through the class posterior probability:

$$P(C_t = c|X_t = x_t) = \frac{P(C_t = c)f_{X|C}(x_t, c)}{\sum_{c' \in C} f_{X|C}(x_t, c')} \quad where \ \mathrm{X|C} = \mathrm{c} \sim \mathrm{N}(\mu_c, \Sigma_c)$$

In the following, we report the results for different scenarios:

	RAW			*Z-SCORE*		
	$\pi = 0,5$	$\pi = 0,1$	$\pi = 0,9$	$\pi = 0,5$	$\pi = 0,1$	$\pi = 0,9$
MVG	0.113	0.297	0.350	0.113	0.297	0.350
NB	0.463	0.771	0.777	0.463	0.771	0.777
MVGT	0.109	0.299	0.342	0.109	0.299	0.342
MVGT(NB)	0.457	0.770	0.781	0.457	0.770	0.781

Gaussian Classifier on RAW and Z-normalized data

As mentioned before the MVGT performs very similar to the MVG model but with slightly some improvement overall, while the Naïve Bayes model is the worst because of its assumption (not applicable for our dataset). The Z-Score tries to transform the data by simply centring and dividing by the standard deviation, so the results are like what we see with the RAW features. Following we have the results applying PCA:

	RAW			*Z-SCORE*		
	$\pi = 0,5$	$\pi = 0,1$	$\pi = 0,9$	$\pi = 0,5$	$\pi = 0,1$	$\pi = 0,9$
MVG	0.117	0.308	0.348	0.122	0.312	0.358
NB	0.127	0.324	0.370	0.124	0.312	0.349
MVGT	0.118	0.289	0.355	0.118	0.299	0.356
MVGT(NB)	0.124	0.302	0.360	0.123	0.294	0.355

Gaussian Classifier with PCA(11)

	RAW			*Z-SCORE*		
	$\pi = 0,5$	$\pi = 0,1$	$\pi = 0,9$	$\pi = 0,5$	$\pi = 0,1$	$\pi = 0,9$
MVG	0.163	0.402	0.493	0.187	0.407	0.538
NB	0.168	0.449	0.469	0.184	0.435	0.546
MVGT	0.161	0.393	0.474	0.183	0.428	0.535
MVGT(NB)	0.170	0.396	0.479	0.179	0.421	0.543

Gaussian Classifier with PCA(10)

	RAW			*Z-SCORE*		
	$\pi = 0,5$	$\pi = 0,1$	$\pi = 0,9$	$\pi = 0,5$	$\pi = 0,1$	$\pi = 0,9$
MVG	0.187	0.406	0.557	0.220	0.500	0.578
NB	0.193	0.447	0.544	0.208	0.486	0.597
MVGT	0.185	0.397	0.542	0.212	0.476	0.577
MVGT(NB)	0.185	0.415	0.541	0.210	0.482	0.589

Table 1: Gaussian Classifier with PCA(9)

As we can see from the tables PCA seems to not improving the performance of our classifiers, or better, it improves the performance of the Naïve Bayes classifiers, but not of the others. The reason why this happens is because for Naïve Bayes models features must be the most uncorrelated possible in order to have a covariance matrix as diagonal as possible, PCA gives us orthogonal directions where to plot our data, given that orthogonality means independence, so our projected data will be independent.

Looking at the results obtained, we can see that PCA doesn't improve the performance of the model. The only dimensionality that seems to not lose too much information is 11. However, not having

substantial differences between the various minDCFs, we opted to verify with other models if PCA(11) could be a worth preprocessing analysis. Moreover, we will continue using the Z-scored data.

3.2. Logistic Regression

Now we will focus on the discriminative models such as Logistic Regression Model (LR) and Quadratic Logistic Regression Model (QLR). These types of models don't make any assumption over the distribution of the data but simply find a hyperplane that maximizes the posterior probability. To compute the score of a sample this formulation is used:

$$s(x) = w^T x + b$$

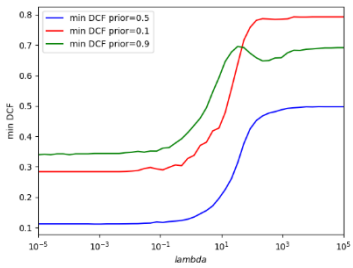
Where w^T represents the orthogonal vector with respect to the hyperplane and b represents the bias term. (w, b) are the model parameters. To estimate them we need to minimize the following objective function (since the classes are unbalanced, we used the prior-weighted regularized version):

$$J(w,b) = \frac{\lambda}{2}\|\mathrm{w}\|^2 + \frac{\pi_\mathrm{T}}{n_\mathrm{T}} \sum_{i=1|c_i=1}^{n_T} \log\left(1 + e^{-z_i s_i}\right) + \frac{1-\pi_\mathrm{T}}{n_\mathrm{F}} \sum_{i=1|c_i=0}^{n_F} \log\left(1 + e^{-z_i s_i}\right)$$

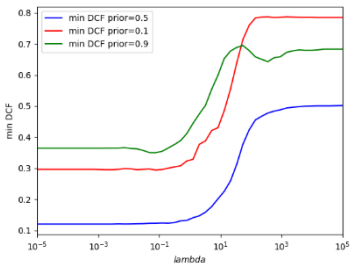
Where:

- $z_i = 1$ if $c_i = 1$ or $z_i = -1$ if $c_i = 0$.
- π_T is the prior probability that permits us to generalize the model to our target application.
- $\frac{\lambda}{2}\|\mathrm{w}\|^2$ is a regularization term that helps obtaining a w with lower norm (reducing the risk of over-fitting the training data), with λ is an hyperparameter called regularization coefficient.

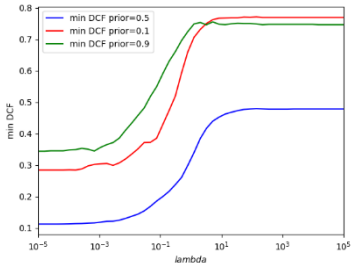
To choose the best value for lambda we plot the DCFmin over different possible values in a range from 10^{-5} to 10^5 with prior probability = 0,5 (our target application):



Plot LR with prior=0.5, RAW data



Plot LR with prior=0.5, PCA(11)



Plot LR with prior=0.5, Z-scored data

As we see from these plots, there is no reason to preprocess our data with PCA since it doesn't improve the performance of the model. So, from now on we will use only RAW and Z-scored data. Another important thing to point out is that the minimum is reached by setting λ=0, so we could consider it as the best value for hyper parameter λ. To analyze the performance, we will consider the following Table:

