Politecnico di Torino

GENDER IDENTIFICATION PROJECT

Master of Science in Computer Engineering

Machine learning and Pattern recognition Exam

2022/2023

Gelsi Alessandro

S303525

Abstract

This report is dedicated to the analysis, employing a range made of different algorithms, of a dataset consisting of low-level images related to males and females. The goal is to determine the models that achieve the highest classification performance. Initially it was conducted an examination of the dataset's features, followed by the exploration of various classifiers, including Multivariate Gaussian Models (MVG), Logistic Regression (both linear and quadratic), Support Vector Machine (Linear, RBF, and Quadratic), Gaussian Mixture Models, and Fusion. A validation dataset is derived from the training dataset using K-fold cross-validation, in order to find the best hyperparameters for each model. The performances are evaluated at first taking into account minimum detection cost function (min DCF), followed then also by considerations of actual DCF and score calibration. Finally, the models equipped with the chosen hyperparameters were tested on evaluation set, made of unseen data.

It will be demonstrated that all classifiers yield overall good results on the provided dataset. As a best model it was selected GMM Full-tied covariance model with 4 components trained over Z-score pre-processed features and without PCA.

Index

Dataset overview and analysis

The training dataset is made of 2400 samples, comprising 720 males and 1680 females. These are made extracting speaker embeddings from face images. A speaker embedding is a small-dimensional fixed sized representation of an image, where features are continuous values that represent a point in the m-dimensional embedding space. The dataset results in a substantial bias towards females, accounting for 70% of the dataset. Each sample is made of 12 features that do not have a direct physical interpretation. It is also known that the samples belong to three distinct age groups, each characterized by a different distribution of embeddings. We do not have availability of age information. The test dataset instead is characterized by 6000 samples, with 4200 males and 1800 females. Dataset is imbalanced, with the training set that has significantly more female samples, whereas the test set has significantly more male samples.

Before involving any classifier, it is necessary first to perform an analysis of the dataset features. The mean (μ) and standard deviation (σ) of each feature for the training dataset are:

$$\mu = [\,-3.1,\ \ 13.2, -10.4, -1.2, -3.7, -0.5, -6.5, -9.0,\ \ 5.8, -3.0,\ \ 5.1,\ \ 14.8\ ]$$

$$\sigma = [\ \ 8.0, 10.6,\ \ 6.8,\ \ 4.5,\ \ 8.5,\ \ 9.4,\ \ 6.5,\ \ 6.5,\ \ 10.4,\ \ 4.3,\ \ 6.2\ ]$$

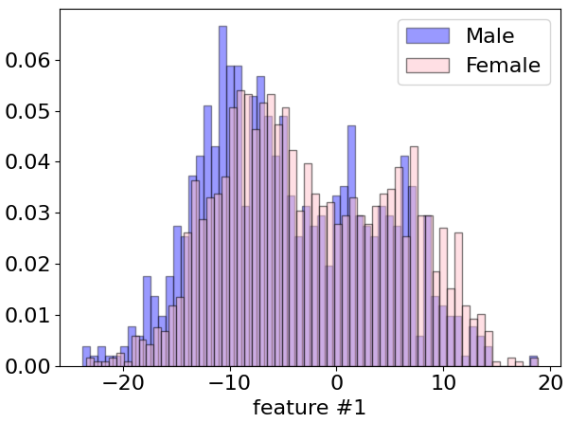
The features do not exhibit significantly different scales, there is not a large difference between their ranges. A technique called Z-normalization generally is useful to bring all the features on the same scale, by centering the feature columns at mean 0 and with standard deviation 1. It is possible to consider it inside this study, despite not expecting substantial improvements. So, in parallel with the analysis of the raw features, an analysis of the normalized features was conducted by normalizing before applying any operation. Each sample of the training set has been transformed through the expression:

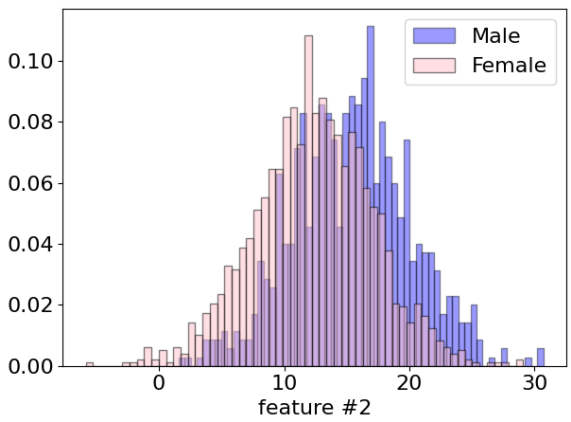
$$\mathrm{x}' = \frac{x - \mu}{\sigma}$$

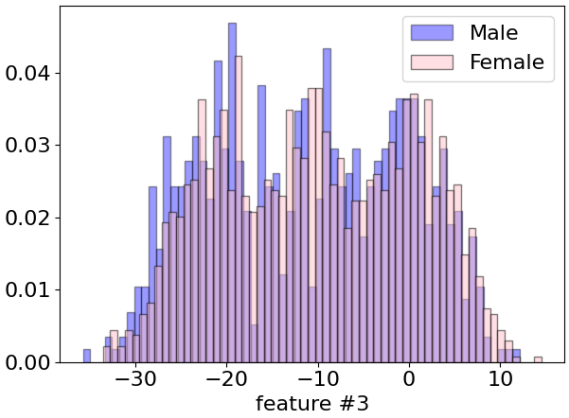
Where $\mathbf{x}'$ is the sample after the Z-score normalization, while $\mathbf{x}$ is the original sample in the data set.

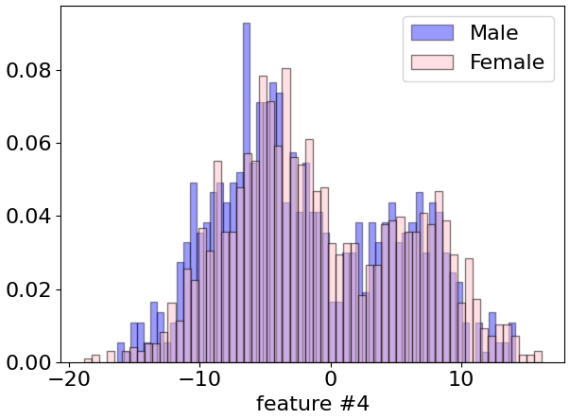
Histograms

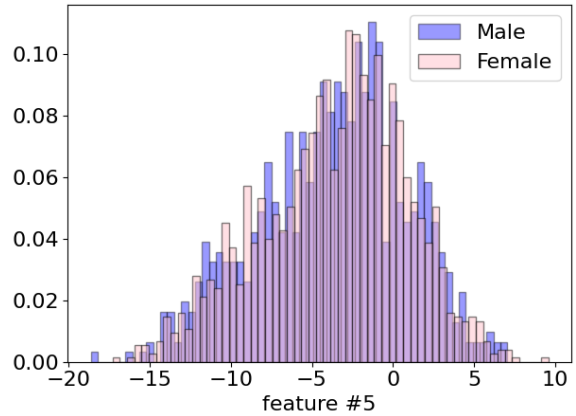
The initial step involves plotting histograms of each dataset feature to examine their distributions.

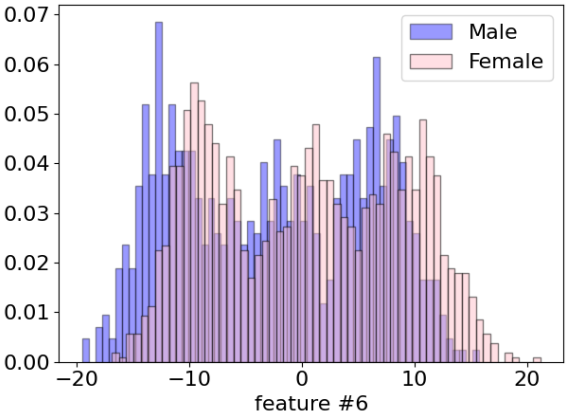


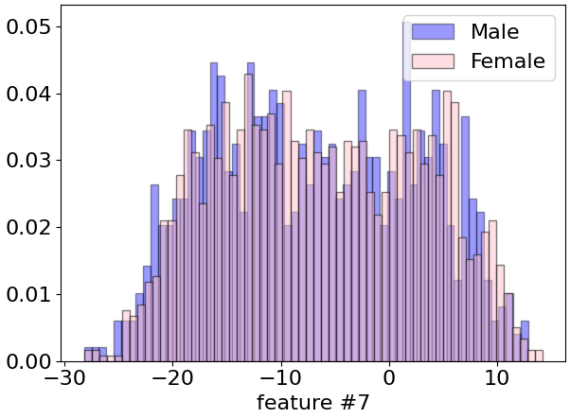


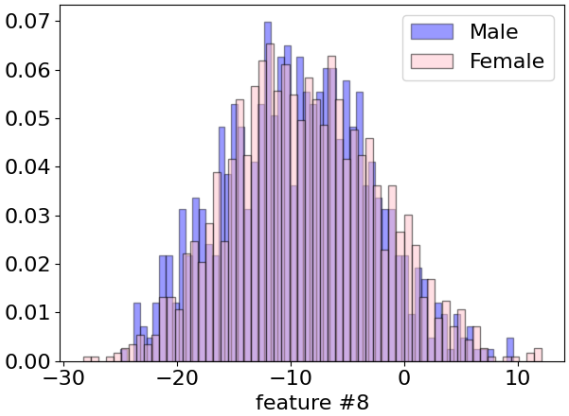

Male
Female
feature #3

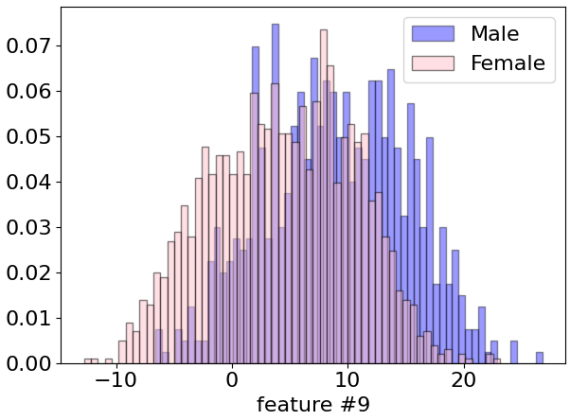

Male
Female
feature #4

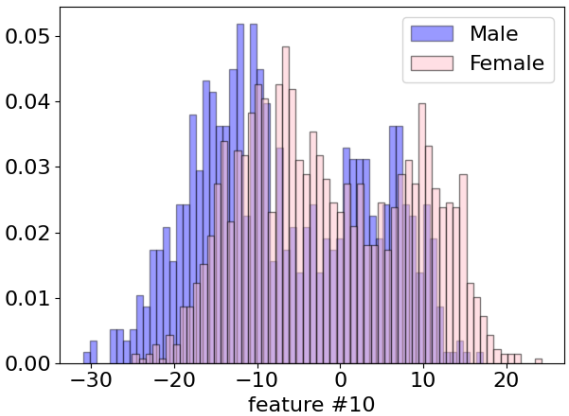

Male
Female
feature #5

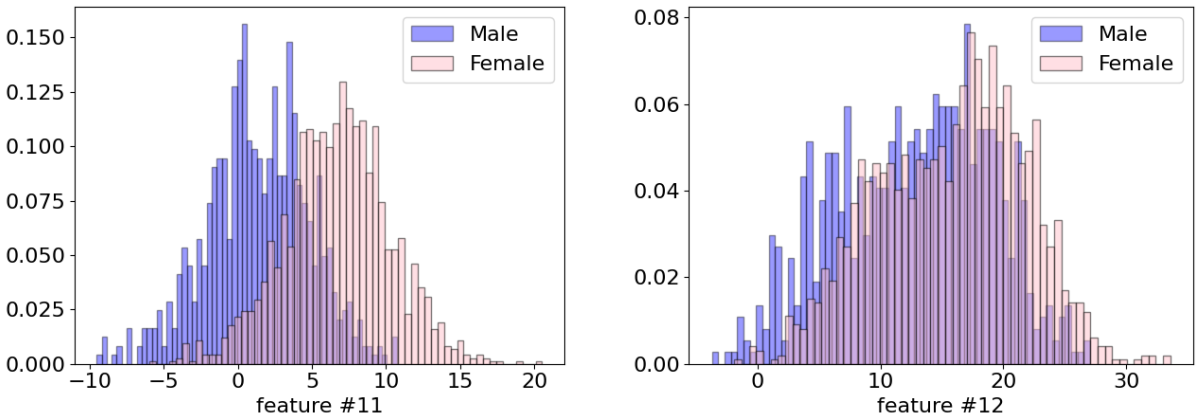

Male
Female
feature #6


Male
Female
feature #7


Male
Female
feature #8


Male
Female
feature #9

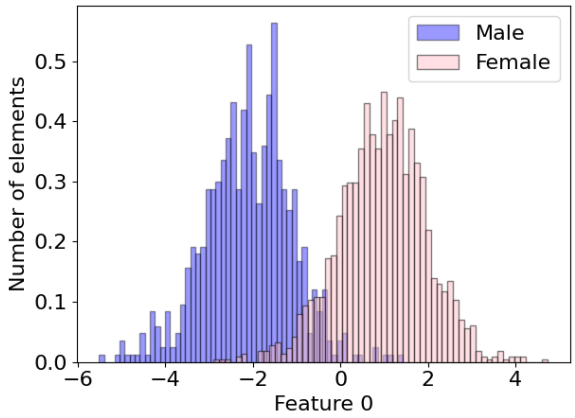

Male
Female
feature #10



Histograms of features for different classes

We can see that there are some distributions, such as the ones relative to features #5, #8, #2, that recall directly to a gaussian density. It is also important to observe that some plots, like for example plot of feature #3, #6 and #7, are similar to a distribution made of three gaussian. This can be associable to the 3 groups of ages from where the features are extracted of. Altogether, the distribution of individual features is consistent across both classes, but there are certain distributions that enable us to differentiate between classes in an easier way. This happens for example in the histogram relative to feature #9 and #11, where it is possible to observe the most distinguishable feature distributions in our dataset.

