Gender Identification

Giuseppe Pellegrino 303999

July 17, 2023

Contents

1 Introduction

The objective of this project is to create a classifier that can determine the gender of individuals based on high-level features extracted from facial images. The dataset for this project contains image embeddings, which are condensed representations of images achieved by mapping face images onto a shared, low-dimensional space (usually with a few hundred dimensions) using neural networks or similar techniques. To ensure the model remains manageable, the dataset primarily consists of synthetic data, and the embeddings have lower dimensions compared to real-world scenarios.
Each row corresponds to a different person and contains 12 features followed by the gender label (1 for female, 0 for male). The datasets are imbalanced, with the training set having significantly more female samples, whereas the test set has significantly more male samples. In particular the training set consists of 720 samples for male class and 1680 for female class, whereas the test set contains 4200 samples belonging to male class and 1800 belonging to female class.

1.1 features analysis

Below are histograms displaying the distribution of each of the 12 features. From the histograms, we can observe that the raw features exhibit an approximate Gaussian distribution.

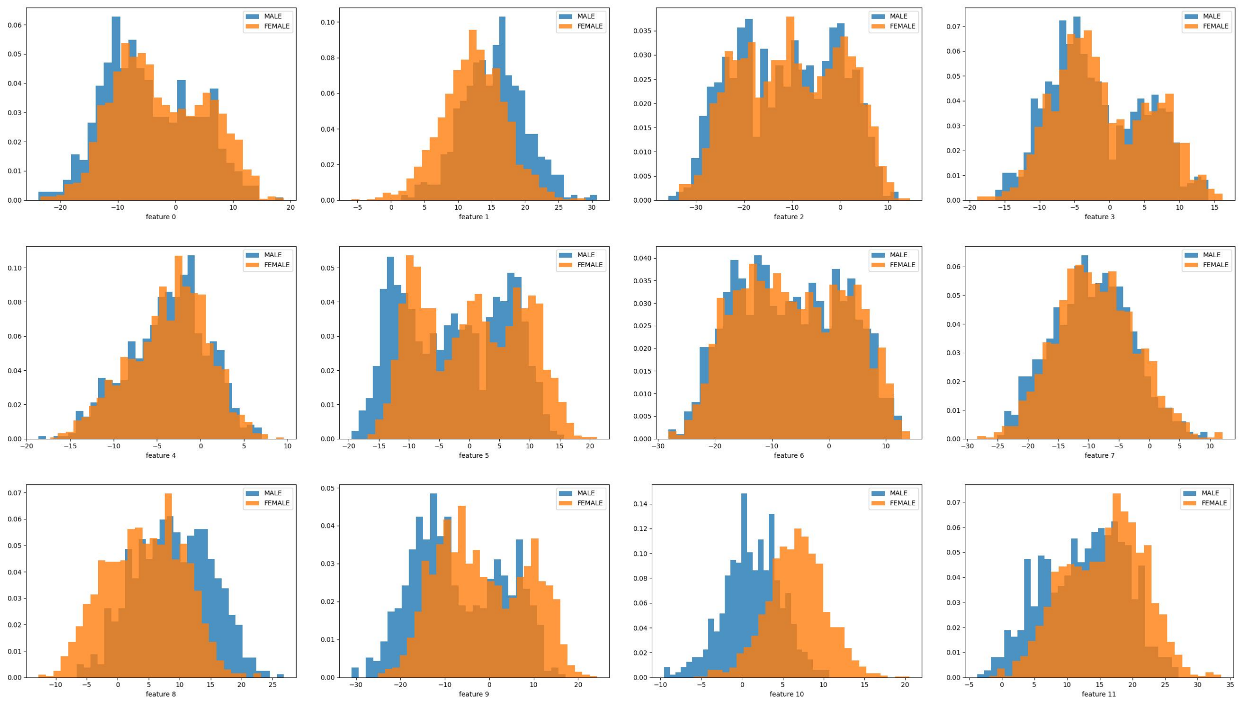


Figure 1: Raw Features

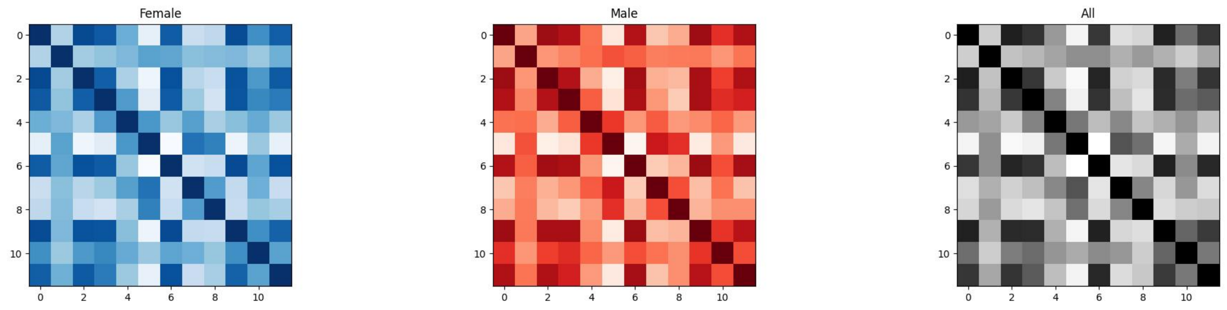


Figure 2: Pearson Correlation

Feature 6 seems to be highly correlated to the 11, 9, 2 and 0 ones. This suggests we may benefit from using PCA to map data to less correlated features, but we will check later this assumption. However, here, you will find the distributions of the features after they have undergone preprocessing, specifically Gaussianization of the features.