	RAW			Z-SCORE		
	$\boldsymbol{\pi = 0,5}$	$\boldsymbol{\pi = 0,1}$	$\boldsymbol{\pi = 0,9}$	$\boldsymbol{\pi = 0,5}$	$\boldsymbol{\pi = 0,1}$	$\boldsymbol{\pi = 0,9}$
LR($\boldsymbol{\pi_T = 0,5}$)	0.112	0.284	0.341	0.112	0.284	0.340
LR($\boldsymbol{\pi_T = 0,1}$)	0.120	0.299	0.377	0.120	0.299	0.377
LR($\boldsymbol{\pi_T = 0,9}$)	0.111	0.315	0.344	0.111	0.315	0.344

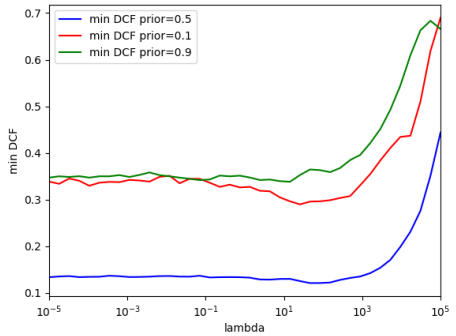
Logistic Regression on RAW and Z-normalized data, λ = 0

The Z-normalization we do on data is linear, therefore the performance obtained are equivalent. We can now make some considerations on λ: a large value of λ causes the model to over-generalize, leading in poor classification performance (and we have a proof of that looking at the plot of minDCF in function of λ); on the other hand, a small value of λ creates a good separation on the training set, risking to overfit the data. To strike a balance between overfitting and underfitting, a suitable value of λ could be around 10^{-1}. Using different values for π_T does not improve a lot the model, as we can see there is a slightly improvement if we try to use π_T =0.9 but it's caused by the fact that the dataset is imbalanced to the class 1 (female class).

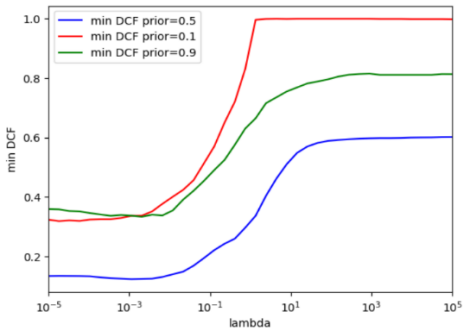
Now we will focus on Quadratic Logistic Regression (QLR), it is a type of regression model that allows us to estimate the probabilities of belonging to a binary class as a function of continuous or categorical explanatory variables, including quadratic terms of continuous variables. This allows to capture any non-linear effects of explanatory variables on the response variable. Another aspect to consider in quadratic logistic regression is the transformation of explanatory variables. We can use the:

$$\Phi(x) = \begin{pmatrix} vec(xx^T) \\ x \end{pmatrix}$$

function to expand our feature space. In this way, we can train the logistic regression model using the ϕ(x) feature vectors instead of x. This allows us to calculate linear separation rules for ϕ(x), which corresponds to estimating quadratic separation surfaces in the original space. As we did for LR we first compute the best value for λ and then we'll use it to train our model. We expect worse performance from the LR model because the analysis of features highlighted that linear rules can well separate the two classes (as shown in LDA plot).



*Plot minDCF in function of λ for QLR ($\boldsymbol{\pi_T = 0,5}$) , **RAW***



Plot minDCF in function of λ for QLR ($\boldsymbol{\pi_T = 0,5}$), $\boldsymbol{Z - Score}$

The results are different from Raw features and Z-Scored because of the feature expansion operation, in fact since we are computing a dot product inside another embedding space the model becomes sensitive to data transformation. We have decided a value of λ = 50 for the raw data, while λ = 0 for the Z-normalized data.

	RAW($\lambda = 50$)			Z-SCORE($\lambda = 0$)		
	$\pi = 0,5$	$\pi = 0,1$	$\pi = 0,9$	$\pi = 0,5$	$\pi = 0,1$	$\pi = 0,9$
QLR($\pi_T = 0,5$)	0.120	0.295	0.367	0.134	0.322	0.359
QLR($\pi_T = 0,1$)	0.127	0.307	0.404	0.139	0.388	0.381
QLR($\pi_T = 0,9$)	0.130	0.394	0.369	0.135	0.338	0.383

Quadratic Logistic Regression on RAW and Z-normalized data

3.3. Support Vector Machine

In this section, we will focus on Support Vector Machine (SVM) model. The first important thing to remark is that these models produce a score that has no probabilistic interpretation. The problem of minimizing risk in Logistic Regression can be extended to a more general problem that allows for the separation of samples up to a certain margin. This approach is known as Support Vector Machine. To solve the SVM problem, we can use the dual formulation, which is easier to optimize because its complexity depends only on the number of samples. This also allows us to compute non-linear hyperplanes without having to explicitly expand the features:

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

$$\text{with } 0 \le \alpha_i \le C_i, \forall i \in \{1, \dots, n\} \; and \; \sum_{i=1}^{n} \alpha_i z_i = 0$$

As for logistic regression we can consider a balanced version of the problem, and this is made by considering different values of C for each class in the box constraint of the dual formulation:

$$C_i = \begin{cases} C\frac{\pi_T}{\pi_T^{emp}} \; if \; i \in Class \; 1 \\ C\frac{\pi_F}{\pi_F^{emp}} \; if \; i \in Class \; 0 \end{cases}$$

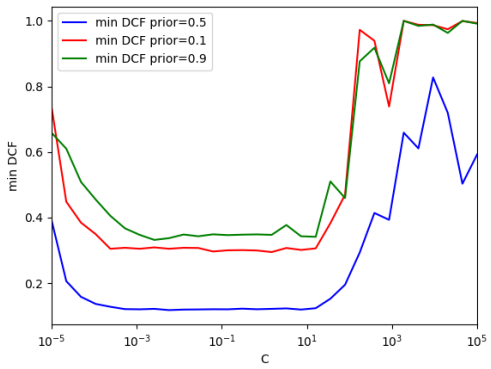
We'll start our analysis from the linear version of SVM in which we reformulate the primal problem as the minimization of:

$$\hat{J}(\hat{w}) = \frac{1}{2}\|\hat{w}\|^2 + C\sum_{i=1}^{n} \max(0, 1 - z_i(\hat{w}^T \hat{x}_i))$$

$$where \; \hat{w} = \begin{bmatrix} w \\ b \end{bmatrix}, \qquad \hat{x_i} = \begin{bmatrix} x_i \\ 1 \end{bmatrix}$$