Linear discriminant analysis



LDA application

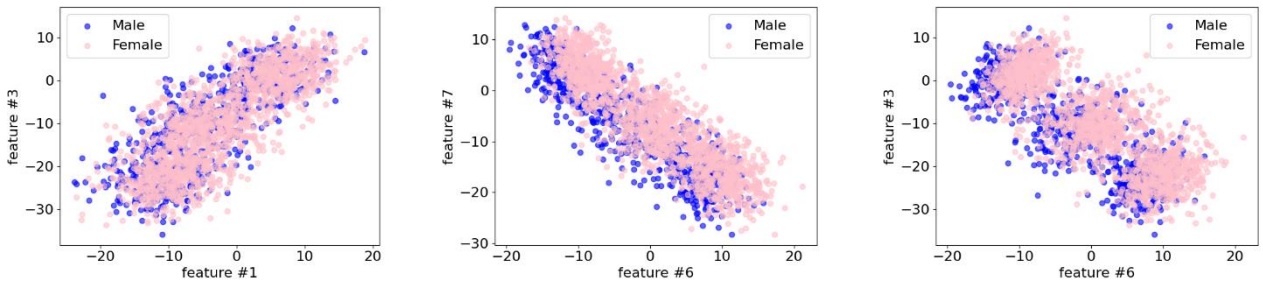
Linear Discriminant Analysis (LDA) is a transformation that enables us to assess whether the features exhibit linear discriminability. It is generally used as a dimensionality reduction technique. LDA is useful to identify C-1 directions where to project our features. In practice, we can use to determine if a linear and/or Gaussian model might perform better than a non-Gaussian and/or nonlinear one.

As evident from the plot above, the classes within the dataset are discriminable. However, upon closer examination, we can observe the presence of a relatively small region where the potential for classification errors may arise. We can analyze the scatter plot to better understand features relationships.

Scatter plots

The analysis continues by leveraging scatter plots, which are particularly useful to visualize the relationship between two continuous variables. They can help identify patterns, trends, correlations, or clusters within the data. By examining the distribution and dispersion of the dots, it is possible to see how the variables interact with each other. For our dataset, these plots are aligned to the one present in the gaussian model, and for this reason, we expect that gaussian model are able to perform well on this kind of data.

The following images contains some of the most significant plot. For example, in the scatter plot relative to feature #6 it is evident the presence of three clusters. It can be associable again to a gaussian distribution with more components, and directly related to the three groups of age from where the dataset sample are taken.



Scatter plots of features of different pairs of classes

Correlation

A way to analyze features interaction is to compute the correlation of features. This is useful also to understand if PCA (Principal Component Analysis) could be useful and how many features can be discarded. Pearson correlation coefficient can be used to measure correlation between two features, and it can be computed as:

$$\frac{Cov(X,Y)}{\sqrt{Var(x)}\sqrt{Var(y)}}$$

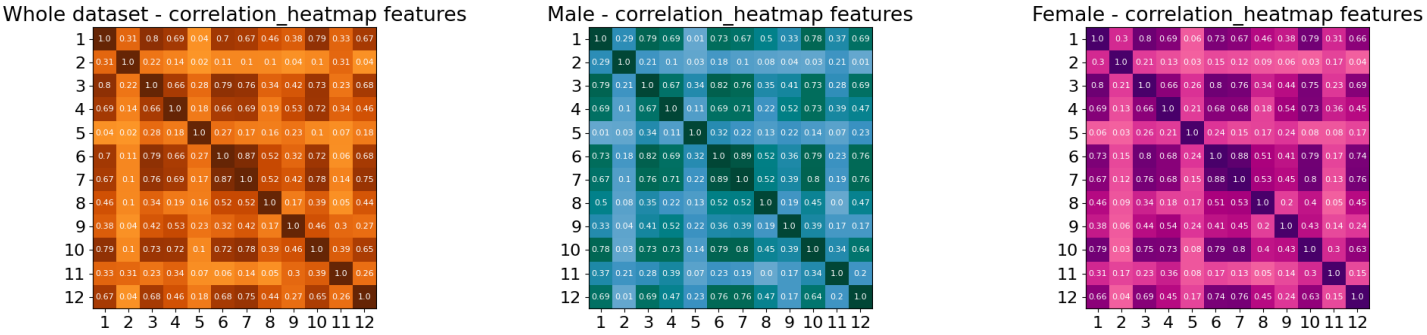
In this analysis only the absolute value of Pearson correlation is considered, because we are only interested to understand if there is correlation or not:

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

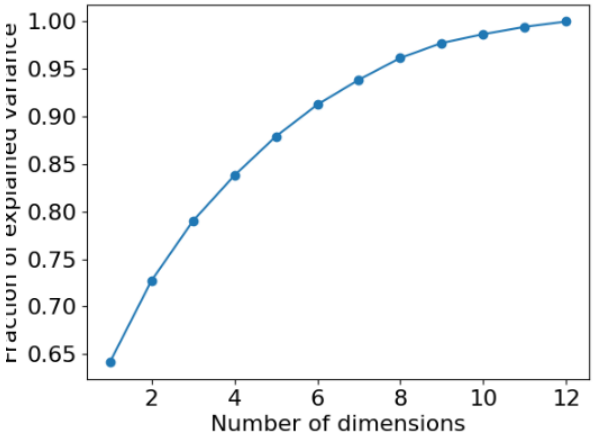
The absolute value of Pearson correlation coefficient can take value between 0 and 1. If 0 it means that the two considered features are uncorrelated, while 1 means that the features are completely correlated (one feature is the scaled version of the other).
To visualize the correlation between features we employed heatmaps. In the following heatmaps, darker colors indicate a strong correlation between two features, while lighter colors suggest a weaker correlation between the two features.

We plot heatmaps for the correlation considering the whole data training samples, training samples belonging to Male class and training samples belonging to Female class.

Correlation of raw features using abs of Pearson correlation.



Focusing to the heatmap of the whole dataset, it becomes evident that some features exhibit strong correlations, such as 1-10 and 3-6, while others appear uncorrelated, like 5-11. Most of the features demonstrate mild correlation between each other, with coefficients hovering around 0.4 to 0.6. This observation hints that it is possible to gain advantages by transforming our data from a 12-dimensional space to a 10-dimensional one, effectively reducing the number of parameters to be estimated. So we can to apply PCA, that is a technique useful in order to reduce the dimensionality of a dataset. PCA finds a m-dimensional subspace, that is a set of directions over which to project our data set points. More in details PCA finds map projection that minimize the average reconstruction error (i.e. the reconstructed points are as close as possible to the original points). The dimension of the subspace (m) is an hyperparameter that needs to be tuned using a validation set.



Explained variance of PCA

In the following steps we considered results of classifiers with hyperparameter m up to 10, as suggested in the explained variance graph, where it is possible to notice that we will maintain the

96% of the variance by removing 1 or 2 dimensions, maintaining the most of the discriminative information.

Moving the focus over single class heatmaps, we can notice a strong similarity between the two plots, meaning that the gaussian model (Multivariate Gaussian Model-MVG and Tied-MVG) will behave in a similar way in terms of performances. At the same time the reasonable correlation between the features should bring the gaussian models based on the Naïve-bayes assumption to give worst performances. Anyway, we will evaluate also the Gaussian classifiers with diagonal covariance matrix to compare them with other Gaussian classifiers.

Validation approach

In the following we took into consideration different machine learning models for classification. Each of them is first trained and then validated on the training dataset and we obtained costs.

Generally, for binary problems, the cost function can be divided into two components: the goodness of classifier itself and the goodness of the threshold chosen. At the beginning the threshold problem is not considered, we just need to know which classifier better discriminate the scores, for this reason minimum detection cost (min DCF) will be used to measure the performance.

So models are trained using training set (without validation set samples) then scores and min DCF are computed on validation samples. The model can be trained with different combinations of hyperparameters, results (min DCF computed on the scores of validation samples) of the same model with different hyperparameters are compared to find the most promising combination of hyperparameters.

K-fold cross-validation operates by dividing the training dataset into K folds. In each iteration, K-1 of these folds are utilized for training the model, while the remaining fold is used for validation. This process is repeated K times, and during each iteration, scores are computed for the validation samples. After K iterations, scores have been computed for every sample in the training set, allowing us to calculate the minimum Detection Cost Function (min DCF) using this set of scores.

Once selected all the hyperparameters, the final classifier will be obtained training again over the whole training dataset. Altogether K-fold cross-validation has a significant drawback: it necessitates training and validating models K+1 times, which can demand a substantial amount of computational time.

Generally if feasible the K-fold cross-validation leads to more robust results rather than single split, so to measure the performance of the different classifiers we will employ K-fold cross-validation over the single-fold. Data has been shuffled before splitting, so that the data of different folds are homogeneous.

Because of the increase of the amount of time required to estimate model parameters with higher value of K, we will consider K= 5.

We will consider different applications (π_T, C_{fp}, C_{fn}), reduced to the effective prior $\tilde{\pi}$. In particularly the applications considered are:

- Primary application: uniform prior application : (π_T, C_{fp}, C_{fn}) = (0.5, 1, 1)
- Unbalanced application: (π_T, C_{fp}, C_{fn}) = (0.1, 1, 1)
- Unbalanced application:(π_T, C_{fp}, C_{fn}) = (0.9, 1, 1)

Multivariate Gaussian classifier (MVG)

The first family of models that we consider in this analysis are MVG classifiers. We try different pre-processing techniques, such as Z-Score and PCA, to determine if they are effective, also in configurations based on a combination of them.

MVG are generative models**,** based on the idea of trying to model the class distribution of observed samples. MVG classifiers assume that both the training set and evaluation samples are independent and identically distributed *(i.i.d.)* given a set of parameters θ. In particular, Gaussian distribution for a sample given the class is assumed:

$$(X_i|C_i = c, \theta) \sim (X_t|C_t = c, \theta) \sim (X|C = c, \theta) \sim \mathcal{N}(\mu c, \Sigma c)$$

Using the maximum likelihood estimation is it possible to estimate the model parameters:

$$\mu_c{}^* = \frac{1}{N_c} \sum_{i|c_i=c} x_i$$

$$\Sigma_c{}^* = \frac{1}{N_c} \sum_{i|c_i=c} (x_i - \mu_c)(x_i - \mu_c)^T$$

Nc is the number of samples belonging to class c. The log-likelihood ratio (llr) acts as a score:

$$s(\boldsymbol{x}_t) = \text{llr}(\boldsymbol{x}_t) = \log \frac{f_{X|C}(\boldsymbol{x}_t \mid c_1)}{f_{X|C}(\boldsymbol{x}_t \mid c_0)}$$

The models we consider are different in terms of covariance matrix computation:

• **Full Covariance matrix** (referred as Full-Cov): computes covariance matrix (Σc^*) without any simplifications. This model is expected to be robust if the number of training samples is far bigger than number of dimensions of samples.

• **Diagonal Covariance matrix – Naïve Bayes** (referred as Diag-cov): Multivariate Gaussian classifier with diagonal covariance matrices where the diagonal element of row-*i* are the variances of the feature-*i* of the training samples. Diag-Cov is a diagonal version of the original full covariance matrix. This model works well in scenario where features are enough uncorrelated. In our dataset there is

a reasonable correlation between features, so the Gaussian classifier with diagonal hypothesis is expected to have worst results than full covariance Gaussian classifier.

• **Tied Covariance matrix**: it assumes that gaussian parameters of different classes are related one to each other, and so that covariance matrices of different classes are the same. There is only one covariance matrix corresponding to a weighted average of empirical covariance matrix for each class. This model works well when there are classes with few samples.

Our analysis takes into account all the previously mentioned model and in the following the respective results are reported:

Table 1: Min DCF for MVG with K-fold cross validation (K=5)

	RAW			Z-score		
Model	**π= 0.5**	**π= 0.1**	**π= 0.9**	**π= 0.5**	**π= 0.1**	**π= 0.9**
Full-cov	0.113	0.297	0.350	0.113	0.297	0.350
Diag-cov	0.463	0.770	0.777	0.463	0.770	0.777
Tied full-cov	0.109	0.299	0.341	0.109	0.299	0.341
Tied diag-cov	0.457	0.769	0.780	0.457	0.769	0.780

Table 2: Min DCF for MVG with K-fold cross validation (K=5) and PCA(m=11)

	RAW- PCA(11)			Z-score -PCA(11)		
Model	**π= 0.5**	**π= 0.1**	**π= 0.9**	**π= 0.5**	**π= 0.1**	**π= 0.9**
Full-cov	0.117	0.308	0.349	0.121	0.311	0.358
Diag-cov	0.126	0.324	0.370	0.124	0.311	0.348
Tied full-cov	0.118	0.288	0.355	0.118	0.298	0.355
Tied diag-cov	0.124	0.302	0.360	0.123	0.294	0.354

Table 3: Min DCF for MVG with K-fold cross validation (K=5) and PCA(m=10)

	RAW- PCA(10)			Z-score -PCA(10)		
Model	**π= 0.5**	**π= 0.1**	**π= 0.9**	**π= 0.5**	**π= 0.1**	**π= 0.9**
Full-cov	0.163	0.401	0.492	0.187	0.406	0.537
Diag-cov	0.168	0.448	0.468	0.184	0.434	0.545
Tied full-cov	0.161	0.392	0.474	0.183	0.427	0.535
Tied diag-cov	0.170	0.396	0.479	0.182	0.421	0.543

Overall the results are aligned with the previous expectation. Relatively to the Naïve-Bayes assumption, it is observable that, as expected, it is not so successful. This because of the presence

of a reasonable correlation between the features, which is in contrast with the suitable working conditions for this classifier.