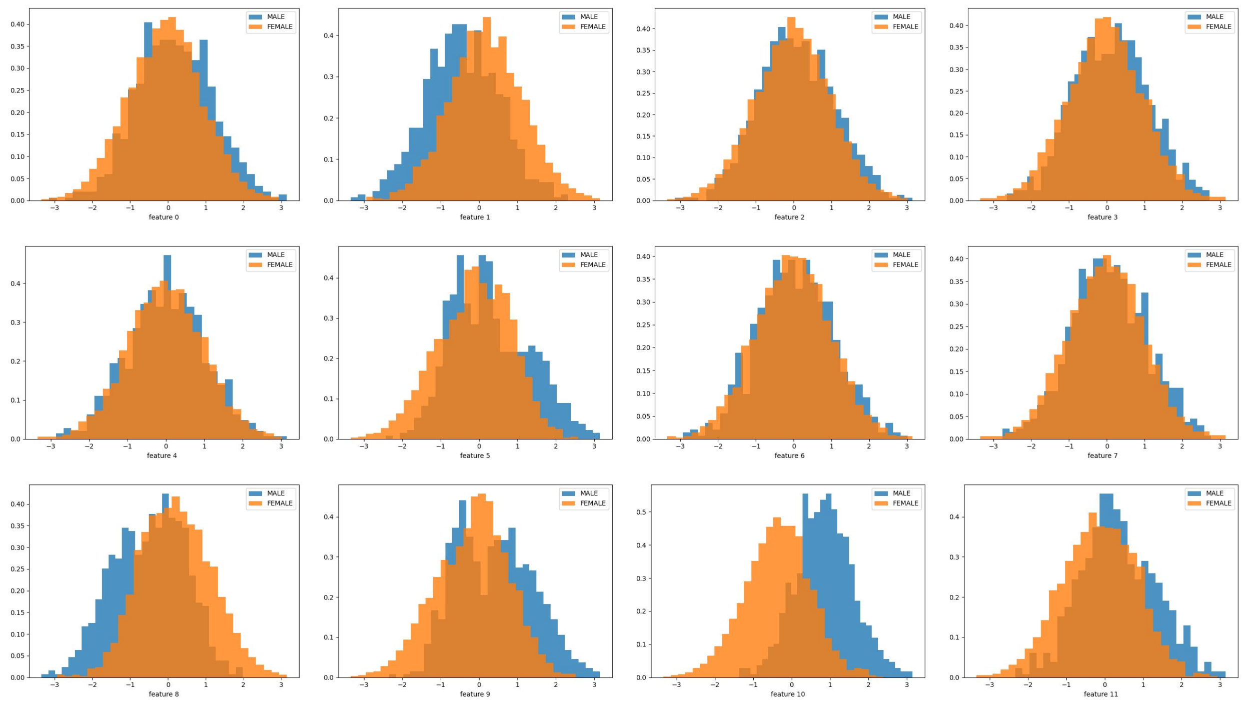


Figure 3: Gaussianized Features

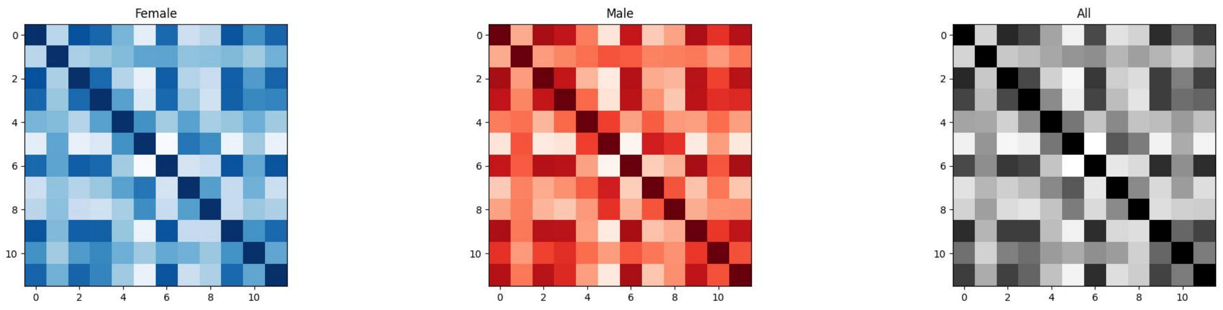


Figure 4: Pearson Correlation - Gaussianized features

2 Training Classification

Our main application will be a uniform prior one (0.5,1,1), but we will also consider unbalanced application (0.1,0.9). We chose to employ K-Fold cross-validation, with 5 folds, in order to have more data available for training and validation and to obtain more robust results. Data has been shuffled before splitting, so that the data of different folds are homogeneous.