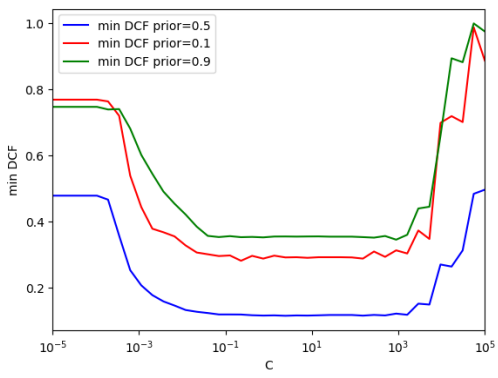
The first thing to do is to find a good value for the hyperparameter C:

RAW



Plot of minDCF in function of C for SVM($\pi_T = 0.5$)

Z-score



Plot of minDCF in function of C for SVM($\pi_T = 0.5$)

In general, C represents the margin, and it can be seen as a trade-off between a low error in training or a large margin. If C goes to infinite, we do not generalize very well and overfit the data; differently if C goes to 0, it means we do not care about the training data and we want a large margin. The choice of C is critical and looking at the plot a value of C=10 seems to optimize the different applications.

	RAW(C= 10)			*Z-SCORE(C= 10)*		
	$\boldsymbol{\pi = 0,5}$	$\boldsymbol{\pi = 0,1}$	$\boldsymbol{\pi = 0,9}$	$\boldsymbol{\pi = 0,5}$	$\boldsymbol{\pi = 0,1}$	$\boldsymbol{\pi = 0,9}$
SVM($\boldsymbol{\pi_T = 0,5}$)	0.117	0.299	0.343	0.115	0.293	0.355
SVM($\boldsymbol{\pi_T = 0,1}$)	0.127	0.316	0.391	0.126	0.315	0.372
SVM($\boldsymbol{\pi_T = 0,9}$)	0.120	0.311	0.336	0.116	0.321	0.333

SVM on RAW and Z-normalized data with C=10

Looking at the results, our initial assumption about the linear decision rule being one of best option to separate the classes is reinforced. The table shows similar results to the LR's one but has an important difference: our model performs better on Z-normalized data differently from what happened in the LR model. This is because SVM tries to find the optimal hyperplane that separates the data points of different classes with the maximum margin, if the features are on different scales, the hyperplane will be heavily influenced by those with larger values, potentially leading to suboptimal results. Z-normalizing the data helps in avoiding this issue.

What makes SVM a powerful tool is the possibility to train the model to obtain multidimensional hyperplanes without the need to transform the feature space. In fact, we know that the model depends mathematically only on the dot product of training and test samples, so if we define a transformation $\Phi(x_i)$ to go in the expanded feature space, we don't need to transform our features if we can build a function that directly computes dot product. This function is called kernel function and both training scoring can be performed using it:

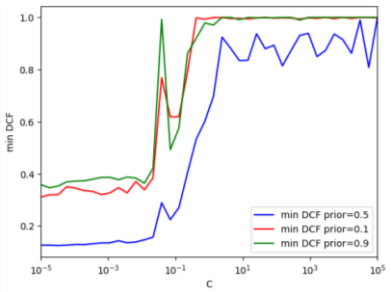
$$k(x_1, x_2) = \Phi(x_1)^T \Phi(x_2), \qquad H_{ij} = z_i z_j k(x_i, x_j) \quad and \quad s(x_t) = \sum_{i=1|\alpha_{i>0}}^{n} \alpha_i z_i k(x_i, x_j) + b$$

We will explore well known kernel functions like:

- **Polynomial**: $k(x_1, x_2) = (x_1^T x_2 + c)^2$;
- **RBF-Radial Basis Kernel Function**: $k(x_1, x_2) = e^{-\gamma \|x_1 - x_2\|}$;

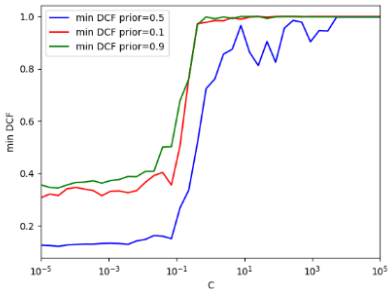
Both the models depend on the hyperparameter C, Polynomial version also on c and the radial one on γ. So, we have decided to plot some possibilities for c (0,1,10) and γ (10^{-1}, 10^{-2}, 10^{-3}) in function of C. We started from the Polynomial version:

RAW



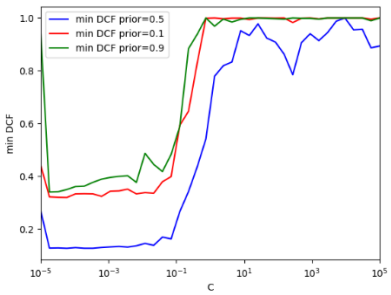
Polynomial SVM ($\pi_T = 0.5$), c=0

RAW



Polynomial SVM ($\pi_T = 0.5$), c=1

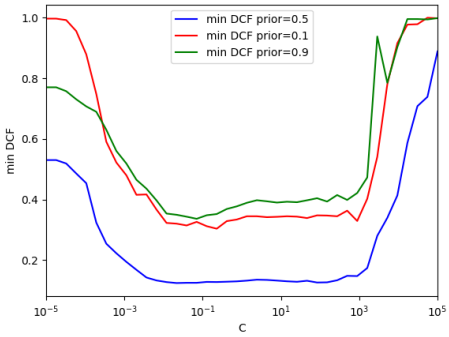
RAW



Polynomial SVM ($\pi_T = 0.5$), c=10

From the plots the best value to tune c seems to be c=1, for this reason we decided to also plot the z-score for the corresponding value:

Z-score



Polynomial SVM (π_T=0.5), c=1

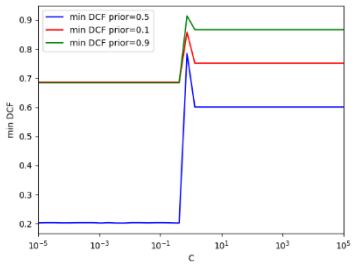
Now we have to decide the value for C, taking care of all the considerations we did before about its importance and its significance. We opted for C = 10^{-3} for raw data and C = 10^{-1} for the Z-normalized features.