A possible solution in order to avoid this problem is to leverage Principal Component Analysis (PCA). PCA preserves the principal discriminant information in the leading directions, and in this way Full-cov MVG and Tied full MVG are in condition not to lose information and to keep performances the same. From the theoric definition we know that Tied full MVG is similar to the Full covariance one, but with the particularity to have the same covariance matrix among the classes. So, if the classes are characterized by a similar distribution and correlation between features, the results of Tied MVG is similar to the full one.

The Z-score normalization pre-processing technique provides results equals to the one of the raw features, because it simply subtracts the mean and scale by the standard deviation each feature. MVG are invariant to this kind of transformation.

Taking into account the results relative to dimensionality reduction, it can be observed that results for the 11 dimensions space are similar to the 12 one. As expected after observing the explained variance plots, in this way it is possible to maintain a good fraction of variance.

Best results are obtained without applying PCA, even if the ones obtained with PCA(11) are quite similar. Furthermore, the features pre-processed with Z-score normalization provide results slightly better than the raw ones. Given the limited effectiveness of PCA for generative models, we are going to focus on the whole set of features, but still using PCA in the following analysis to be sure.

To conclude, the best MVG classifier option is Tied Full covariance MVG, trained without applying PCA.

Logistic Regression

In the following we refers instead to an analysis of discriminative models**,** starting from Logistic Regression. Discriminative models, differently from the generative ones, try to directly model the class posterior distribution, rather than modelling the distribution of observed samples (generative models). It can be expected that PCA has limited effects on Logistic Regression models, since discriminative models does not require specific assumptions on data distribution. Despite this, in the following, we continue taking into consideration results on data pre-processed with PCA and Z-score normalized features to check it.

Linear Logistic Regression

The first logistic regression model that we consider is the regularized linear one. It has 2 parameters *(w,b)* obtainable by minimizing the following expression :

$$J(\boldsymbol{w}, b) = \frac{\lambda}{2} \| \boldsymbol{w} \|^2 + \frac{1}{n} \sum_{i=1}^{n} \log\left(1 + e^{-z_i(\boldsymbol{w}^T \boldsymbol{x}_i + b)}\right), \qquad z_i = \begin{cases} 1 & \text{if } c_i = 1 \\ -1 & \text{if } c_i = 0 \end{cases} (\text{ i.e. } z_i = 2c_i - 1)$$

$J(w, b)$ expression refers to the average cross-entropy between the distribution of observed and predicted labels, plus a regularization term $\frac{\lambda}{2} \|w\|^2$. λ is an hyper-parameter used in order to weight the regularization term.

Since classes on training set are unbalanced, we rebalance the cost of the different classes by minimizing:

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

where:

• "π_T" represents the prior that allow to generalize

• "z" is the vector that has elements -1 or 1 for x

• "n_x" represents the numbers of training samples belonging to class 1 and 0

As first steps of our analysis we consider the main application, so $\pi_T = 0.5$ is used to estimate the value of λ. Then results with $\pi_T = 0.1$ and $\pi_T = 0.9$ will be explored. After the minimization, that is performed leveraging *scipy.optimize.fmin_l_bfgs_b* function, the score of a sample x_t can be computed as:

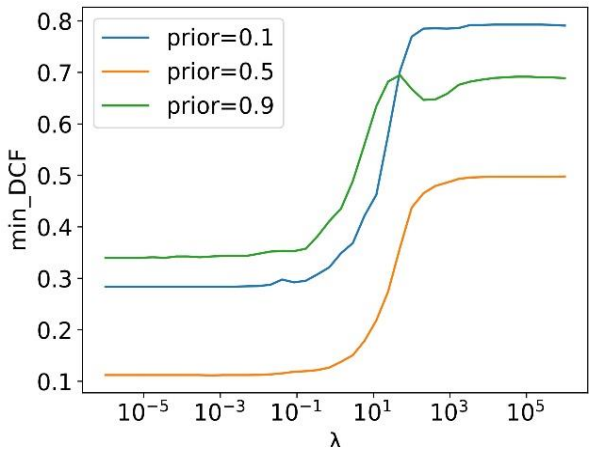
$$s(x_t) = w^t x_t + b$$

where:

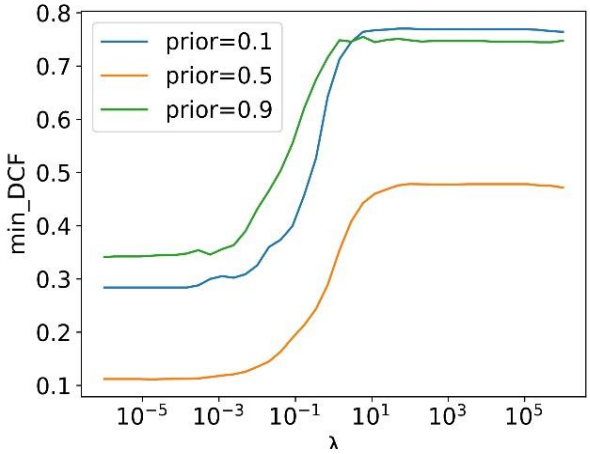
• "w^T" refers to the orthogonal vector respect to the hyperplane that we have defined

• "b" refers to the bias term

The estimation of a value of λ that give good results is made computing minDCF with different values. The interval chosen to find the best value is from 10^{-6} to 10^{6}.



Linear Logistic Regression (πT=0.5) : Min DCF with respect to λ – Raw features



Linear Logistic Regression (πT=0.5) : Min DCF with respect to λ – ZScore features

From the plots we can observe that with large value of λ the model performs worst and is not so good in correctly classify samples. This is related to the fact that the increase of the contribution of the regularization makes the model too simple and unable to behave in an efficient way. A good value for the hyper-parameter could be λ =10^{-5}, so the following results are obtaining considering this value, which can be considered as a useful trade-off between overfitting and underfitting risk.

Table 4: Min DCF results for Linear logreg with K-fold cross validation (K=5)

	Linear LR-**RAW**		
Model	**π= 0.5**	**π= 0.1**	**π= 0.9**
LR(λ=1e-5, π_T=0.5)	0.112	0.283	0.339
LR(λ=1e-5, π_T=0.1)	0.120	0.299	0.377
LR(λ=1e-5, π_T=0.9)	0.110	0.315	0.344

Table 5: Min DCF results for Linear logreg with K-fold cross validation (K=5)

	Linear LR-**ZSCORE**		
Model	**π= 0.5**	**π= 0.1**	**π= 0.9**
LR(λ=1e-5, π_T=0.5)	0.112	0.283	0.339
LR(λ=1e-5, π_T=0.1)	0.120	0.299	0.377
LR(λ=1e-5, π_T=0.9)	0.110	0.315	0.344

Table 6: Min DCF results for Linear logreg with K-fold cross validation (K=5) and PCA

	RAW- PCA(11)			Z-score -PCA(11)		
Model	**π= 0.5**	**π= 0.1**	**π= 0.9**	**π= 0.5**	**π= 0.1**	**π= 0.9**
LR(λ=1e-5, π_T=0.5)	0.120	0.295	0.365	0.120	0.295	0.365
LR(λ=1e-5, π_T=0.1)	0.127	0.305	0.376	0.127	0.305	0.376
LR(λ=1e-5, π_T=0.9)	0.117	0.313	0.360	0.117	0.313	0.360

The results obtained with features pre-processed with Z-score normalization are exactly the same of the ones on raw features. Using different values for π_T does not help significantly in the classification performance for specific application. We can however notice that slightly better results are obtained using π_T = 0.9, probably because of the class imbalance of the dataset. As we expected and we can observe in the tables above, scores after PCA pre-processing are really like the one obtained without the technique.

Quadratic Logistic Regression

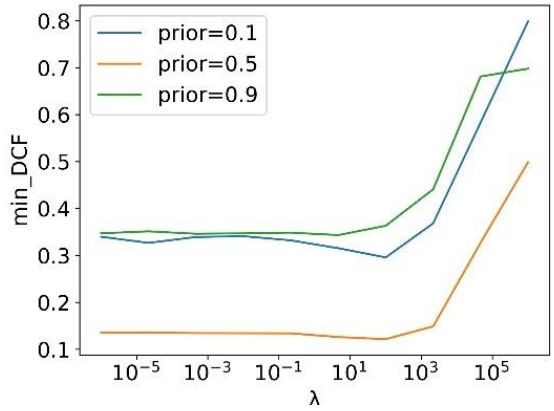
The other kind of logistic regression model that we are going to consider in our analysis is non-linear Logistic Regression model. To compute quadratic separation surfaces it is possible to define an expanded feature space $\phi(x)$, where:

$$\phi(x) = \left[\frac{vec(x\,x^T)}{x}\right]$$

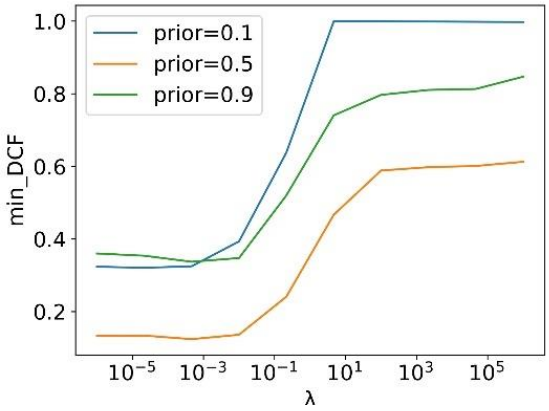
and $vec(x\,x^T)$ is the operator that stacks the columns of matrix $x\,x^T$.

Non-linear LR model works by leveraging $\phi(x)$ and not simply x**.** In this way the model has is characterized by a linear separation surface in the expanded space, but effectively computes quadratic separation boundaries within the original space.

It is expected that the quadratic logistic regression model performs worse than the linear one, because of the characteristics of data. As seen in the dataset analysis section, the scatter plots and the histograms show that classes are suitable for being separated with linear decision rules.



Quadratic Logistic Regression (πT=0.5) : Min DCF with respect to λ – Raw features



Quadratic Logistic Regression (πT=0.5) : Min DCF with respect to λ – Zscore features

Plotting the min DCF with different values of λ , we find out that the best value of lambda is $\lambda = 10^2$ for raw features and $\lambda = 10^{-3}$. Differently from the linear model, results on raw features and on normalized ones are different, because of the feature expansion and non-linear transformation to a different embedding space. Another difference from the linear model is that, for raw features, in this case the regularization term is useful and provides benefit to the quadratic classifier.

Altogether the results are worse than the linear model. PCA does not improve the performance and so results are not reported. Overall the most promising option is Linear Logistic regression with $\lambda = 10^{-5}$ and $\pi_T = 0.9$, both on raw and z-scored features and without PCA.

Table 7: Min DCF results for Quadratic logreg with K-fold cross validation (K=5)

	Quadratic LR-**RAW**		
Model	**π= 0.5**	**π= 0.1**	**π= 0.9**
LR(λ=100, πT=0.5)	0.121	0.299	0.363
LR(λ=100, πT=0.1)	0.125	0.313	0.406
LR(λ=100, πT=0.9)	0.122	0.331	0.356

Table 8: Min DCF results for Quadratic logreg with K-fold cross validation (K=5)

	Quadratic LR-**ZSCORE**		
Model	**π= 0.5**	**π= 0.1**	**π= 0.9**
LR(λ=1e-3, πT=0.5)	0.125	0.334	0.332
LR(λ=1e-3, πT=0.1)	0.131	0.353	0.360
LR(λ=1e-3, πT=0.9)	0.135	0.323	0.313

Support Vector Machines

Our analysis now shifts the focus on Support Vector Machines (SVM), which represents another instance of a supervised discriminative model for classification.

Similar to logistic regression (LR), the SVM model aims to estimate a hyperplane that effectively separates the classes. However, unlike the LR model, SVM want to identify the hyperplane that maximizes the margin, so the distance of the closest point from the separation surface.

Linear SVM

Linear SVM seek for separation hyperplane which maximizes the margin. The linear SVM objective consists in minimizing the following expression, referred as primal formulation:

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

Where $\hat{w}$ also includes the bias term b and $\hat{x}_i$ is x_i extended with 1. C is an hyperparameter.

$$\hat{\boldsymbol{x}}_i = \begin{bmatrix} \boldsymbol{x}_i \\ 1 \end{bmatrix}, \quad \hat{\boldsymbol{w}} = \begin{bmatrix} \boldsymbol{w} \\ b \end{bmatrix}$$

We take into consideration the **dual formulation,** because more easily manageable. It can be obtained using the Lagrange multiplier. SVM objective becomes the maximization of the expression:

$$\hat{J}^D(\boldsymbol{\alpha}) = -\frac{1}{2} \boldsymbol{\alpha}^T \hat{\boldsymbol{H}} \boldsymbol{\alpha} + \boldsymbol{\alpha}^T \mathbf{1}$$
$$\text{s. t. } 0 \leq \boldsymbol{\alpha}_i \leq C, \forall i \in \{1 \dots n\}$$

with

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

To maximize the dual formulation we used the function *scipy.optimize.fmin_l_bfgs_b.* This implementation compute minimizer of a function, for this reason we perform minimization of and $-\hat{J}^D(\alpha)$ not maximization of $\hat{J}^D(\alpha)$.