2.1 Multivariate Gaussian Classifiers

We start considering Gaussian classifiers, both diagonal and full-covariance models. These classifiers belong to the generative model category, where the data is assumed to follow a Gaussian distribution given the class:

$$X|C = c \sim N(\mu_c, \Sigma_c) \tag{1}$$

Given that histograms indicate that the features follow an approximate Gaussian distribution, it is anticipated that Generative Models will perform effectively on this dataset. Additionally, heatmaps reveal a widespread correlation among the features. Hence, it is expected that models based on Naïve Bayes assumptions will exhibit poor performance.

	RAW			Z-Score			Gauss		
Model	0.5	0.1	0.9	0.5	0.1	0.9	0.5	0.1	0.9
FULL	0.113	0.297	0.350	0.113	0.297	0.350	0.140	0.361	0.352
DIAG	0.463	0.771	0.777	0.463	0.770	0.777	0.454	0.758	0.748
TIED - FULL	0.109	0.299	0.342	0.109	0.299	0.342	0.129	0.349	0.346
TIED -DIAG	0.457	0.770	0.780	0.457	0.770	0.780	0.454	0.762	0.765

it can be seen that the values, applying z-score do not differ at all from the raw data, instead applying gaussianization worsens the results.

PCA(11)	RAW			Z-Score			Gauss		
Model	0.5	0.1	0.9	0.5	0.1	0.9	0.5	0.1	0.9
FULL	0.117	0.310	0.353	0.119	0.315	0.356	0.144	0.365	0.356
DIAG	0.126	0.318	0.370	0.123	0.298	0.345	0.136	0.386	0.345
TIED - FULL	0.117	0.294	0.357	0.117	0.299	0.358	0.131	0.358	0.353
TIED -DIAG	0.120	0.298	0.364	0.124	0.285	0.350	0.132	0.343	0.358

We can see that by applying PCA the results are slightly worse, but the trend does not change, in fact the best results for each application are given by the tied version of full covariance. We note, however, that the results obtained from the naive hypothesis, despite being generally worse than the full, are definitely improved over those seen previously.

Although not reported here, versions with smaller features numbers of the PCA were also tested, but turns out that the results get much worse.

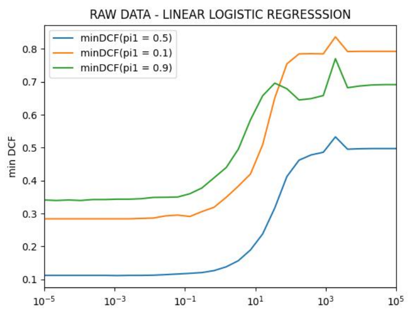
2.2 Logistic Regression

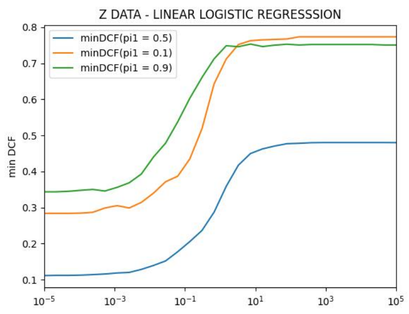
2.2.1 Linear Logistic Regression

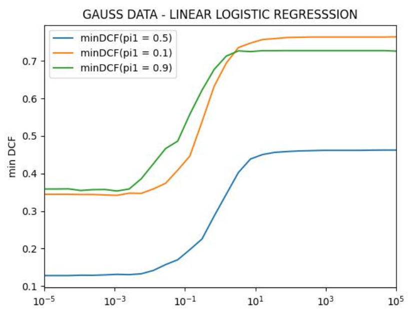
To account for various applications with distinct priors, we have incorporated a prior-weighted version of Logistic Regression. In simpler terms, the objective function takes into consideration the priors for different scenarios. The objective function we implemented is denoted as $R(w)$ and can be expressed as follows:

$$R(w) = \lambda \|w\|_2^2 + \frac{\pi^T}{n_T} \sum_{i:z_i=1} l(z_i, s_i) + \frac{1-\pi^T}{n_F} \sum_{i:z_i=-1} l(z_i, s_i) \tag{2}$$

In this expression, λ represents a regularization parameter, $|w|_2^2$ denotes the squared Euclidean norm of the weight vector w, π is a vector of prior weights, n_T represents the number of positive samples, n_F represents the number of negative samples, and $l(z_i, s_i)$ is a loss function that computes the loss between the true label z_i and the predicted label s_i for the $i-th$ sample. Considering that the 'TIED FULL' method has shown effective linear separation rules and has yielded favorable outcomes, it is anticipated that Logistic Regression will exhibit strong performance as well. The plots show how minDCF is affected by different values of λ in order to calibrate it.







We can see from the graphs that as the value of lambda increases, the results begin to deteriorate. Again we analyzed both types of pre-processing, although we saw that with gaussianization the results were worse. We took 10^-5 as the value of lambda, although up to 10^-3 the results are stable for all three graphs.