	***RAW*($C = 10^{-3}$, *c=1*)**			***Z-SCORE*($C = 10^{-1}$, $c = 1$)**		
	$\boldsymbol{\pi = 0,5}$	$\boldsymbol{\pi = 0,1}$	$\boldsymbol{\pi = 0,9}$	$\boldsymbol{\pi = 0,5}$	$\boldsymbol{\pi = 0,1}$	$\boldsymbol{\pi = 0,9}$
PolySVM($\pi_T = 0,5$)	0.135	0.336	0.369	0.127	0.318	0.346
PolySVM($\pi_T = 0,1$)	0.150	0.411	0.457	0.143	0.364	0.443
PolySVM($\pi_T = 0,9$)	0.143	0.415	0.360	0.139	0.401	0.318

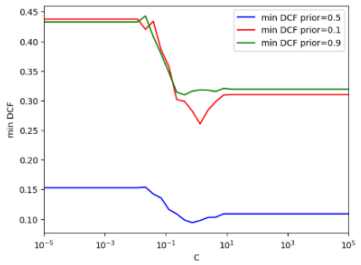
Polynomial SVM on RAW and Z-normalized data, c=1

Focusing on the RBF version, we need to find good values for hyperparameters (γ, C); in theory, since there is no correlation between them, we should try every possible couple of values for our model. This is not possible not only for a matter of time but also because γ represents the width of the kernel (where kernel is a synonym of distance), "similar", i. e. closer, points have a higher kernel value. So, if it's small the S.V.s influence most other points, while if big they tend to lose importance for points that are not close. For this reason, we decided to try 3 possible values for γ (10^{-1}, $10^{-2}, 10^{-3}$).

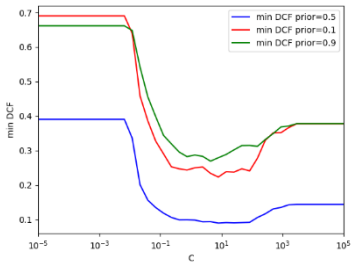
RAW



RBF SVM (π_T=0.5), $\gamma = 10^{-1}$



RBF SVM (π_T=0.5), $\gamma = 10^{-2}$



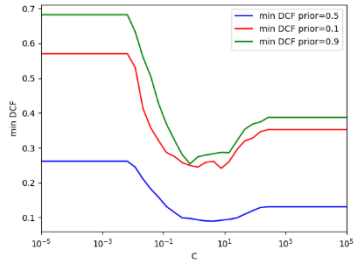
RBF SVM (π_T=0.5), $\gamma = 10^{-3}$

	minDCF		
	$\boldsymbol{\pi = 0,5}$	$\boldsymbol{\pi = 0,1}$	$\boldsymbol{\pi = 0,9}$
y = 0,1	0.202	0.686	0.684
y = 0,01	0.094	0.261	0.310
y = 0,001	0.090	0.223	0.269

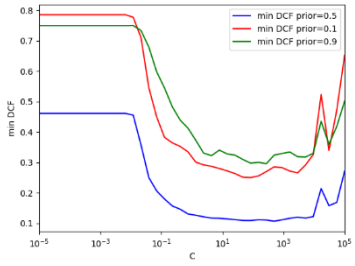
minDCF for each value of y, RAW data

It's obvious that the best value for γ is 10^{-3}, with a value for C of 10. In the following we plot the Z-normalized data to see if it improves the performance of our model and if so for which value of γ and C:

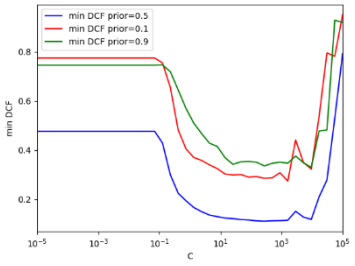
Z-score



RBF SVM (π_T=0.5), $\gamma = 10^{-1}$



RBF SVM (π_T=0.5), $\gamma = 10^{-2}$



RBF SVM (π_T=0.5), $\gamma = 10^{-3}$

	minDCF		
	$\boldsymbol{\pi = 0,5}$	$\boldsymbol{\pi = 0,1}$	$\boldsymbol{\pi = 0,9}$
y = 0,1	0.089	0.241	0.254
y = 0,01	0.106	0.250	0.295
y = 0,001	0.111	0.274	0.330

minDCF for each value of y, Z-score data

Looking at the table above we selected $\gamma = 10^{-1}$ and C = 5. And there are the results obtained:

	RAW(C= 10, $\gamma = 10^{-3}$)			***Z-SCORE(C= 5, $\gamma = 10^{-1}$)***		
	$\boldsymbol{\pi = 0,5}$	$\boldsymbol{\pi = 0,1}$	$\boldsymbol{\pi = 0,9}$	$\boldsymbol{\pi = 0,5}$	$\boldsymbol{\pi = 0,1}$	$\boldsymbol{\pi = 0,9}$
RBF_SVM($\boldsymbol{\pi_T = 0,5}$)	0.088	0.218	0.290	0.089	0.258	0.285
RBF_SVM($\boldsymbol{\pi_T = 0,1}$)	0.111	0.267	0.353	0.110	0.255	0.364
RBF_SVM($\boldsymbol{\pi_T = 0,9}$)	0.104	0.311	0.254	0.106	0.336	0.254

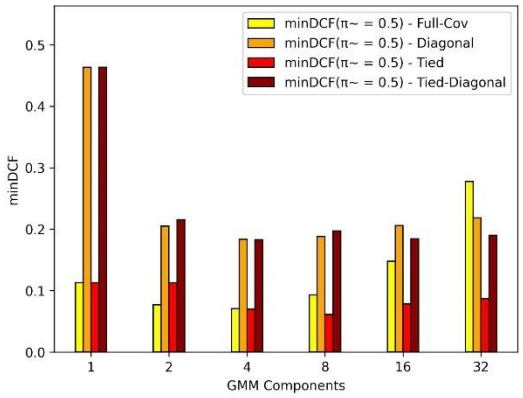
RBF SVM on RAW and Z-normalized data

As a conclusion, for the various SVM models: the linear version has more or less the same performance of the LR, even though this seems to be a little better; QLR performs better than its SVM counterpart (Polynomial SVM) and the RBF version is the best one analyzed till now.