Once solution with respect to $\boldsymbol{\alpha}$ is computed, the primal solution can be retrieved as:

$$\hat{\boldsymbol{w}}^* = \sum_{i=1}^{n} \boldsymbol{\alpha}_i z_i \hat{\boldsymbol{x}}_i$$

Finally, the score for a sample x_t is:

$$s(\boldsymbol{x}_t) = \hat{\boldsymbol{w}}^{*T}\hat{\boldsymbol{x}}_t$$

Different values of hyperparameter C are used for two different classes. This because we want to make the classes balanced:

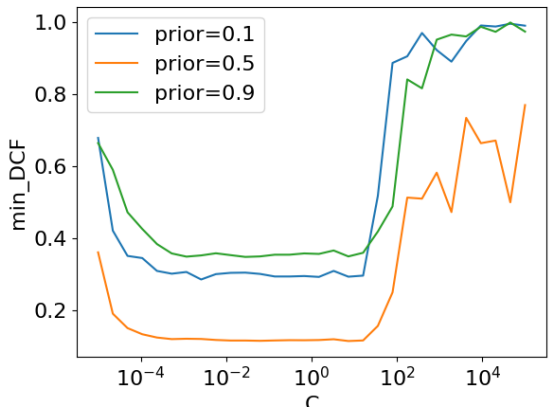
- $C_i = C_T = C\frac{\pi_T}{\pi_T{}^{emp}}$ is relative to sample of class 1
- $C_i = C_F = C\frac{\pi_F}{\pi_F{}^{emp}}$ is relative to sample of class 0

With $\pi_T{}^{emp}$ and $\pi_F{}^{emp}$ representing the empirical prior evaluated over the training dataset.

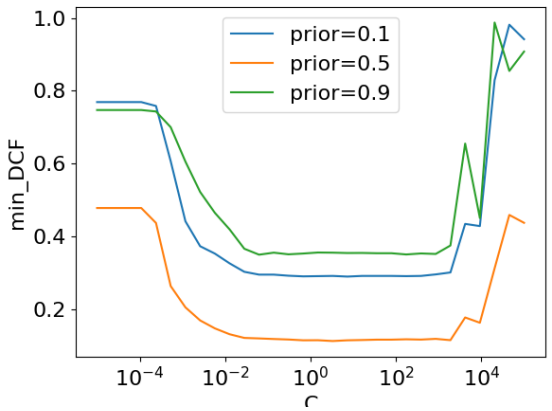
In the analysis we used linear SVM with $\pi_T = 0.5, \pi_T = 0.1, \pi_T = 0.9$, together with linear SVM without re-balancing. π_F is computed as $\pi_F = 1 - \pi_T$.

We need to tune the hyperparameter C. In general C represents a trade-off between minimizing training errors and maximizing the margin. When C approaches infinity, the model minimizes training errors but tends to overfit and generalize poorly. Once again, we resort to K-fold cross validation with 5 folds.

From the previous considerations and dataset analysis, we expect that the linear model performs better than the quadratic one. For the hyper-parameter tuning we will consider an interval between 10^{-5} and 10^{5} in a logarithmic way.



Linear SVM (πT=0.5) : Min DCF with respect to C – Raw features



Linear SVM (πT=0.5) : Min DCF with respect to C – Zscore features

In the range between 10^{-3} and 10^{1} for raw features the min DCF is quite steady both for raw and z-score features. Outside this range the min DCF has higher values. $C = 1$ is selected. In the following are reported the performances of the classifier with this hyperparameter.

Table 9: Min DCF results for Linear SVM with K-fold cross validation (K=5)

Linear	RAW		
Model	**π= 0.5**	**π= 0.1**	**π= 0.9**
SVM(C=1, π_T=0.5)	0.115	0.294	0.351
SVM(C=1, π_T=0.1)	0.128	0.316	0.377
SVM(C=1, π_T=0.9)	0.116	0.327	0.334
SVM(C=1, no rebalancing)	0.112	0.320	0.341

Table 10: Min DCF results for Linear SVM on Zscore features with K-fold (K=5)

Linear	ZSCORE		
Model	**π= 0.5**	**π= 0.1**	**π= 0.9**
SVM(C=1, π_T=0.5)	0.115	0.294	0.351
SVM(C=1, π_T=0.1)	0.128	0.316	0.377
SVM(C=1, π_T=0.9)	0.116	0.327	0.334
SVM(C=1, no rebalancing)	0.112	0.320	0.341

Table 11: Min DCF results for Linear SVM on Zscore features with K-fold (K=5)

	RAW- PCA(11)			Z-score -PCA(11)		
Model	**π= 0.5**	**π= 0.1**	**π= 0.9**	**π= 0.5**	**π= 0.1**	**π= 0.9**
SVM(C=1, π_T=0.5)	0.119	0.301	0.361	0.119	0.301	0.361
SVM(C=1, π_T=0.1)	0.126	0.317	0.384	0.126	0.317	0.384
SVM(C=1, π_T=0.9)	0.117	0.319	0.352	0.117	0.319	0.352
SVM(C=1, no rebalancing)	0.115	0.310	0.339	0.115	0.310	0.339

As for Logistic Regression and MVG models PCA does not improve the performance a lot: results on data with PCA(11) are quite similar to results on data without PCA. While outcomes on data PCA with m<11 are poorer than outcomes on data without PCA.

It can be observed that, for z-score features, Linear SVM with $\pi_T = 0.9$, Linear SVM with $\pi_T = 0.5$ and SVM without rebalancing, obtain similar results. At the same time SVM with $\pi_T = 0.1$ performs slightly worse. Rebalance does not consistently improve results.

Altogether results obtained are fine. It is possible to observe that they are really similar to the ones obtained with linear logistic regression.

The best configuration of Linear SVM($C = 1$) is with $\pi_T = 0.9$ on z-score features.

Quadratic SVM

Given that non-linear models do not obtain very good results, we expect that also non-linear SVM gives outcomes poorer than the linear one. Despite this, for completeness, this type of models will still be addressed here below.

Considering the dual formulation of SVM problem, embedding a non–linear transformation only is based on performing dot products in the expanded space. It is feasible to calculate a linear separation boundary in the expanded space, which corresponds to a non-linear separation boundary in the original feature space.

The kernel function *k* computes the dot product in the expanded space:

$$k(\boldsymbol{x}_1, \boldsymbol{x}_2) = \boldsymbol{\Phi}(\boldsymbol{x}_1)\boldsymbol{\Phi}(\boldsymbol{x}_2)$$

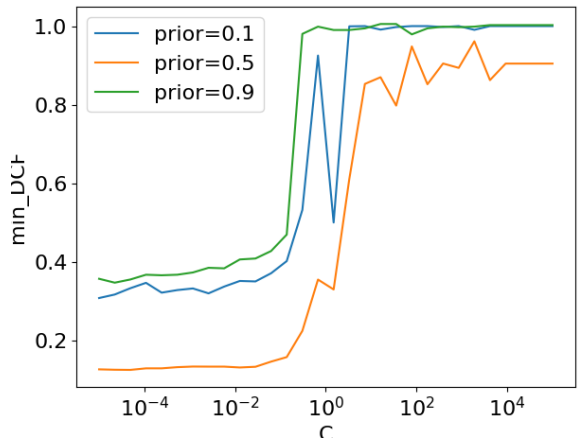
With this, the matrix H should be written as:

$$\boldsymbol{H}_{i,j} = z_i z_j \boldsymbol{\Phi}(\boldsymbol{x}_i)^T \boldsymbol{\Phi}(\boldsymbol{x}_j) = z_i z_j \, k(\boldsymbol{x}_1, \boldsymbol{x}_2)$$

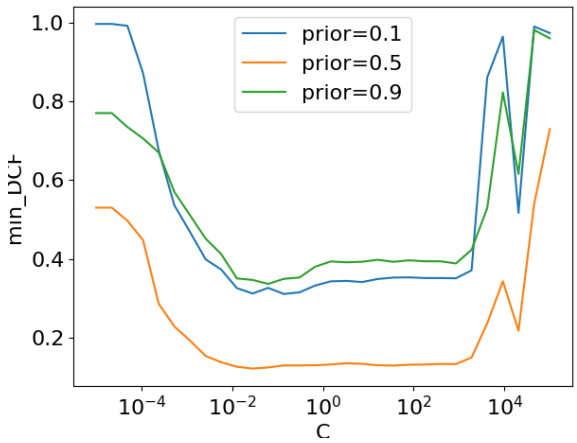
The kernel for the quadratic SVM can be represented as:

$$k(\boldsymbol{x}_1, \boldsymbol{x}_2) = (\boldsymbol{x}_1^T \boldsymbol{x}_2 + 1)^2$$

Also in this case the model depends on the hyperparameter C, so, to choose the best value, min DCF for different C is plotted.



Quadratic SVM (πT=0.5) : Min DCF with respect to C – Raw features



Quadratic SVM (πT=0.5) : Min DCF with respect to C – Zscore features

Once again, the selection of C is a crucial factor, and guided by the plots presented earlier.

These bring us to choose C = 10^{-3} in the following for the raw features, and C = 10^{-1} for z-score ones.

Table 12: Min DCF results for Quadratic SVM on Raw features with K-fold (K=5)

Quadratic	Raw		
Model	**π= 0.5**	**π= 0.1**	**π= 0.9**
QSVM(C=10^{-3}, π_T=0.5)	0.134	0.335	0.368
QSVM(C=10^{-3}, π_T=0.1)	0.150	0.404	0.455
QSVM(C=10^{-3}, π_T=0.9)	0.144	0.414	0.360
QSVM(C=10^{-3}, no rebalancing)	0.131	0.332	0.336

Table 13: Min DCF results for Quadratic SVM on Z-Score features with K-fold (K=5)

Quadratic	Z-Score		
Model	**π= 0.5**	**π= 0.1**	**π= 0.9**
QSVM(C=10^{-1}, π_T=0.5)	0.128	0.342	0.405
QSVM(C=10^{-1}, π_T=0.1)	0.154	0.412	0.468
QSVM(C=10^{-1}, π_T=0.9)	0.156	0.379	0.398
QSVM(C=10^{-1}, no rebalancing)	0.124	0.308	0.334

Observing the results above, it is possible to say that Quadratic SVM exhibits poorer performance compared to Quadratic LR. In this case z-score preprocessing provides results quite different from the ones on raw features, bringing a quite consistent improvement. Furthermore, both non-linear models, as expected, do not perform as well as their linear counterparts.

RBF SVM

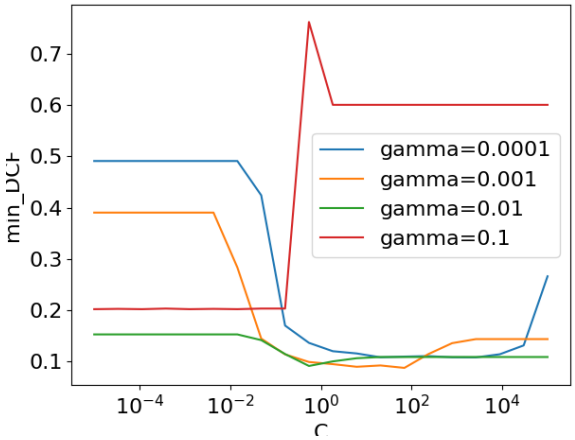
In the following we will treat another kind of SVM based on a Radial Basis Function (RBF) kernel.

This kind of kernel function is structured in this way:

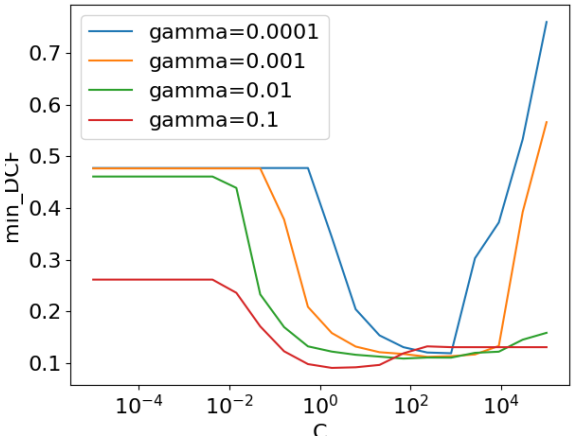
$$k(\boldsymbol{x}_1, \boldsymbol{x}_2) = e^{-\gamma \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|^2}$$

This function is used to substitute the dot product inside the H matrix in the dual problem.

RBF SVM also depends on an additional hyperparameter γ which establish the width of the kernel. So, this model requires the selection of 2 hyperparameters. To make this choice we plot min DCF with respect to C for different values of γ.



RBF SVM (πT=0.5) : Min DCF with respect to C and γ– Raw features



RBF SVM (πT=0.5) : Min DCF with respect to C and γ– Zscore features

After looked at the plot, we choose the couple of hyperparameters with $C = 10^2$ and $\gamma = 0.001$ for raw feature, while for z-score features we choose $C = 10$ and $\gamma = 0.1$.

Table 14: Min DCF results for RBF SVM on Raw features with K-fold (K=5)

RBF	Raw		
Model	**π= 0.5**	**π= 0.1**	**π= 0.9**
RBSVM(C=100, γ=0.001, πT=0.5)	0.094	0.240	0.310
RBSVM(C=100, γ=0.001, πT=0.1)	0.104	0.288	0.350
RBSVM(C=100, γ=0.001, πT=0.9)	0.103	0.327	0.326
RBSVM(C=100, γ=0.001, no rebalancing)	0.092	0.254	0.345

Table 15: Min DCF results for RBF SVM on Z-Score features with K-fold (K=5)

RBF	Z-Score		
Model	**π= 0.5**	**π= 0.1**	**π= 0.9**
RBSVM(C=10, γ=0.1, πT=0.5)	0.093	0.254	0.288
RBSVM(C=10, γ=0.1, πT=0.1)	0.112	0.270	0.353
RBSVM(C=10, γ=0.1, πT=0.9)	0.105	0.347	0.271
RBSVM(C=10, γ=0.1, no rebalancing)	0.091	0.233	0.293

Overall, the RBF kernel based SVM gives good results in terms of minDCF. Performances are better than the ones obtained with Quadratic SVM, and there is also a slight improvement with respect to Linear SVM. Results on data with PCA(11) are quite similar to results on data without PCA, so they are not reported.

It can be observed that, for z-score features, RBF SVM with $\pi_T = 0.5$ and SVM without rebalancing obtain similar results. RBF SVM with $\pi_T = 0.1$ performs slightly worse. Rebalance does not consistently improve results, but it does not even downgrade them.