$\lambda = 10^-5$	RAW			Z-Score			Gauss		
Model	0.5	0.1	0.9	0.5	0.1	0.9	0.5	0.1	0.9
FULL	0.113	0.297	0.350	0.113	0.297	0.350	0.140	0.361	0.352
TIED - FULL	0.109	0.299	0.342	0.109	0.299	0.342	0.129	0.349	0.346
LogReg($\pi = 0.5$)	0.112	0.284	0.341	0.111	0.284	0.343	0.128	0.344	0.359
LogReg($\pi = 0.1$)	0.120	0.299	0.376	0.121	0.298	0.376	0.131	0.340	0.367
LogReg($\pi = 0.9$)	0.111	0.315	0.344	0.111	0.315	0.344	0.130	0.355	0.355

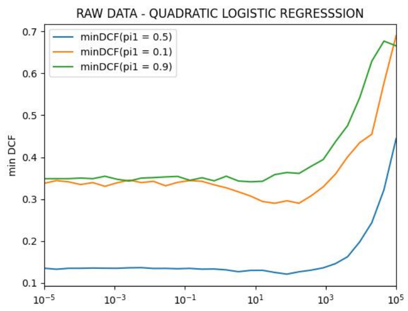
As expected the results of linear logistic regression are certainly comparable with the tied version of MVG. We can see that for the other applications the results are little better, but not so significant as to prefer the latter. Once again, gaussianization performs worse than the others, and z-score normalization gets the same results as the raw version.

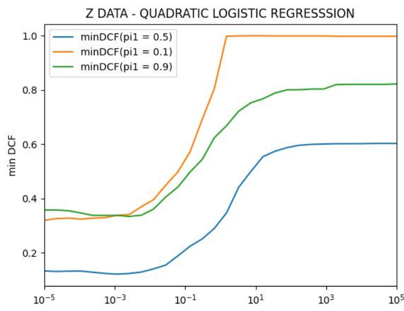
2.2.2 Quadratic Logistic Regression

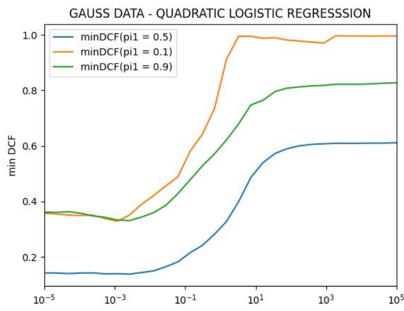
It is feasible to establish non-linear separation rules by constructing an expanded feature space, defined as follows:

$$\phi(x) = \begin{pmatrix} \text{vec}(xx^T) \\ x \end{pmatrix} \tag{3}$$

In this expression, x represents the input feature vector, and $vec()$ denotes the vectorization operation. The transformation $\phi(x)$ expands the feature space by including both the original feature vector x and the vectorized outer product xx^T. So, we can train the Logistic Regression model using the aforementioned definition, but this time utilizing the feature vectors $\phi(x)$ instead of x. By employing $\phi(x)$, we are able to compute linear separation rules in the transformed feature space, which corresponds to estimating quadratic separation surfaces in the original space.







Again, regularization is effective, however, we can already see from the graphs that the results are worse, which the table below confirms.

$\lambda = 10^{-}2$	RAW			Z-Score			Gauss		
Model	0.5	0.1	0.9	0.5	0.1	0.9	0.5	0.1	0.9
FULL	0.113	0.297	0.350	0.113	0.297	0.350	0.140	0.361	0.352
TIED - FULL	0.109	0.299	0.342	0.109	0.299	0.342	0.129	0.349	0.346
Quad LogReg($\pi = 0.5$)	0.134	0.335	0.347	0.125	0.335	0.335	0.137	0.329	0.333
Quad LogReg($\pi = 0.1$)	0.144	0.351	0.378	0.132	0.354	0.360	0.143	0.405	0.361
Quad LogReg($\pi = 0.9$)	0.142	0.391	0.393	0.136	0.322	0.313	0.141	0.350	0.352

Based on our observations, it can be concluded that linear decision rules result in better separation between classes, but we can notice that in this case the z-score normalization outperforms the other. The investigation for this particular application proceeds with the training of Support Vector Machines (SVMs) and Gaussian Mixture Models (GMMs). We will initially focus on SVMs and anticipate favorable outcomes. Seeing that gaussianization is not very effective, we decided to abandon it, as well as with PCA.

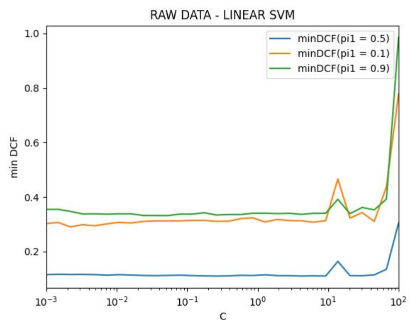
2.3 Support Vector Machines

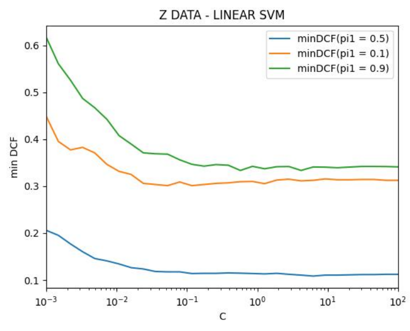
2.3.1 Linear SVM

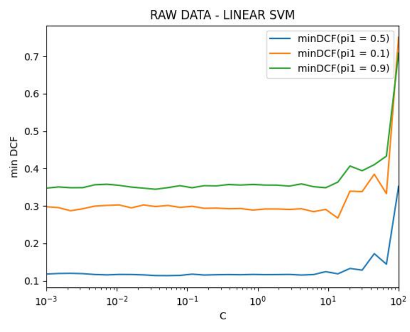
The key idea behind linear SVM is to find a hyperplane that maximizes the margin between the classes. The margin is defined as the distance between the hyperplane and the nearest data points from each class. By maximizing the margin, linear SVM seeks to achieve better generalization and robustness to unseen data. The objective is to minimize the classification error while simultaneously maximizing the margin, so the objective can be represented as follows:

$$J_b(w_b) = \frac{1}{2}\|w_b\|^2 + C\sum_{i=1}^{n}\max(0, 1 - z_i w_b^T x_{bi}) \tag{4}$$

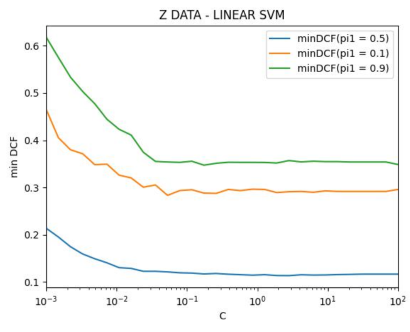
In this expression, $J_b(w_b)$represents the objective function, $\|w_b\|^2$ denotes the squared Euclidean norm of the weight vector w_b, C is the regularization parameter that controls the trade-off between margin maximization and misclassification penalty, n represents the number of training samples, z_i is the true label of the i_{th} training sample, x_{bi}represents the i_{th} feature vector, and $\max(0, 1-z_i w_b^T x_{bi})$ is the hinge loss term that penalizes misclassifications.







balanced version



balanced version

Above we have the graph showing us the minDCF as the regularization parameter C changes, both classic (the two above) and balanced versions are shown. We can see that there are no substantial differences between the two versions.

$C = 1$	RAW			Z-Score		
Model	0.5	0.1	0.9	0.5	0.1	0.9
TIED - FULL	0.109	0.299	0.342	0.109	0.299	0.342
Linear SVM ($\pi = 0.5$)	0.116	0.291	0.355	0.115	0.295	0.351
Linear SVM ($\pi = 0.1$)	0.130	0.294	0.395	0.127	0.305	0.384
Linear SVM ($\pi = 0.9$)	0.116	0.327	0.335	0.112	0.314	0.340

After seeing that for $C = 1$ the best results were obtained in the main application, we tested the other applications as well. As we expected, the results are in line with the linear classifiers, but it fails to improve performance over the tied version of the MVG

2.3.2 Quadratic SVM

The key idea behind quadratic SVM is to transform the original feature space into a higher-dimensional space, where non-linear patterns become linearly separable. This transformation is achieved through a kernel function, such as the polynomial or radial basis function (RBF) kernel, which maps the data into a higher-dimensional space. For the polynomial kernel of degree d, the kernel it is represented by:

$$k(x_1, x_2) = (x_1^T x_2 + 1)^d \tag{5}$$

Only minDCF values based on parameter C are reported here, as the results are poor.

d = 2	RAW			Z-Score		
Model	0.5	0.1	0.9	0.5	0.1	0.9
C = 0.1	0.140	0.403	0.403	0.605	0.998	0.956
C = 1	0.256	0.605	0.605	0.604	0.998	0.955
C = 10	0.289	0.650	0.612	0.606	0.998	0.957
C = 100	0.346	0.967	0.996	0.607	0.999	0.960

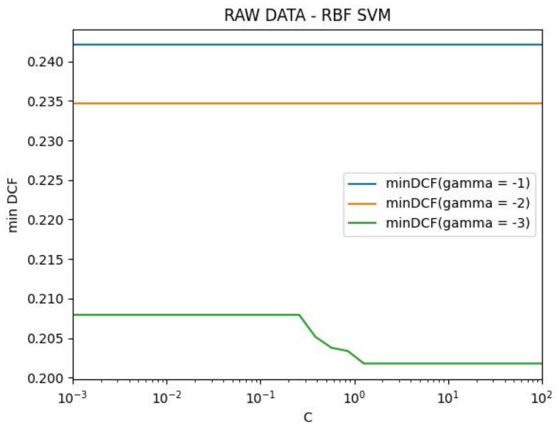
As we expected, again the quadratic version does not perform very well, indeed in this case with z-score normalization we have very bad results

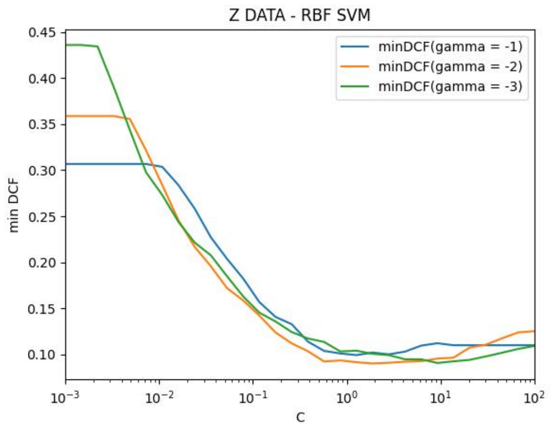
2.3.3 RBF SVM

The SVM with a Radial Basis Function (RBF) kernel is obtained by solving the same problem as mentioned earlier. However, in this case, the kernel function used is defined as follows:

$$k(x_1, x_2) = e^{-\gamma \|x_1 - x_2\|^2} \tag{6}$$

The generated plot for this specific SVM case differs from the previous ones. Notably, it provides the ability to simultaneously tune two hyperparameters: C and γ. By visualizing this plot, it becomes possible to explore and optimize the performance of the SVM model by adjusting both C and γ simultaneously.