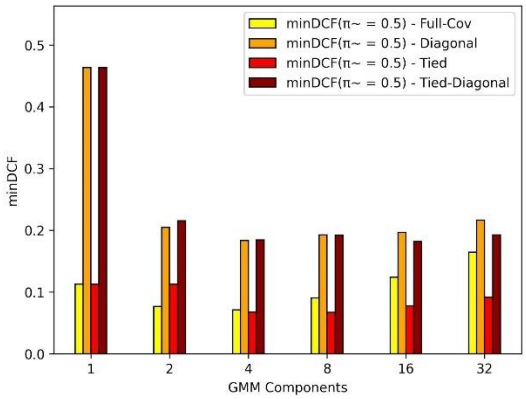
3.4. Gaussian Mixture Models

The last type of classifier we test is the Gaussian mixture model (GMM), capable of better approximating generic distributions because assumes that it is more convenient to represent data with Gaussian distributions composed of multiple components (or clusters). We explore both full covariance and diagonal models, with and without covariance tying even if, as mentioned in the feature analysis and then demonstrated through the results of the Multivariate Gaussian (MVG) models, we expect a good performance with full-covariance and tied models, while diagonal and tied-diagonal models will probably perform poorly (for the same reason explained in the Gaussian Classifier section). Moreover, given that our dataset exhibits three distinct age-related clusters, we anticipate obtaining favourable results with component values ranging from 2 to 4 (we have excluded 3 as an option because GMM models are typically trained with clusters in powers of 2). We decided to make our analysis on a set of components that ranges from 1 to 32 (as powers of 2) using both raw and Z-normalized data. In the following we present a comparison of the four models:

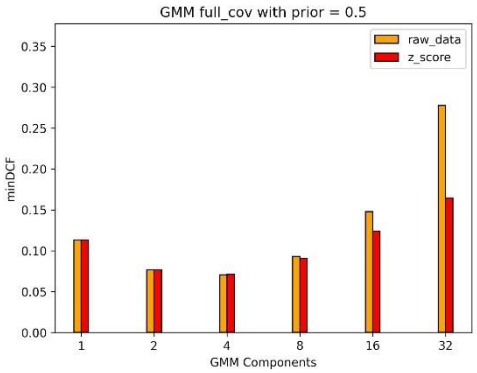
RAW ($\pi_T = 0.5$)

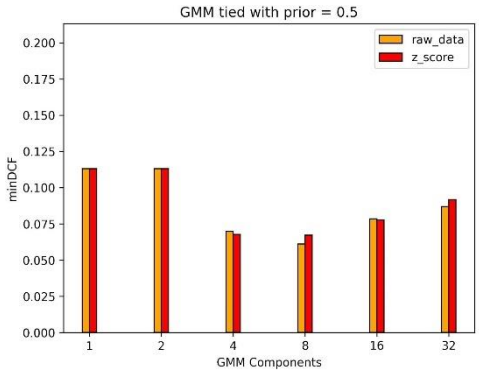


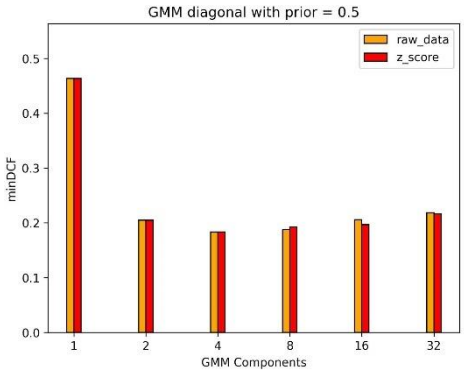
Z-Score ($\pi_T = 0.5$)

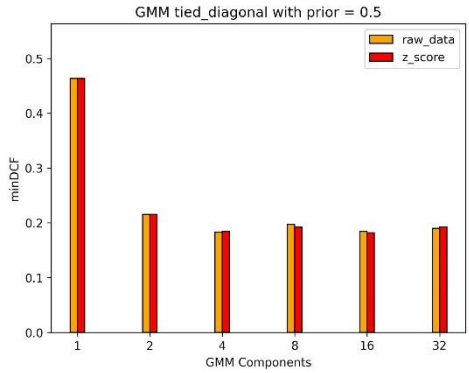


All our expectations were confirmed by the plots: full covariance and full covariance tied models have the best performances overall. To better see the performance's differences between raw and z-normalized data we decided to plot a model per figure with both:









The Z-normalization doesn't seem to enhance the performance of our models, so we report in the following table only the results on RAW data for the best version of each type of GMM model:

	RAW		
	$\boldsymbol{\pi = 0,5}$	$\boldsymbol{\pi = 0,1}$	$\boldsymbol{\pi = 0,9}$
GMM Full-Cov (4 components)	0.071	0.201	0.204
GMM Diagonal (4 components)	0.183	0.461	0.474
GMM Tied (8 components)	0.061	0.216	0.209
GMM Tied-Diag (16 components)	0.184	0.496	0.471

Best GMM components per model with respective minDCF

GMM yields favourable results. The GMM Tied model, trained with raw data and 8 components, emerges as the top-performing model (for our target application), aligning well with the data distributions depicted in the scatter plots above. Although the model trained with eight components is

slightly superior to the one with four components, we opt for the latter as it better captures the natural clusters within the dataset representing three age groups:

	RAW		
GMM Tied (4 components)	0.071	0.201	0.204

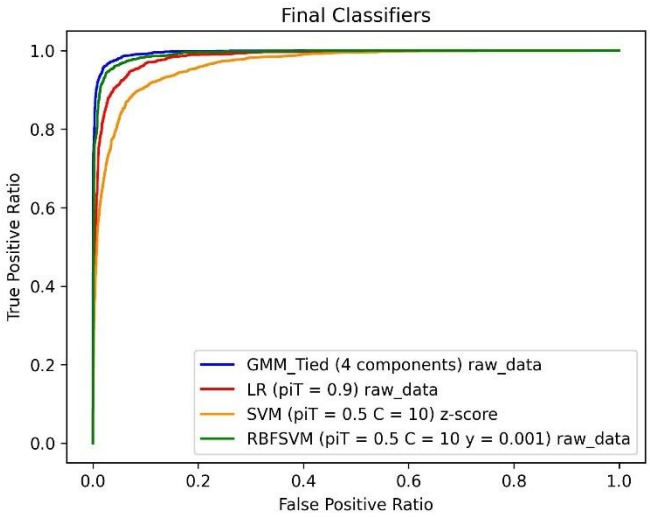
GMM Tied (4 components)

3.5. Best Models – Classification and validation

The best classifiers, one for each category, are:

MODEL	*PRIOR*		
	$\boldsymbol{\pi = 0,5}$	$\boldsymbol{\pi = 0,1}$	$\boldsymbol{\pi = 0,9}$
MVGT(Raw)	0.109	0.299	0.342
LR($\pi_T = 0.9, \lambda = 0$, Raw)	0.111	0.315	0.344
RBSVM($\pi_T = 0.5, C = 10, \gamma = 10^{-3}$, Raw)	0.088	0.218	0.290
GMM Tied (4 components, Raw)	0.071	0.201	0.204

In the next section we are going to calibrate all these models (except MVGT because it's a specific version of the GMM Tied), even if the GMM seems the most promising one. Moreover, since the linear version of SVM had similar results to the LR model and RBSVM with the Z-score is equivalent to the one trained with RAW data, we will calibrate them too for completeness. Before going into the next section here it is the ROC Curve of the various models:



In this way we have another visual confirmation that the GMM model is the best one in general.