RBF SVM with $C = 10, \gamma = 0.1$ and without rebalancing on z-score is considered as the best option for SVM models.

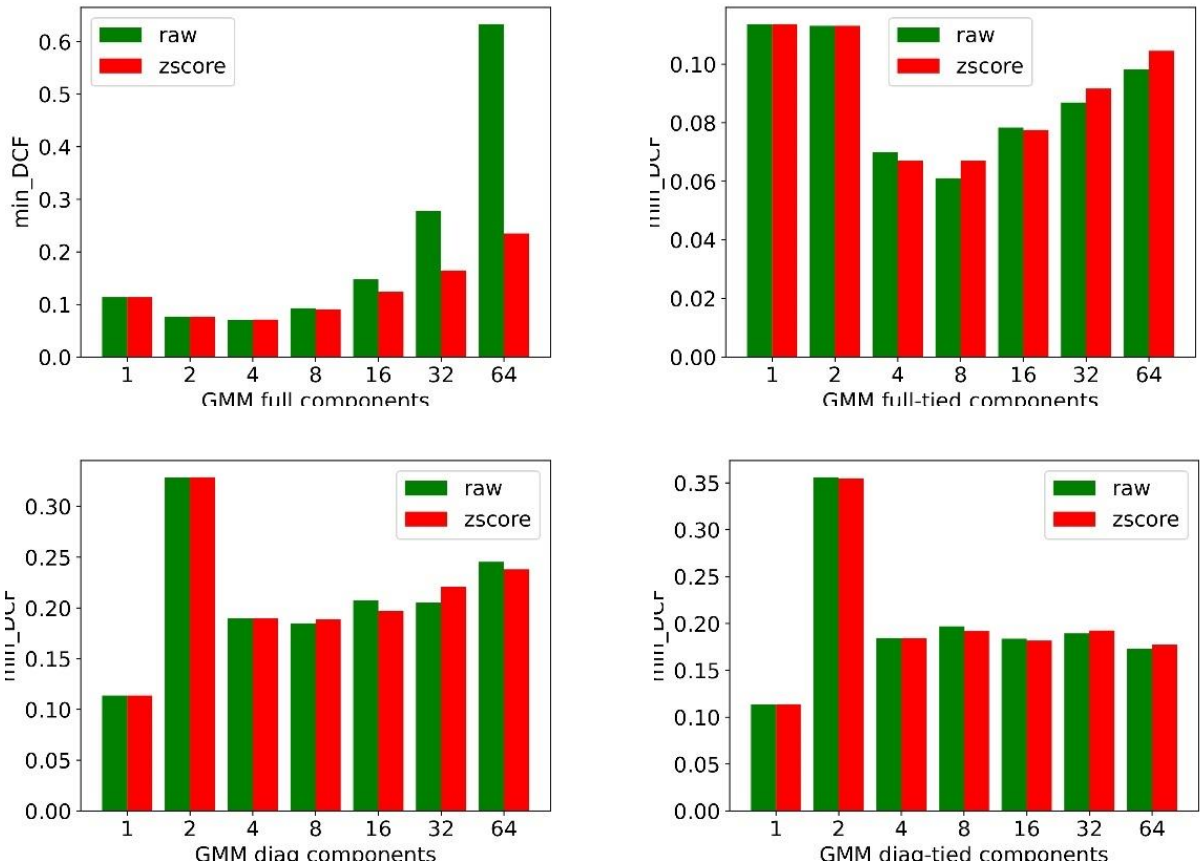
Gaussian Mixture Model

Our analysis takes into considerations as last classifier Gaussian Mixture Model (GMM). This is a generative approach which can approximate generic distributions without any supposition on distribution of data (unlike MVG). In general, these assumes that data can be distributed as gaussian with one or more components. The models that we take into consideration are Full covariance, diagonal covariance, full tied covariance and diagonal tied covariance. Once again, a tuning to select the number of Gaussians G was done utilizing the K-fold approach.

In the following we plot the min DCF with different number of components to understand which is the optimal number of components for each GMM models. The plotting is made without PCA and with $\pi_T = 0.5$. We leverage LBG algorithm to incrementally start from a GMM with 2 components to a GMM with G components. In this case the hyperparameter to tune is the number of components.

The previous analysis regarding Gaussian models and dataset analysis indicates that the models that can perform very well are GMM and Tied GMM, for the same reasons we explained earlier.

Remembering that the training set is characterized by a distribution that is similar to a gaussian with three components, we expect that best results are obtained with a GMM model with two components or with four components.



Min DCF with respect to different number of components for different option of GMM

For each GMM option we select the number of components that provides the lowest min DCF for and we proceed with a further analysis. Here are shown the results of the selected GMM's.

Table 16: Min DCF results for GMM on Rawfeatures with K-fold (K=5)

GMM	**Raw**		
Model	**π= 0.5**	**π= 0.1**	**π= 0.9**
Full-cov, 4 components	0.071	0.201	0.203
Tied Full-cov, 8 components	0.061	0.216	0.199
Diag-cov, 4 components	0.189	0.452	0.449
Tied Diag-cov, 16 components	0.183	0.504	0.470

Table 17: Min DCF results for GMM on Rawfeatures with K-fold (K=5)

GMM	Z-Score		
Model	**π= 0.5**	**π= 0.1**	**π= 0.9**
Full-cov, 4 components	0.072	0.200	0.204
Tied Full-cov, 8 components	0.067	0.229	0.208
Tied Full-cov, 4 components	0.067	0.236	0.222
Diag-cov, 4 components	0.190	0.452	0.450
Tied Diag-cov, 16 components	0.181	0.509	0.473

As in the MVG classifier, diagonal models have performances that are lower than the full ones, again because of the reasonable correlation between features, as shown in the heatmaps. We do not consider gaussian with one single component.

Altogether the GMM produces really encouraging results. The best options are GMM Full covariance with 4 components and GMM Tied full covariance with 8 components. Results relative to raw features and z-score ones are very similar. Despite the Full-Tied Covariance GMM with 8 components provide results slightly better than the one with 4 components, our knowledge on the fact that the data samples are taken from three groups of different ages brings to choose 4 components.

Best Model

Table 18: Min DCF results for best models

Model	π= 0.5)	π= 0.1	π= 0.9
Tied Full MVG(Raw/Zscore features)	0.109	0.299	0.341
Linear LR (λ=1e-5, πT=0.9)	0.110	0.315	0.344
RBSVM(C=10, γ=0.1; Z-Score features)	0.091	0.233	0.293
GMM Full Tied(4 components; Z-Score features)	0.067	0.236	0.222

From the overall results relative to the validation phase, the model that we will consider as the best for the evaluation phase is GMM Full-tied Covariance with 4 components, without PCA. During the evaluation phase we will also perform experiments with other classifiers for completeness.

Score calibration

Until this moment we use minDCF to compare the models' performances. This is the cost if the optimal threshold is acknowledged. MinDCF measures the cost to pay if optimal decisions for the evaluation set are made using the recognizer scores. However what it is paid in practice is not the minimum cost, but the actual cost. In practice, determining the optimal threshold for evaluation data is impossible, since it necessitates knowledge of the evaluation labels, which is unfeasible. If scores are well calibrated, the optimal threshold is a threshold that optimize the Bayes risk, so scores have an optimal threshold that corresponds to the theoretical one:

$$t = -log\frac{\tilde{\pi}}{1-\tilde{\pi}}$$

where $\tilde{\pi}$ is the effective prior. Score could be well calibrated and this is related to the fact that recognizer has given well calibrated scores or to the fact that has been performed re-calibration. In this case it is possible to use theoretical threshold t to compute the actual cost. Sometimes it can happen that score are not well calibrated, and in this case it is necessary to estimate, leveraging the validation set, a good threshold for the application. It is possible to check if the output scores of selected the models are well calibrated comparing the min DCF with the actual cost obtained using the theoretical threshold for each application.

Score recalibration approach transforms the scores in a way that the theoretical threshold can be used as optimal. A calibration function take as input the scores output of a classifier and gives us new calibrated scores that can be compared with the theoretical threshold to obtain optimal decisions. The calibration function $f(s)$ should be monotone because the higher non-calibrated scores should favor class 1 while lower non-calibrated scores should favor class 0. We will use a discriminative score model to make the transformation. This consists of training a new machine learning model directly on the set of scores obtained with k-fold cross-validation. The idea is to use a transformation function that apply a linear scaling to original set of scores and then sum a bias term:

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

Function f(s) should be interpreted as a calibrated log-likelihood ratio for the 2 classes:

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

If $f(s)$ is a log-likelihood ratio we can use the theoretical threshold $t = -log\frac{\tilde{\pi}}{1-\tilde{\pi}}$ to assign the labels. The class posterior probability is:

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

This log posterior ratio has a very similar form of the log posterior of logistic regression. Setting:

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

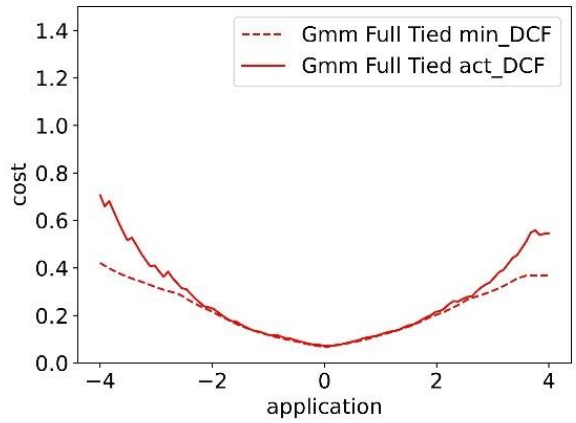
Then log posterior ratio of this new calibration model is the same as logistic regression. So, the linear transformation is estimated using the prior-weighted logistic regression model to learn the model parameters α, β'. The transformation of the score provided by this model corresponds to class posterior ratio. The log-likelihood ratio can be obtained starting from the transformation compensating with the prior. The calibrated score $f(s)$ can be retrieved as:

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

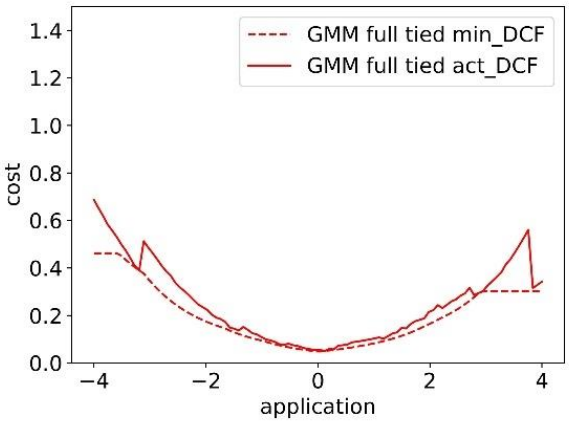
To train the model a prior $\tilde{\pi}$ must be specified, the calibration is still optimized for a specific application $\tilde{\pi}$ as for the previous approach based on optimal threshold estimation. Usually the score calibration solution with this approach works well for a wide range of applications.

We learn the model parameters α and β' on the training-scores set and then the transformation is applied on evaluation-scores samples.

In the following we compare minimum and actual cost on z-score features, relative to target application, with different prior and leveraging Bayes error plot. To estimate the calibration function's parameters, we will utilize a K-Fold Approach.



Min and Act DCF with respect to different application priors π_T for GMM



Bayes error plot for selected models with recalibrated scores π_T=0.5 for GMM

As we can see from the bayes error plot, calibration does not help with any consistent improvement, so overall we can say that this transformation is unnecessary. As demonstrated by the results below,

there is not a consistent benefit, but only the unbalanced application with $\tilde{\pi}$ = 0.9 showed a small enhancement.

Table 19: Min DCF and Act DCF for GMM full tied with 4 components

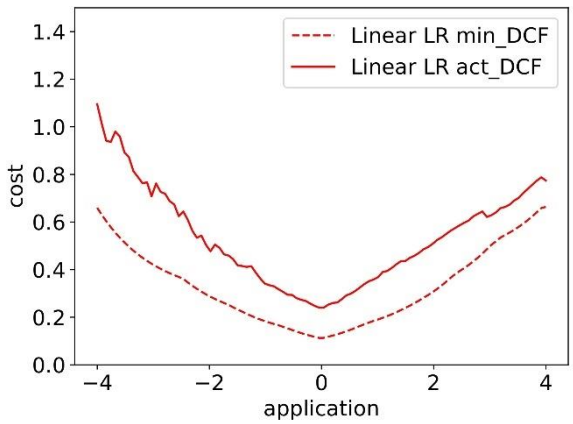
GMM	Z score feature		
	π= 0.5	**π= 0.1**	**π= 0.9**
MinDCF	0.067	0.236	0.222
ActDCF	0.071	0.249	0.244

Table 20: Min DCF and Act DCF for GMM full tied after calibration

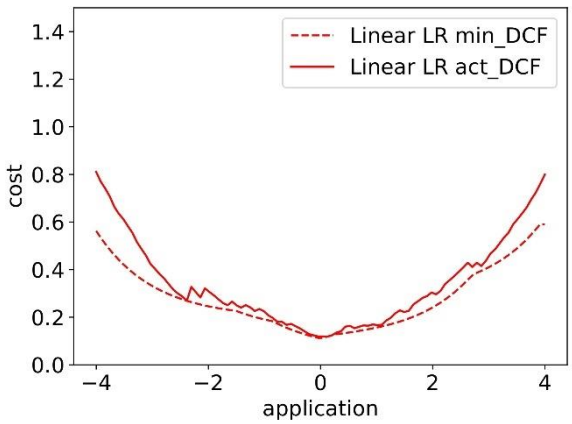
	Prior-weighted logistic regression calibration for GMM classifier		
GMM Full tied	$\tilde{\pi}$**=0.5**	$\tilde{\pi}$**= 0.1**	$\tilde{\pi}$**= 0.9**
MinDCF	0.047	0.196	0.189
ActDCF($\tilde{\pi} = 0.5$)	0.054	0.259	0.229
ActDCF($\tilde{\pi} = 0.1$)	0.054	0.262	0.229
ActDCF($\tilde{\pi} = 0.9$)	0.054	0.253	0.228
ActDCF(no calibration)	0.051	0.205	0.209

We can see that there is slight difference between results on calibrated and uncalibrated scores. Despite the improvement, we can say that GMM already provides good performance without a consistent need of calibration, so in the following we continue, for simpleness, to operate without taking into account this transformation.