as we can see, as far as the raw data graph is concerned, only for $log\gamma = -3$ does the regularization make sense, for the other values of γ in addition to not being useful, the results are quite bad. On the other hand, regarding z-score normalization, we can see that for the 3 range values tested there is more or less the same performance, in all 3 cases regularization is needed. In our case, after evaluating the different parameters, we decided to use the following values: $C = 10$ and $log\gamma = -3$

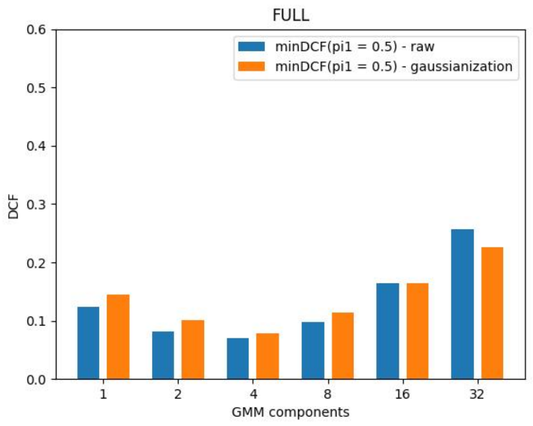
$C = 10, log\gamma = -3$	RAW			Z-Score		
Model	0.5	0.1	0.9	0.5	0.1	0.9
TIED - FULL	0.109	0.299	0.342	0.109	0.299	0.342
RBF SVM	0.202	0.564	0.433	0.094	0.252	0.297

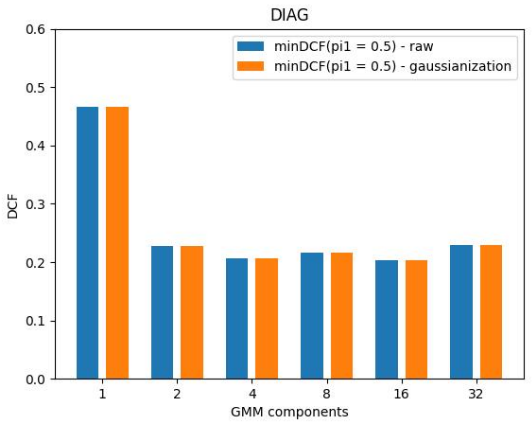
As we can see from the values given in the table above, the SVM with an RBF kernel, outscales the performance of the best classifier so far, the Tied-Full MVG, in every application

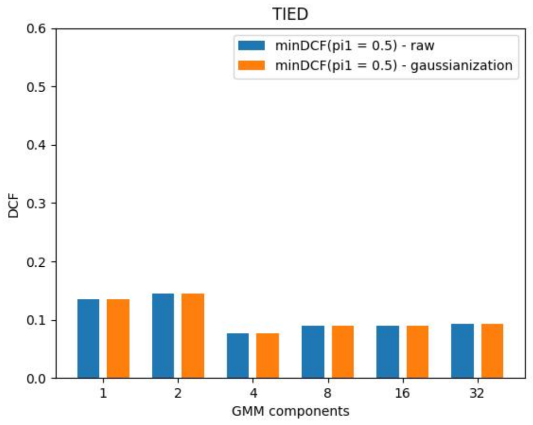
2.4 Gaussian Mixture Models

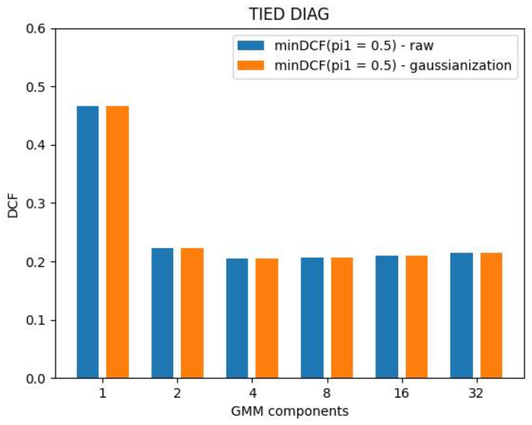
Gaussian Mixture Models (GMMs) are probabilistic models used for modeling complex data distributions. GMMs represent the data as a mixture of multiple Gaussian distributions, each with its own mean and covariance matrix. The key idea behind GMMs is to approximate the underlying probability density function of the observed data using a weighted combination of Gaussian distributions. Each Gaussian component represents a cluster or subpopulation within the data.
Given that GMMs have the capability to approximate various types of distributions, we

anticipate achieving superior results compared to using a single Gaussian model. In our analysis, we will train GMMs with up to 32 components for different covariance types, including full, diagonal, and tied/non-tied covariances. In this case we revert to gaussianization as a preprocessing method because z-score normalization brings no benefit.