4. Calibration

To evaluate the different models, so far, we have used only the minimum cost of detection (minDCF). The minDCF assesses the potential capabilities of our systems to make informed decisions. However, in practical scenarios, we must make decisions without knowing the optimal rule for our evaluation data. In our case, we need to map our scores to class labels, but we lack knowledge about the optimal score threshold. Consequently, we aim to evaluate the quality of the decisions we can actually make using recognizer scores. To assess the decision-making capabilities of our systems, we will consider a metric called Actual DCF (actDCF), which represents the normalized cost we would incur based on our actual decisions. To achieve this, we attempt to employ a score calibration function involving the computation of a mapping transformation from uncalibrated scores *s* to calibrated scores. Assuming that this function is linear in s, it can be expressed as $f(s) = \alpha s + \gamma$. f(s) can be interpreted as the likelihood ratio (LLR) for the two-class hypotheses:

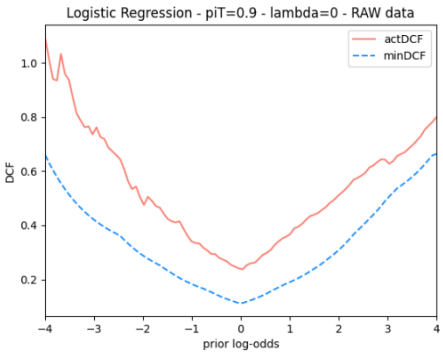
$$\log \frac{P(C = \mathcal{H}_T|s)}{P(C = \mathcal{H}_F|s)} = \alpha s + \gamma + \log \frac{\tilde{\pi}}{1-\tilde{\pi}} = \alpha s + \beta$$

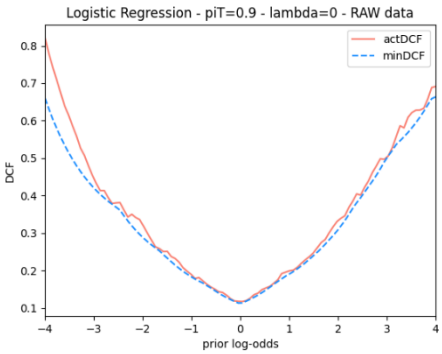
To obtain the calibrated score, we will need to compute:

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

To estimate the parameters of the calibration function, we will use a K-Fold approach (with k=5), since the number of samples we have is limited.

4.1. Logistic Regression





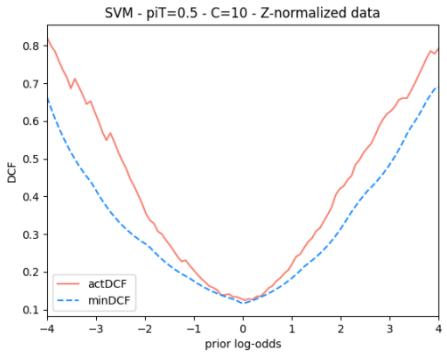
We can see an important improvement through calibration, this is confirmed by the following results:

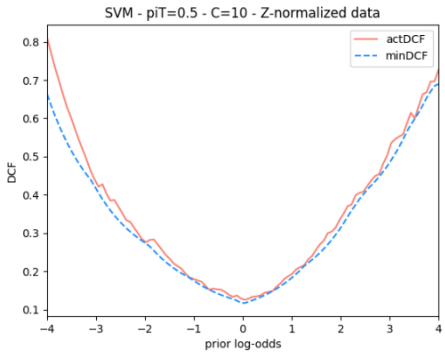
	Uncalibrated actDCF			*Calibrated actDCF*		
	$\boldsymbol{\pi = 0,5}$	$\boldsymbol{\pi = 0,1}$	$\boldsymbol{\pi = 0,9}$	$\boldsymbol{\pi = 0,5}$	$\boldsymbol{\pi = 0,1}$	$\boldsymbol{\pi = 0,9}$
LR	0.234	0.559	0.546	0.115	0.342	0.360

	minDCF		
	$\boldsymbol{\pi = 0,5}$	$\boldsymbol{\pi = 0,1}$	$\boldsymbol{\pi = 0,9}$
LR($\pi_T = 0.9, \lambda = 0$, Raw)	0.111	0.315	0.344

4.2. SVM

As we mentioned before we are going to calibrate 3 different models, SVM linear and RBSVM for Raw and Z-scored data:



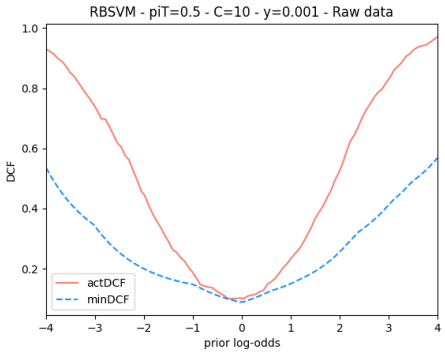


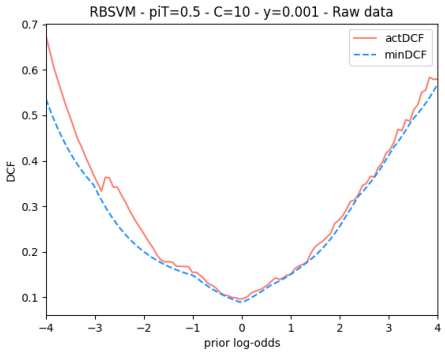
	Uncalibrated actDCF			*Calibrated actDCF*		
	$\boldsymbol{\pi = 0,5}$	$\boldsymbol{\pi = 0,1}$	$\boldsymbol{\pi = 0,9}$	$\boldsymbol{\pi = 0,5}$	$\boldsymbol{\pi = 0,1}$	$\boldsymbol{\pi = 0,9}$
SVM	0.125	0.421	0.457	0.125	0.310	0.377

	minDCF		
	$\boldsymbol{\pi = 0,5}$	$\boldsymbol{\pi = 0,1}$	$\boldsymbol{\pi = 0,9}$
SVM($\pi_T = 0.5$, Z-Score)	0.115	0.293	0.355

The table above shows us that it's not always necessary to calibrate the models, for example for our target application the calibration doesn't bring any benefit, differently for the two other priors.