In order to check its effectiveness, we also tried score calibration on other models, such as logistic regression classifier. In the part below is it possible to see the Bayes error plot of uncalibrated and calibrated versions, on Z-score features, of logistic regression:



Min and Act DCF with respect to different application priors π_T for Linear LR



Bayes error plot for selected models with recalibrated scores π_T=0.5 for GMM

Table 21: Min DCF and Act DCF for Linear LR

GMM	Z score feature		
	π= 0.5	**π= 0.1**	**π= 0.9**
MinDCF	0.110	0.315	0.344
ActDCF	0.234	0.534	0.545

It is clear that the scores in this case are not well calibrated, for this reason a calibration can easily bring to a consistent improvement in terms of results, as can be seen in the table below.

Table 22: Min DCF and Act DCF for Linear LR after calibration

	Prior-weighted logistic regression calibration for the LR classifier		
Linear LR	$\tilde{\pi}$**=0.5**	$\tilde{\pi}$**= 0.1**	$\tilde{\pi}$**= 0.9**
MinDCF	0.111	0.257	0.270
ActDCF($\tilde{\boldsymbol{\pi}} = \mathbf{0.5}$)	0.117	0.296	0.330
ActDCF($\tilde{\boldsymbol{\pi}} = \mathbf{0.1}$)	0.111	0.311	0.363
ActDCF($\tilde{\boldsymbol{\pi}} = \mathbf{0.9}$)	0.117	0.268	0.336
ActDCF(no calibration)	0.220	0.546	0.558

Fusion

As said before, the model that we are considering as the best (GMM full-tied with 4 components) does not require score calibration. Furthermore, the analysis in the following will proceed taking into consideration only z-score features.

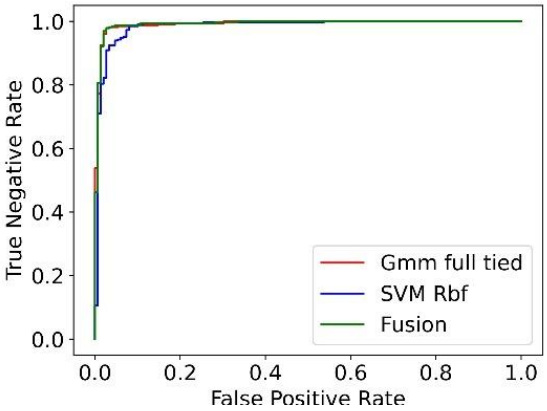
In this section we leverage the idea of combining different classifier with a fusion approach. In general classifiers, because of their different assumptions, will provide different results depending on the type. They could agree on some decisions while disagree on others. The overall idea is to combine the decisions of both in order to result in better predictions labels. In practice, two classifiers can be combined in simple voting scheme approach: each classifiers assign a label and at the end the label assigned more often is selected. The simple voting approach has some issues, if one classifier is almost certain about class 1 and two other classifiers are only slightly in favor of class 0 it is not granted that assigning label 0 is a good choice. So, rather than fusing classifiers at decision level, is better to perform a score-level fusion voting. The idea is to introduce a fused score which is a function of the scores of different classifiers. Considering a sample x_t, if $s_{t,a}$ is the score provided by classifier A, while $s_{t,b}$ is the score provided by classifier B, the fused score for sample x_t will be a fusion of this, based on a function like this:

$$f(s_{t,a}, s_{t,b}, \dots) = \alpha_A s_{t,A} + \alpha_B s_{t,B} + \cdots + \beta$$

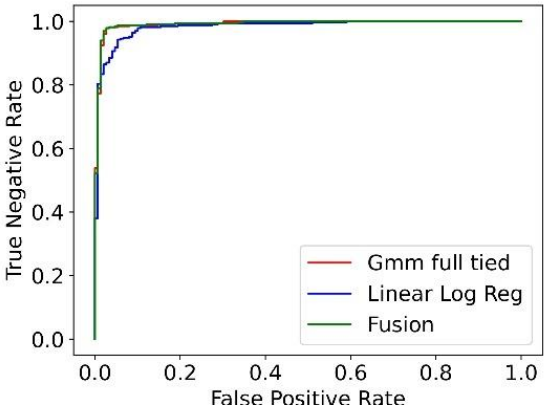
where α, β are parameters to be estimated.

The scores of different classifiers are treated as a feature vector. A prior-weighted logistic regression is used to train the model parameters similar to what has been done for score calibration.

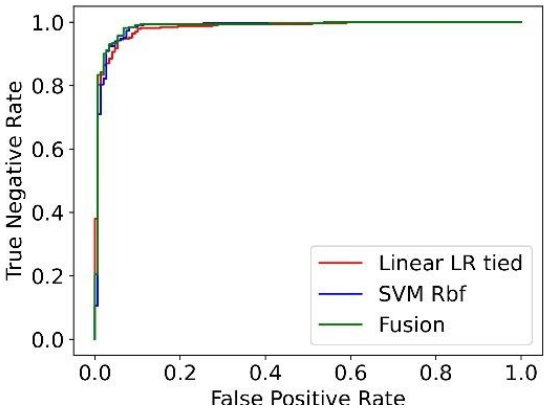
We perform fusion considering three different models: GMM full tied with 4 components, Linear logistic regression with $\lambda = 10^{-5}$ and $\pi_T = 0.9$, and SVM RBF with $\gamma = 0.1, C = 10$ and without rebalancing.



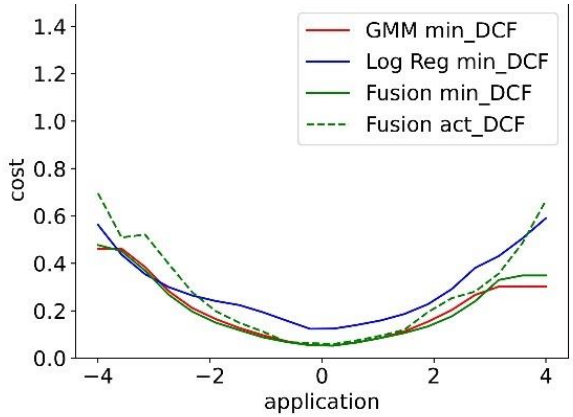
Roc plot for GMM and SVM fusion



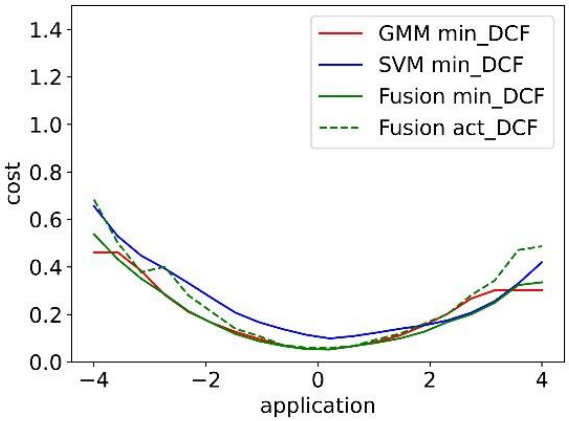
Roc plot for GMM and LogReg fusion



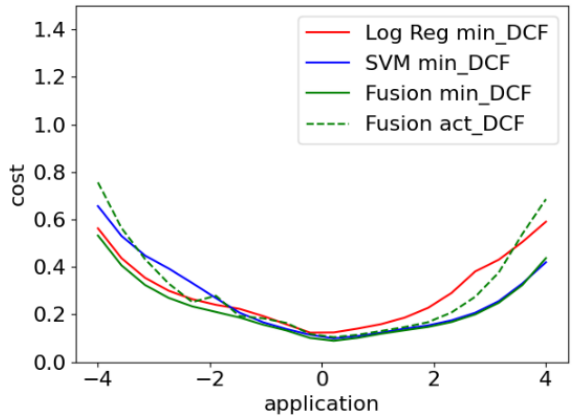
Roc plot for LogReg and SVM fusion



Bayes error plot for GMM and LogReg fusion



Bayes error plot for GMM and SVM fusion



Bayes error plot for LogReg and SVM fusion

From the ROC plots it is possible to observe that, overall, the fusion model provides slightly better results respect to the single models. However, the ROC curves relative to GMM are similar and almost overlapped to the fusion ones. In terms of calibration, we can see from the Bayes error plot that the fused model does not require calibration.

In the following are reported the results of the fusion models compared with the results of single models.

Table 23: Min DCF results for Fusion models on Z-Score features with K-fold (K=5)

	Z-Score		
Model	**π= 0.5**	**π= 0.1**	**π= 0.9**
GMM + Linear LR	0.048	0.181	0.162
GMM + RBF SVM	0.048	0.199	0.155
Linear LR + RBF SVM	0.085	0.226	0.161
GMM	0.067	0.236	0.222

We can observe that GMM+LR and GMM+SVM gives similar results, providing a consistent improvement to the single SVM and LR, and a small one respect to GMM. Also the model built upon LR+SVM gives results better than the single one. Altogether, again for simplicity, the model considered as the best to be tested in the evaluation phase is the GMM Tied full covariance trained on z-score normalized features.

EXPERIMENTAL RESULTS ON EVALUATION SET

In this section we analyze the choices that were made, checking the quality of our models trained on the whole training set directly on the evaluation set. Again, in this phase we will once more calculate on unseen data the metrics for the classifiers, with all the hyperparameter combinations previously considered. This step is crucial to verify if the promising options identified during the validation set analysis continue to yield good results when applied to the evaluation set. When we refer to z-score normalized evaluation set, we have applied Z-normalization directly using mean and standard deviation of training set.

Multivariate Gaussian Classifier (MVG)

Once again we use as a measure of performance the minimum DCF, to verify if the proposed solution is the one that can achieve the best accuracy.

Table 24: Min DCF for MVG on the evaluation set

	RAW			Z-score		
Model	**π= 0.5**	**π= 0.1**	**π= 0.9**	**π= 0.5**	**π= 0.1**	**π= 0.9**
Full-cov	0.119	0.312	0.312	0.119	0.312	0.312
Diag-cov	0.434	0.817	0.705	0.434	0.817	0.705
Tied full-cov	0.116	0.301	0.308	0.116	0.301	0.308
Tied diag-cov	0.435	0.815	0.711	0.435	0.815	0.711

Table 25: Min DCF for MVG with PCA(m=11) on the evaluation set

	RAW- PCA(11)			Z-score -PCA(11)		
Model	**π= 0.5**	**π= 0.1**	**π= 0.9**	**π= 0.5**	**π= 0.1**	**π= 0.9**
Full-cov	0.123	0.321	0.332	0.126	0.324	0.331
Diag-cov	0.130	0.355	0.323	0.122	0.323	0.339
Tied full-cov	0.121	0.314	0.322	0.123	0.313	0.312
Tied diag-cov	0.124	0.326	0.320	0.120	0.309	0.338

All the observations regarding the validation set results remain valid. For the primary application, Tied Full Covariance emerges as the optimal choice, and it's worth noting that PCA do not bring to significant enhancements inside the results. As expected results with PCA (m<11) are worst and for this reason we do not report them.

Logistic Regression

Linear Logistic Regression

In the following the focus turns on linear regression and we can clearly see from the table below that, again, results are overall consistent, despite in this case the best results is obtained with π_T=0.5 and not π_T=0.9.

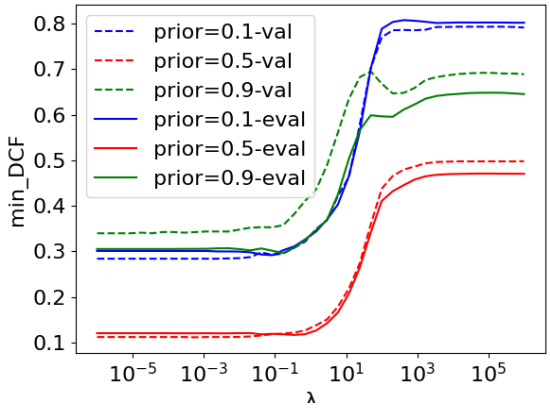
Table 26: Min DCF for Linear LR on the evaluation set

	Linear LR-**RAW**		
Model	**π= 0.5**	**π= 0.1**	**π= 0.9**
LR(λ=1e-5, π_T=0.5)	0.120	0.300	0.305
LR(λ=1e-5, π_T=0.1)	0.122	0.296	0.339
LR(λ=1e-5, π_T=0.9)	0.123	0.340	0.279

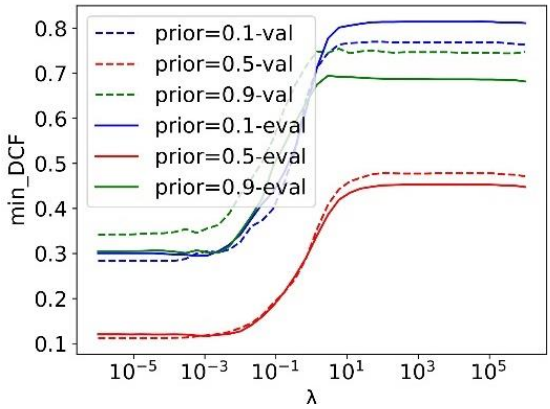
Table 27: Min DCF for Linear LR on the evaluation set

	Linear LR-**Zscore**		
Model	**π= 0.5**	**π= 0.1**	**π= 0.9**
LR(λ=1e-5, π_T=0.5)	0.120	0.300	0.305
LR(λ=1e-5, π_T=0.1)	0.122	0.296	0.339
LR(λ=1e-5, π_T=0.9)	0.123	0.340	0.279

It is also possible to verify if the choice of the optimal value for hyperparameter λ is still valid for evaluation set.