Components	1	2	4	8	16	32
			RAW			
FULL	0.123	0.082	0.071	0.098	0.164	0.257
DIAG	0.465	0.228	0.207	0.216	0.203	0.229
TIED - FULL	0.135	0.145	0.077	0.089	0.090	0.092
TIED -DIAG	0.466	0.223	0.204	0.207	0.210	0.215
			Gauss			
FULL	0.145	0.102	0.078	0.115	0.165	0.225
DIAG	0.465	0.228	0.207	0.216	0.203	0.229
TIED - FULL	0.135	0.145	0.077	0.089	0.090	0.092
TIED -DIAG	0.466	0.223	0.204	0.207	0.210	0.215

$C = 10, log\gamma = -3$	RAW			Z-Score		
Model	0.5	0.1	0.9	0.5	0.1	0.9
TIED - FULL MVG	0.109	0.299	0.342	0.109	0.299	0.342
RBF SVM	0.202	0.564	0.433	0.094	0.252	0.297

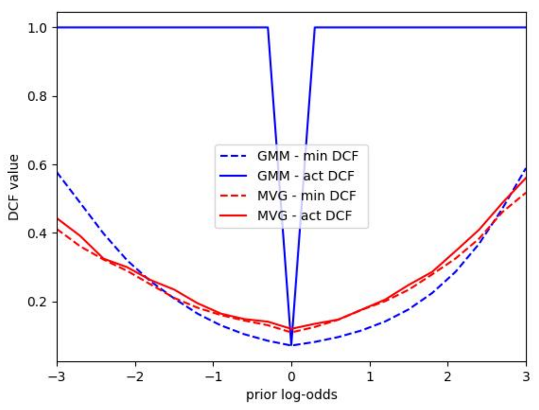
We can see that gaussianization actually never improves performance over raw, but it evens it out on many components. Also, as mentioned earlier the performance is improved over MVG, but also over SVM, indeed we got the best results of the whole analysis. In particular, the FULL model with 4 components achieved the best performance.

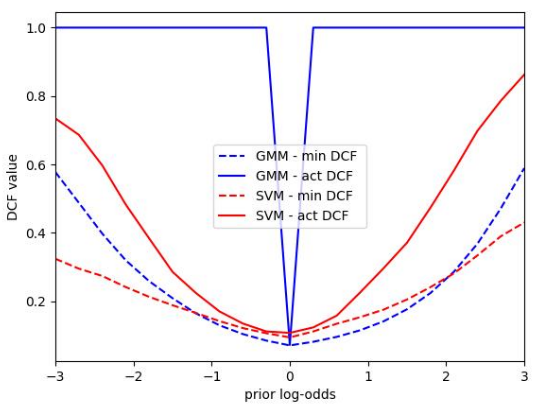
2.5 Score Calibration

Up to now, the models that have given us the best results are the Tied-Full MVG, the SVM with RBF kernel, and the Full GMM with 4 components.
Up until now, we have primarily relied on the minimum detection costs as a metric to evaluate various models. However, an important consideration is that the incurred cost depends on the chosen threshold. To gain a better understanding of whether the threshold aligns with the theoretical optimal value, we will incorporate a metric known as actual Detection Cost Function (DCF). In this assessment, we will utilize the Logistic Regression approach, as it provides a posterior log-likelihood ratio. By subtracting the theoretical threshold from the calibrated score, we can recover the calibrated score and evaluate its alignment with the optimal threshold. This additional analysis using the actual DCF metric will provide a more comprehensive evaluation of the model's performance.

	$\pi = 0.5$		$\pi = 0.1$		$\pi = 0.9$	
	minDCF	actDCF	minDCF	actDCF	minDCF	actDCF
GMM FULL	0.071	0.072	0.343	1.000	0.310	1.000
TIED - FULL MVG	0.109	0.120	0.299	0.310	0.342	0.371
RBF SVM	0.094	0.107	0.251	0.518	0.297	0.612

Looking at this table we can see that for L'MVG the scores, they are fairly calibrated, whereas for the other two models they are fairly scaled and need to be re-calibrated. And this we can see directly from the graphs below. Indeed, we tried to analyze the GMM in relation to the other two models that performed better, over a larger range of applications