For the RBF version (on raw data):

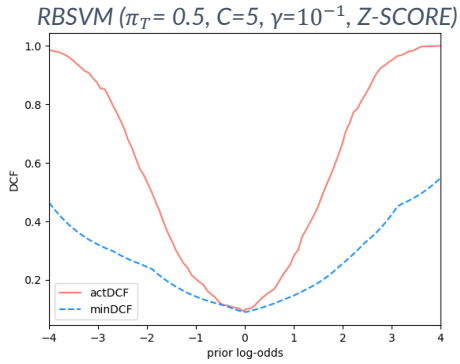


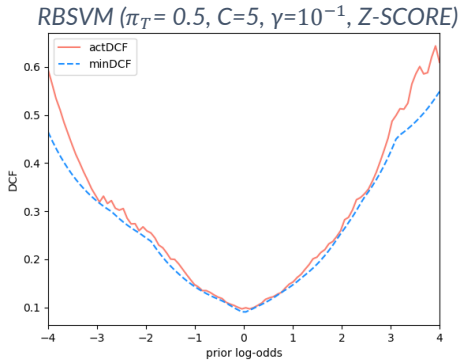


	Uncalibrated actDCF			*Calibrated actDCF*		
	$\boldsymbol{\pi = 0,5}$	$\boldsymbol{\pi = 0,1}$	$\boldsymbol{\pi = 0,9}$	$\boldsymbol{\pi = 0,5}$	$\boldsymbol{\pi = 0,1}$	$\boldsymbol{\pi = 0,9}$
RBF_SVM(Raw)	0.100	0.515	0.612	0.095	0.273	0.306

	minDCF		
	$\boldsymbol{\pi = 0,5}$	$\boldsymbol{\pi = 0,1}$	$\boldsymbol{\pi = 0,9}$
RBSVM($\pi_T = 0.5, C = 10, \gamma = 10^{-3}$, Raw)	0.088	0.218	0.290

On Z-scored-data:





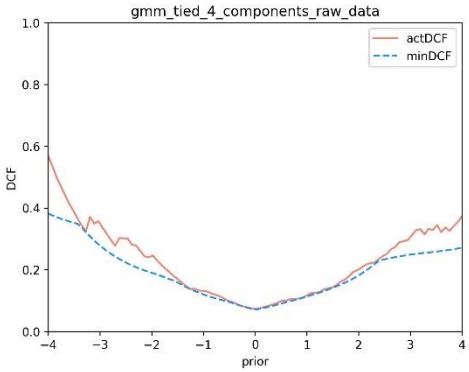
	Uncalibrated actDCF			*Calibrated actDCF*		
	$\boldsymbol{\pi = 0,5}$	$\boldsymbol{\pi = 0,1}$	$\boldsymbol{\pi = 0,9}$	$\boldsymbol{\pi = 0,5}$	$\boldsymbol{\pi = 0,1}$	$\boldsymbol{\pi = 0,9}$
RBF_SVM(Z-Score)	0.093	0.608	0.761	0.093	0.269	0.300

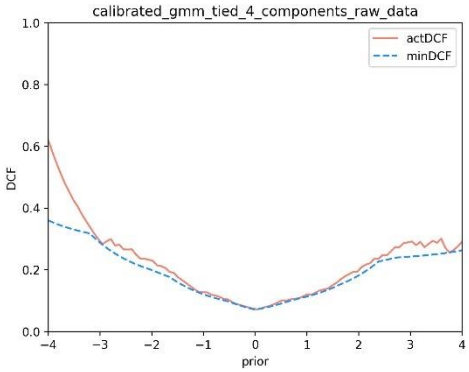
	minDCF		
	$\boldsymbol{\pi = 0,5}$	$\boldsymbol{\pi = 0,1}$	$\boldsymbol{\pi = 0,9}$
RBSVM($\boldsymbol{\pi_T = 0.5, C = 5, \gamma = 10^{-1}}$, Z-Score)	0.089	0.258	0.285

As for the linear version, the calibration is more impactful for the two unbalanced prior, while for our target application it doesn't improve that much. Since our Test set is unbalanced we chose to use calibration for these models during the evaluation phase.

4.3. Gaussian models

We end this section with the most promising model, GMM Tied with 4 components (the label on the x-axis should be "log prior odds" not "prior"):





	Uncalibrated actDCF			*Calibrated actDCF*		
	$\boldsymbol{\pi = 0,5}$	$\boldsymbol{\pi = 0,1}$	$\boldsymbol{\pi = 0,9}$	$\boldsymbol{\pi = 0,5}$	$\boldsymbol{\pi = 0,1}$	$\boldsymbol{\pi = 0,9}$
GMM Tied(Raw)	0.073	0.255	0.224	0.072	0.218	0.219

	minDCF		
	$\boldsymbol{\pi = 0,5}$	$\boldsymbol{\pi = 0,1}$	$\boldsymbol{\pi = 0,9}$
GMM Tied (4 components, Raw)	0.071	0.201	0.204

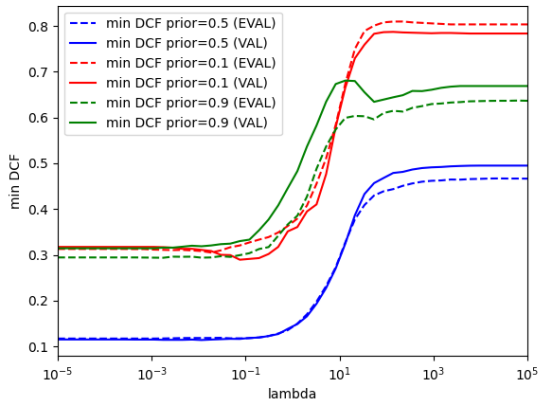
The scores do not need calibration; indeed, the transformation seems to not bring any particular benefit to the model. As we can see from the results above only the unbalanced application with $\pi_T = 0.9$ slightly improved.

5. Evaluation

Now we move on to the testing phase and check if the choices made previously during the Valuation phase were optimal or not. We will carry out the tests only for the 3 best models: GMM Tied (4 components), Logistic Regression and RB SVM on RAW data. The training and the test will be carried out not through K-Fold but directly using the Test set and the whole Training set.

5.1. Logistic Regression

We first consider the logistic regression models plotting λ for the same interval that we use during the training. Our best model, during the training was LR trained with π = 0.9 so we will consider the plot for this type of model:



LR on evaluation set with piT=0.9

The trends of the models are practically the same if we make a comparison with results obtained during Evaluation and those during Validation, and this is confirmed by the following results obtained on the Evaluation set for LR:

MODEL	*PRIOR*		
	$\boldsymbol{\pi = 0,5}$	$\boldsymbol{\pi = 0,1}$	$\boldsymbol{\pi = 0,9}$
LR($\boldsymbol{\pi_T = 0.9, \lambda = 0}$, Raw) minDCF	0.123	0.341	0.279
LR($\boldsymbol{\pi_T = 0.9, \lambda = 0}$, Raw) actDCF - calibrated	0.127	0.350	0.295

During the validation and classification phase we have written that to balance overfitting and underfitting a good value for λ could have been 0.1. So, we tried this value on the evaluation set and these are the results obtained:

MODEL	*PRIOR*		
	$\boldsymbol{\pi = 0,5}$	$\boldsymbol{\pi = 0,1}$	$\boldsymbol{\pi = 0,9}$
LR($\boldsymbol{\pi_T = 0.9, \lambda = 0}$, Raw) minDCF	0.120	0.301	0.289
LR($\boldsymbol{\pi_T = 0.9, \lambda = 0}$, Raw) actDCF-calibrated	0.121	0.306	0.314

We can say that overall our initial decision made on λ lead us to a suboptimal solution for our LR model.