Linear Logistic Regression (πT=0.5) : Min DCF with respect toλ – Raw features - Evaluation



Linear Logistic Regression (πT=0.5) : Min DCF with respect toλ – Zscore features - Evaluation

Quadratic Logistic Regression

The same analysis is done with Quadratic Logistic Regression, and the results reported in the following show congruous performances. Once again it confirms that Quadratic Logistic Regression performs worse than the linear counterparts, as expected. Again, as for the linear one, Quadratic Logistic Regression trained with $\pi_T = 0.9$ is no more the option that reaach the best result, in favor of the $\pi_T = 0.5$ option. We can also underline that in this case the classifier performs better on z-score features respect to raw one.

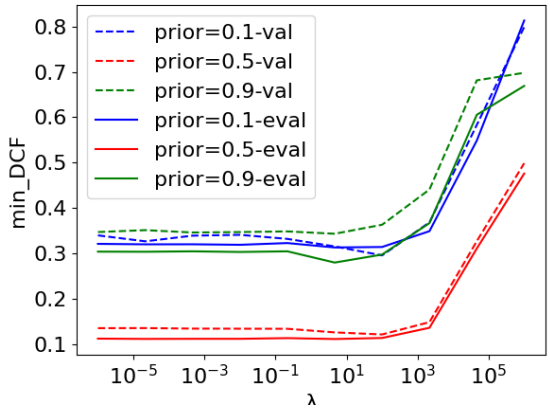
Table 28: Min DCF for Quadratic LR on the evaluation set

	Quadratic LR-**RAW**		
Model	**π= 0.5**	**π= 0.1**	**π= 0.9**
QLR(λ=100, π_T=0.5)	0.114	0.314	0.297
QLR(λ=100, π_T=0.1)	0.116	0.288	0.340
QLR(λ=100, π_T=0.9)	0.129	0.383	0.279

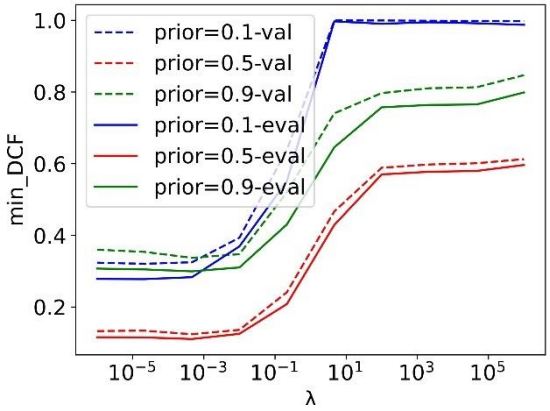
Table 29: Min DCF for Quadratic LR on the evaluation set

	Quadratic LR-**ZSCORE**		
Model	**π= 0.5**	**π= 0.1**	**π= 0.9**
QLR(λ=1e-3, π_T=0.5)	0.113	0.297	0.294
QLR(λ=1e-3, π_T=0.1)	0.122	0.324	0.303
QLR(λ=1e-3, π_T=0.9)	0.120	0.323	0.286

Once more, from the minDCF plots below, it is possible can see that the choice of λ was altogether right for our application.



Quadratic Logistic Regression (πT=0.5) : Min DCF with respect toλ – Raw features - Evaluation



Quadratic Logistic Regression (πT=0.5) : Min DCF with respect toλ – Zscorefeatures - Evaluation

SVM

Linear SVM

In this part we check the results of linear SVM, that are shown in the tables below. We consider for SVM both raw data and data with Z-Score transformation.

Table 30: Min DCF results for Linear SVM on the evaluation set

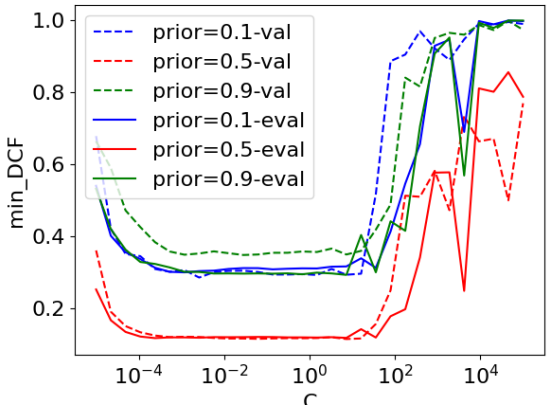
Linear	**RAW**		
Model	**π= 0.5**	**π= 0.1**	**π= 0.9**
SVM(C=1, π_T=0.5)	0.117	0.309	0.295
SVM(C=1, π_T=0.1)	0.124	0.298	0.355
SVM(C=1, π_T=0.9)	0.132	0.360	0.285
SVM(C=1, no rebalancing)	0.123	0.327	0.281

Table 31: Min DCF results for Linear SVM on Zscore features on the evaluation set

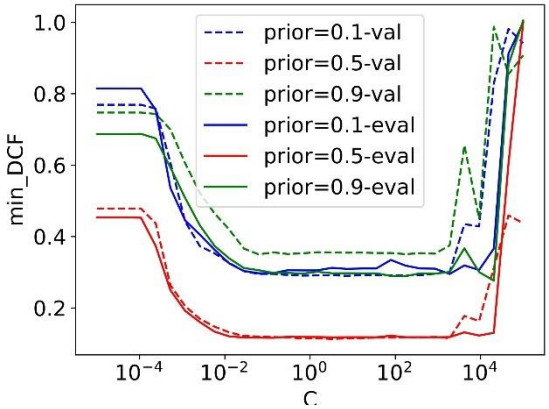
Linear	ZSCORE		
Model	**π= 0.5**	**π= 0.1**	**π= 0.9**
SVM(C=1, π_T=0.5)	0.117	0.309	0.295
SVM(C=1, π_T=0.1)	0.124	0.298	0.355
SVM(C=1, π_T=0.9)	0.132	0.360	0.285
SVM(C=1, no rebalancing)	0.123	0.327	0.281

Also in this case, the overall results are aligned with our expectations.

To verify if our choice of C was good, we repeat the tuning on the unbalanced model. In the figures below we can see that for our main application, both for raw and z-score features, the value of the hyper-parameter C=1 is a good choice.



Linear SVM (πT=0.5) : Min DCF with respect to C – Raw features - Evaluation



Linear SVM (πT=0.5) : Min DCF with respect to C – Zscore features - Evaluation

Quadratic SVM

The same analysis on linear SVM is done with Quadratic SVM, and the results shown in the following give congruous performances. Once again we have that Quadratic SVM performs, as expected, worse than the linear counterparts.

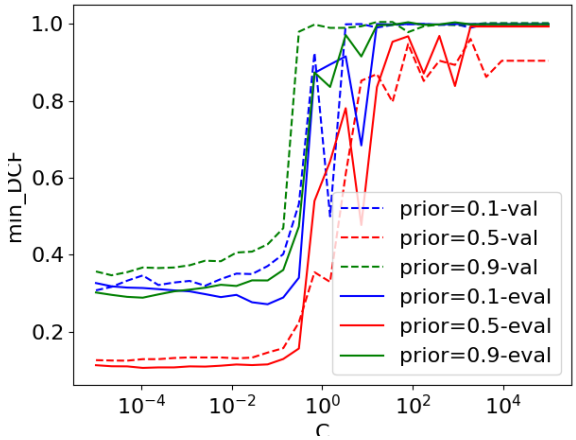
Table 32: Min DCF results for Quadratic SVM on Raw features on the evaluation set

Quadratic	Raw		
Model	**π= 0.5**	**π= 0.1**	**π= 0.9**
QSVM(C=10^{-3}, π_T=0.5)	0.110	0.305	0.311
QSVM(C=10^{-3}, π_T=0.1)	0.130	0.329	0.407
QSVM(C=10^{-3}, π_T=0.9)	0.123	0.344	0.276
QSVM(C=10^{-3}, no rebalancing)	0.111	0.299	0.277

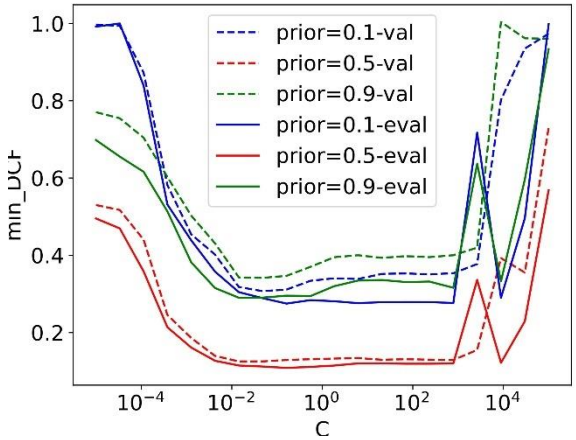
Table 33: Min DCF results for Quadratic SVM on Z-Score features on the evaluation set

Quadratic	Z-Score		
Model	**π= 0.5**	**π= 0.1**	**π= 0.9**
QSVM(C=10^{-1}, π_T=0.5)	0.113	0.274	0.290
QSVM(C=10^{-1}, π_T=0.1)	0.151	0.344	0.425
QSVM(C=10^{-1}, π_T=0.9)	0.125	0.344	0.277
QSVM(C=10^{-1}, no rebalancing)	0.114	0.288	0.268

The plots of min DCF with respect to C confirm the hypothesis made on evaluation set. $C = 10^{-3}$ for raw features and $C = 10^{-1}$ for z-score ones are confirmed to be a good choice.



Quadratic SVM (πT=0.5) : Min DCF with respect to C – Raw features - Evaluation



Quadratic SVM (πT=0.5) : Min DCF with respect to C – Zscore features - Evaluation

RBF SVM

To conclude the evaluation of SVM models, the RBF kernel one is considered. We can remind that in the validation phase this option is the one that return better performances among SVMs. Again, the experiments on unseen data show that RBF SVM version outperforms the other SVM model, obtaining good results. Rebalancing is not necessarily required, the performance without it is even better.

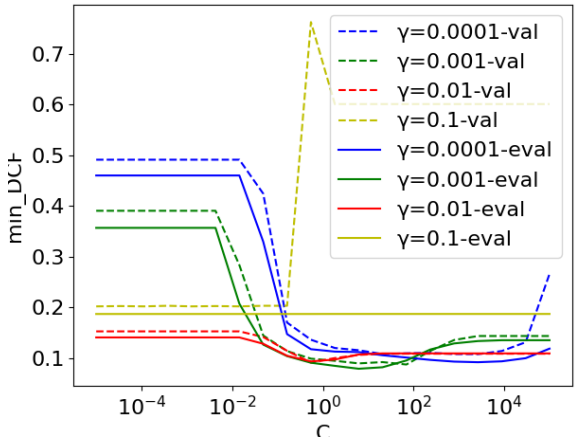
Table 34: Min DCF results for RBF SVM on Raw features on the evaluation set

RBF	Raw		
Model	**π= 0.5**	**π= 0.1**	**π= 0.9**
RBSVM(C=100, γ=0.001, π_T=0.5)	0.101	0.258	0.262
RBSVM(C=100, γ=0.001, π_T=0.1)	0.104	0.303	0.267
RBSVM(C=100, γ=0.001, π_T=0.9)	0.102	0.324	0.213
RBSVM(C=100, γ=0.001, no rebalancing)	0.093	0.270	0.249

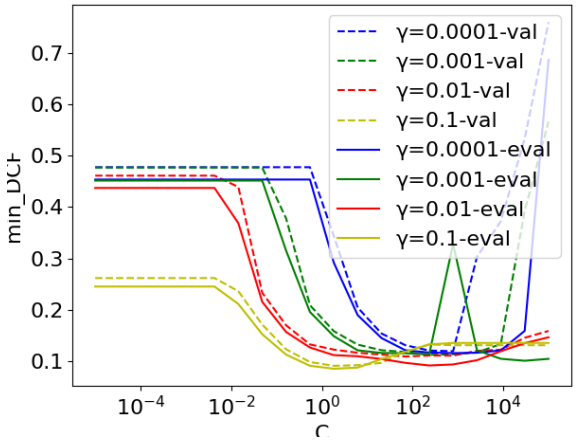
Table 35: Min DCF results for RBF SVM on Z-Score features on the evaluation set

RBF	Z-Score		
Model	**π= 0.5**	**π= 0.1**	**π= 0.9**
RBSVM(C=10, γ=0.1, π_T=0.5)	0.094	0.268	0.254
RBSVM(C=10, γ=0.1, π_T=0.1)	0.102	0.256	0.306
RBSVM(C=10, γ=0.1, π_T=0.9)	0.098	0.343	0.213
RBSVM(C=10, γ=0.1, no rebalancing)	0.089	0.275	0.234

Once more, from the minDCF plots below, it is possible to see that the choices of γ and C were altogether right for our application. Only the raw features value of C is in a position that is not the best but altogether still good.