5.2. SVM

For the Support Vector Machine model, we don't show any plot because they are very similar to the one obtained in the training phase, furthermore here it is the table on the results obtained on the evaluation set only for the raw data because the improvement brought by the Z-normalization during the training and calibration phase are not enough good to justify the request for further confirmation.

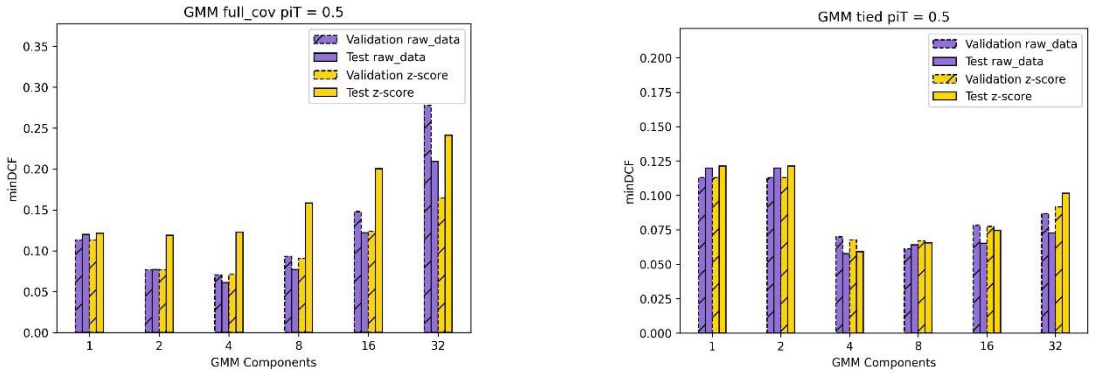
MODEL	*MINDCF*			*ACTDCF*		
	$\boldsymbol{\pi = 0,5}$	$\boldsymbol{\pi = 0,1}$	$\boldsymbol{\pi = 0,9}$	$\boldsymbol{\pi = 0,5}$	$\boldsymbol{\pi = 0,1}$	$\boldsymbol{\pi = 0,9}$
RBSVM($\pi_T = 0.5, C = 10, \gamma = 10^{-3}$, Raw) - calibrated	0.081	0.221	0.215	0.086	0.225	0.251

Comparison between minDCF and actDCF (calibrated) fot evaluation set RBF SVM

The results are as we detected during the analysis on the training set.

5.3. GMM

The last model we consider for the evaluation is the Gaussian Mixture Model. We will focus our attention only on the 2 best versions obtained, i.e. Full-Cov and Tied, and to asses the quality of our decisions we decided to plot the results for both RAW (the one we chose as best model) and Z-scored data:

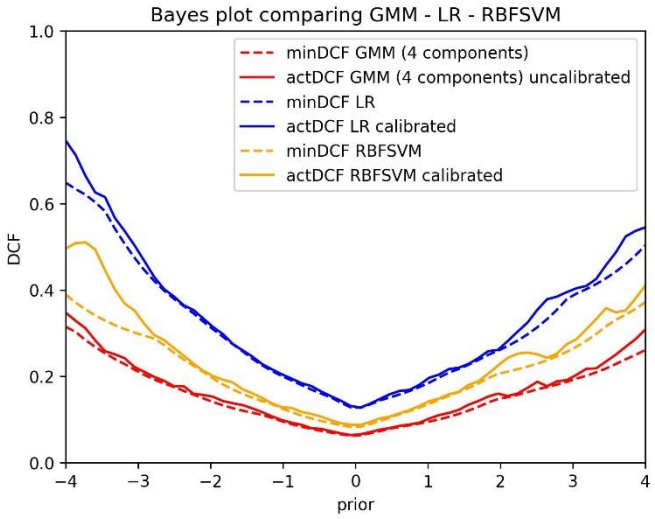


As we can see from the plots, Z-normalization performs slightly worse compared to raw data, and the test scores of the GMM full-cov model are notably higher than the other model. Our GMM Tied model demonstrates good results on the test set. Even though the 8-component GMM performed better during the validation phase, it was outperformed by the 4-component GMM during the test phase. As we explained in the GMM section, this can be attributed to the presence of three distinct age clusters. In the following table the results obtained with both models not calibrated (as we chose in the calibration phase):

MODEL	*RAW*			*Z-SCORE*		
	$\boldsymbol{\pi = 0,5}$	$\boldsymbol{\pi = 0,1}$	$\boldsymbol{\pi = 0,9}$	$\boldsymbol{\pi = 0,5}$	$\boldsymbol{\pi = 0,1}$	$\boldsymbol{\pi = 0,9}$
GMM Tied (4 components)	0.058	0.178	0.160	0.059	0.167	0.192
GMM Full-Cov (4 components)	0.061	0.156	0.156	0.123	0.425	0.248

It turned out that the 2 models are pretty equivalent for our evaluation set and overall the Full Covariance version seem to have slightly better performances.

We end this section by plotting the minDCF and actDCF of the best 3 models we chose on the evaluation set:



6. Conclusion

The results obtained show that the validation and evaluation population have similar distributions. Since we have more samples on the evaluation set the 3 different group ages seem to be even more outlined and that's why the GMM model performs better on it than on the validation set. The choices and strategies used during the training phase have proven effective when applied to test data. Using the Gaussian Mixture Model (GMM) tied, with 4 components, has demonstrated its effectiveness in generating well-tuned scores in various application ambiens: in our main application ($\boldsymbol{\pi}$ = 0.5), we achieved a remarkably low DCF cost of 0.058; and when applied to target application with $\boldsymbol{\pi}$ = 0.1 and $\boldsymbol{\pi}$=0.9 we obtained decent results (actDCF = 0.178 and actDCF = 0.160). Although the results are good, from what we have analyzed the best model on the evaluation set overall is the GMM full-covariance applied on raw data: it got slightly worse performances on our target application (0.061) but it improved for the others ($\boldsymbol{\pi}$ = 0.1 → 0.156, $\boldsymbol{\pi}$ = 0.9 → 0.156).