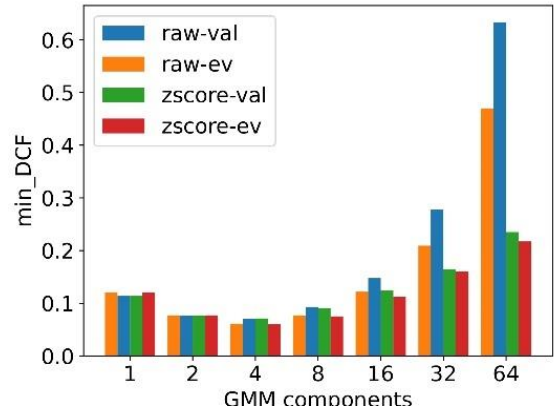
RBF SVM (πT=0.5) : Min DCF with respect to C and γ – Raw features - Evaluation



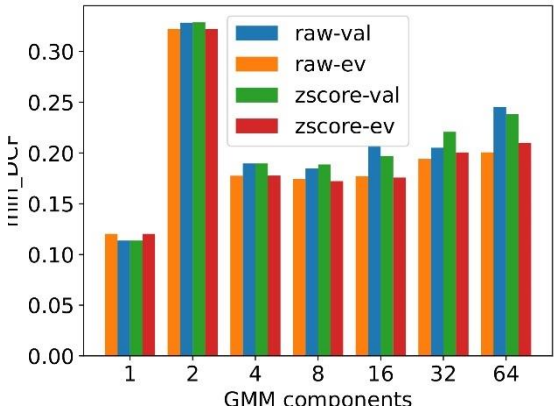
RBF SVM (πT=0.5) : Min DCF with respect to C and γ – Zscore features - Evaluation

Gaussian Mixture Model (GMM)

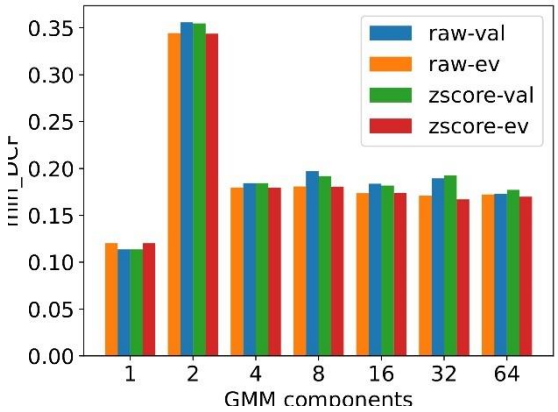
During the validation phase GMM provided very good performances, now we can try to see if results are still consistent. From tuning G we can see, in figures below, that both for raw and z-score features, the values chosen were a good choice. GMM model family confirms to behave as the best kind of model also in the evaluation set, fully supporting choices and analysis made during the validation phase.



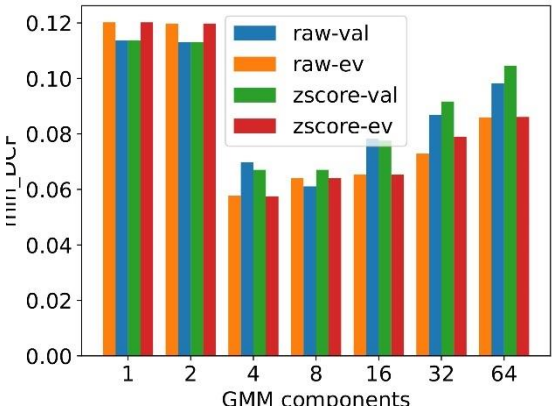
GMM Full covariance– min DCF with respect to #components - Evaluation



GMM Diag covariance– min DCF with respect to #components - Evaluation



GMM Diag Tied covariance– min DCF with respect to #components - Evaluation



GMM Full Tied covariance– min DCF with respect to #components - Evaluation

In the following tables it is possible to find the encouraging results of GMM models on the evaluation set.

Table 36: Min DCF results for GMM on Raw features on the evaluation set

GMM	Raw		
Model	**π= 0.5**	**π= 0.1**	**π= 0.9**
Full-cov, 4 components	0.061	0.155	0.155
Tied Full-cov, 8 components	0.058	0.178	0.159
Diag-cov, 4 components	0.177	0.453	0.419
Tied Diag-cov, 16 components	0.173	0.461	0.397

Table 37: Min DCF results for GMM on Z-Score features on the evaluation set

GMM	Z-Score		
Model	**π= 0.5**	**π= 0.1**	**π= 0.9**
Full-cov, 4 components	0.062	0.146	0.144
Tied Full-cov, 4 components	0.059	0.168	0.170
Diag-cov, 4 components	0.100	0.286	0.257
Tied Diag-cov, 16 components	0.114	0.310	0.290

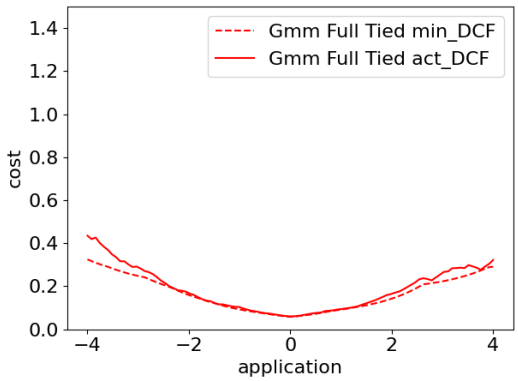
Recap

We can conclude that in general the results on evaluation set confirm what we observed on validation set: all the models considered obtained similar results in terms of minDCF. GMM Full-tied

without PCA are confirmed to be a good choice as candidate model. We consider these model on z-score features for further analysis on evaluation set.

Calibration

Up to here we have considered results on evaluation set only in term of minDCF. It is necessary to check whether the scores of the selected models on evaluation set are calibrated or not. The analysis is made on z-score preprocessed features. As in the validation phase, we will perform the check also on linear logistic regression. We can check for GMM with the following Bayes error plot:



Bayes error plot for GMM model - Evaluation

The trends in the plot align with those observed in the validation set. Recalibration of scores is not necessary. Previously, we leveraged score transformation method. Also now we will us this approach on the evaluation set to confirm the validity of our findings from the validation set.

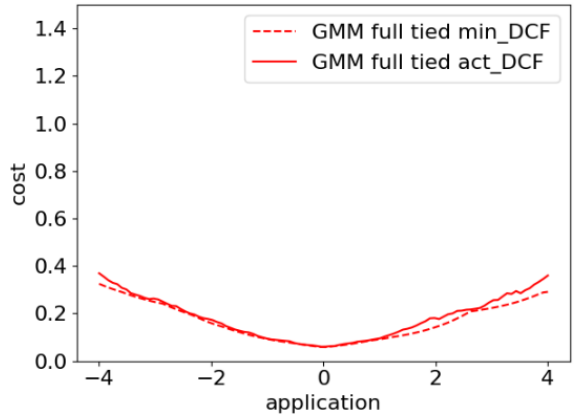
Table 38: Min DCF and Act DCF for GMM full tied with 4 components on the evaluation set

GMM	**Z score feature**		
	$\tilde{\pi}$**= 0.5**	$\tilde{\pi}$**= 0.1**	$\tilde{\pi}$**= 0.9**
MinDCF	0.059	0.168	0.170
ActDCF	0.059	0.182	0.181

Table 39: Min DCF and Act DCF for GMM after calibration and on the evaluation set

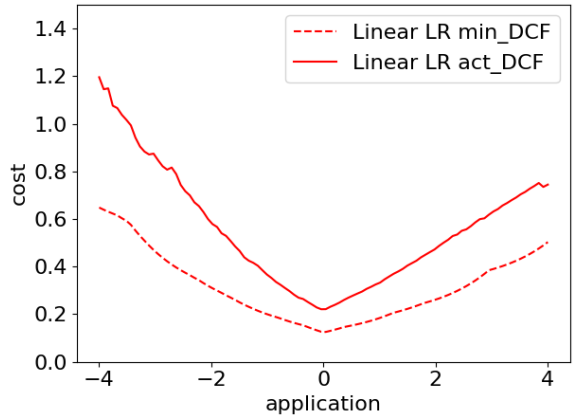
	Prior-weighted logistic regression calibration for GMM classifier		
GMM Full tied	$\tilde{\pi}$**=0.5**	$\tilde{\pi}$**= 0.1**	$\tilde{\pi}$**= 0.9**
MinDCF	0.057	0.177	0.160
ActDCF($\tilde{\pi} = \mathbf{0.5}$)	0.058	0.184	0.194
ActDCF($\tilde{\pi} = \mathbf{0.1}$)	0.058	0.188	0.198
ActDCF($\tilde{\pi} = \mathbf{0.9}$)	0.059	0.187	0.192
ActDCF(no calibration)	0.058	0.182	0.181

Results are good, as expected re-calibration does not bring a consistent improvement, overall the actual DCF is very close to minDCF.
We plot bayes error plot after recalibration (considering score calibration with $\pi = 0.5$) to verify whether evaluation scores are calibrated over a wide range of applications.



Bayes error plot after calibration for GMM model - Evaluation

As made in the validation, we now will evaluate score calibration on evaluation set for Linear logistic regression. In the part below is it possible to see the Bayes error plot of uncalibrated and calibrated versions, on Z-score features. Even in this case we can observe that Linear logistic regression requires score calibration. In the following there are the plots and results before and after re-calibration of the scores, finding even here a consistent improvement.



Bayes error plot for LR model - Evaluation

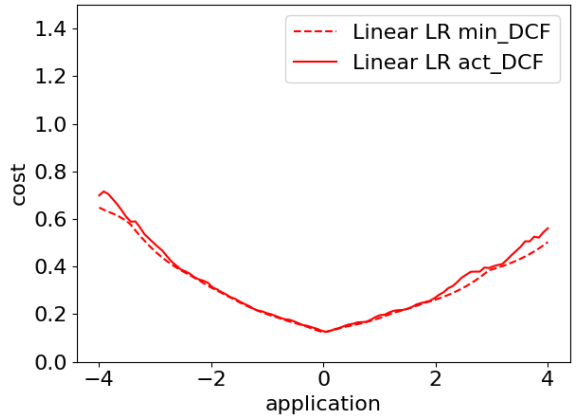
Table 40: Min DCF and Act DCF for Linear LR on the evaluation set

Linear LR	**Z score feature**		
	$\tilde{\pi}$**= 0.5**	$\tilde{\pi}$**= 0.1**	$\tilde{\pi}$**= 0.9**
MinDCF	0.123	0.340	0.279
ActDCF	0.215	0.648	0.510

Table 41: Min DCF and Act DCF for Linear LR after calibration and on the evaluation set

	Prior-weighted logistic regression calibration for Linear LR classifier		
Linear LR	$\tilde{\pi}$**=0.5**	$\tilde{\pi}$**= 0.1**	$\tilde{\pi}$**= 0.9**
MinDCF	0.123	0.339	0.278
ActDCF($\tilde{\boldsymbol{\pi}} = \mathbf{0.5}$)	0.126	0.343	0.330
ActDCF($\tilde{\boldsymbol{\pi}} = \mathbf{0.1}$)	0.124	0.342	0.363
ActDCF($\tilde{\boldsymbol{\pi}} = \mathbf{0.9}$)	0.129	0.344	0.337
ActDCF(no calibration)	0.215	0.648	0.514

After calibration:



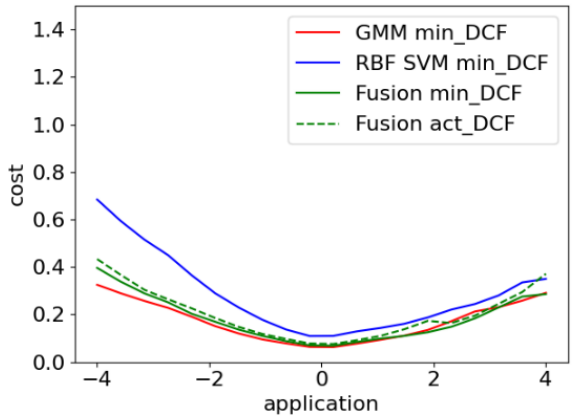
Bayes error plot after calibration for LR model - Evaluation

Fused System

We now shift our focus to the fused models introduced earlier. The goal is to check their performance on the evaluation dataset. The models that we will combine in our experiments are the ones that in the validation phase return best results: GMM Tied full covariance with 4 components together with the SVM RBF with $\gamma = 0.1, C = 10$ and without rebalancing. For simplicity, we do not report only the best combination in the validation phase. The fused system is tested with Z-Score pre-processed features. In the table below it is possible to observe the results.

Table 42: Min DCF results for Fusion models on Z-Score features on the evaluation set

	Z-Score		
Model	**π= 0.5**	**π= 0.1**	**π= 0.9**
GMM + RBF SVM	0.063	0.193	0.141
GMM	0.059	0.168	0.170



Bayes error plot for fusion model - Evaluation

Table 43: Min DCF and Act DCF for GMM full tied with 4 components on the evaluation set

GMM+RBF SVM	**Z score feature**		
	$\tilde{\pi}$**= 0.5**	$\tilde{\pi}$**= 0.1**	$\tilde{\pi}$**= 0.9**
MinDCF	0.063	0.193	0.141
ActDCF	0.069	0.225	1.175

Evaluation scores produces by fusion approach are well-calibrated over a wide range of applications, as observable from the Bayes error plot. The obtained performances are good but are not better than the ones obtained with single GMM Tied-full covariance classifier, that we will consider overall our best model.

Conclusions

In this analysis we studied different models to classify the provided dataset. On validation set we observed that all the models, except for the non-linear ones, obtained similar good results on the main application ($\tilde{\pi}$= 0.5). By the way Gaussian Mixture Model with Tied full covariance matrix outperform the other models in a consistent way. Almost all our experiments were conducted both on raw and z-score features and it is possible to observe that the preprocessing does not provide a real substantial enhancement of performances. In most of the cases we tested PCA did not help.

Evaluation set is characterized by a strong likeness to the training set. Most of the trends and behaviours of the models obtained on the validation set have been confirmed by the evaluation set analysis, underlining the similarity between the two distributions. The decisions and techniques employed during the training phase have proven to be effective when applied to the unseen data.

In the end we considered as best option a GMM Full-tied covariance model with 4 components trained over z-score pre-processed features and without PCA. This model obtains an excellent act DCF ~ 0.06 for the main application ($\tilde{\pi}$= 0.5). Results are also remarkable for the application with $\tilde{\pi} = 0.1$ with an act DCF ~ 0.18, and for the one with $\tilde{\pi} = 0.9$, with an act DCF ~ 0.181.