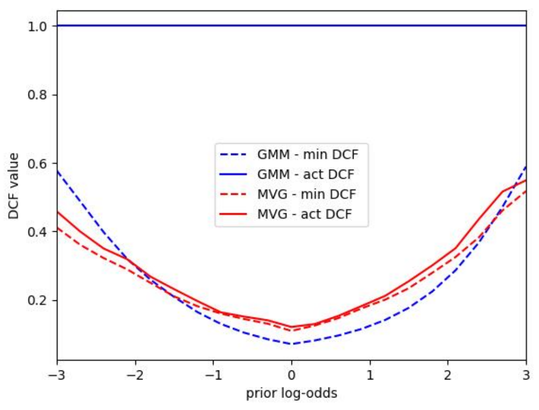




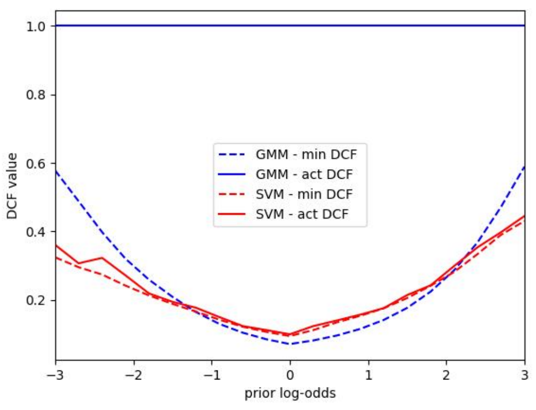
Generally it can be observed that under a minDCF point of view the GMM performs better than the other two, the problem is the uncalibration of the scores. We then tried to recalibrate the scores, getting very good results for the SVM, but failing to recalibrate the GMM.

	$\pi = 0.5$		$\pi = 0.1$		$\pi = 0.9$	
	minDCF	actDCF	minDCF	actDCF	minDCF	actDCF
GMM FULL	0.071	0.225	0.343	0.561	0.310	0.592
TIED - FULL MVG	0.109	0.120	0.299	0.305	0.342	0.375
RBF SVM	0.094	0.099	0.251	0.278	0.297	0.320

Calibrated



Calibrated



2.6 Score Fusion

To further enhance performance, we can explore the combination of different classifiers to leverage their diverse decision-making capabilities. In this regard, we aim to fuse the outputs of the three best models: MVG, Support Vector Machines (SVM) and Gaussian Mixture Models (GMM).

For the fusion process, we will adopt a similar approach as calibration, utilizing the K-Fold method. By employing prior-weighted logistic regression, we can effectively combine the decisions of the individual models. This fusion technique allows us to leverage the strengths of each classifier and potentially achieve improved overall performance.

	$\pi = 0.5$		$\pi = 0.1$		$\pi = 0.9$	
	minDCF	actDCF	minDCF	actDCF	minDCF	actDCF
GMM - MVG	0.109	0.120	0.303	0.306	0.350	0.378
GMM - SVM	0.095	0.099	0.252	0.285	0.297	0.326
GMM - MVG - SVM	0.095	0.101	0.233	0.251	0.285	0.296

So we see that the fusion score does not bring any benefit in the main application, however, the scores are quite calibrated. So our best model remains the Full - GMM with 4 components, and we expect it to perform the same way on the test set as well.

3 Testing

Now, let us focus on the evaluation phase. To assess the performance, it is advantageous to consider the minimum Detection Cost Function (DCF) as the evaluation metric. The minimum DCF provides an optimistic estimate for the actual DCF, assuming an optimal threshold selection for the evaluation set.

Since the validation process involved 5-fold cross-validation, the evaluation phase will be executed using the complete dataset. All the training data will be utilized for training the models, while all the test data will be employed for the final evaluation. This comprehensive evaluation approach allows us to assess the models' performance using the entire dataset, thereby obtaining a more accurate representation of their capabilities.

3.1 Multivariate Gaussian Classifiers

Here are reported the evaluation results for MVG:

	RAW			Z-Score			Gauss		
Model	0.5	0.1	0.9	0.5	0.1	0.9	0.5	0.1	0.9
FULL	0.118	0.312	0.312	0.120	0.312	0.312	0.140	0.348	0.339
DIAG	0.434	0.817	0.705	0.434	0.817	0.705	0.423	0.770	0.685
TIED - FULL	0.116	0.301	0.308	0.116	0.301	0.308	0.133	0.338	0.317
TIED - DIAG	0.435	0.815	0.711	0.435	0.814	0.711	0.426	0.797	0.696

PCA(11)	RAW			Z-Score			Gauss		
Model	0.5	0.1	0.9	0.5	0.1	0.9	0.5	0.1	0.9
FULL	0.123	0.321	0.332	0.126	0.324	0.331	0.138	0.354	0.338
DIAG	0.130	0.355	0.323	0.143	0.404	0.350	0.137	0.368	0.349
TIED - FULL	0.121	0.314	0.323	0.123	0.314	0.313	0.130	0.340	0.317
TIED - DIAG	0.124	0.327	0.320	0.140	0.323	0.370	0.128	0.315	0.294

Again, we tried all three types of preprocessing that we saw in the training phase, and as expected, gaussianization performs slightly worse than the others, as does PCA, which, however, gets better results in Naive models. The results however are in line with what was expected.

3.2 Support Vector Machine

Now we consider the no probabilistic models such as SVM and Radial Based Version, given that the quadratic one does not give good result. In summary, the chosen values for the model parameters in each implemented version of SVM, determined during the previous phase, are the follows:

- Linear SVM (C = 1)
- RBF SVM (C = 10, $\log\gamma = -3$)

	RAW			Z-Score		
Model	0.5	0.1	0.9	0.5	0.1	0.9
Linear SVM	0.118	0.309	0.295	0.117	0.307	0.297
RBF SVM	0.189	0.636	0.446	0.085	0.222	0.226

We can see that again the results are in line with those of the validation phase. by the way, the RBF performance exceeded expectations

3.3 Gaussian Mixture Models

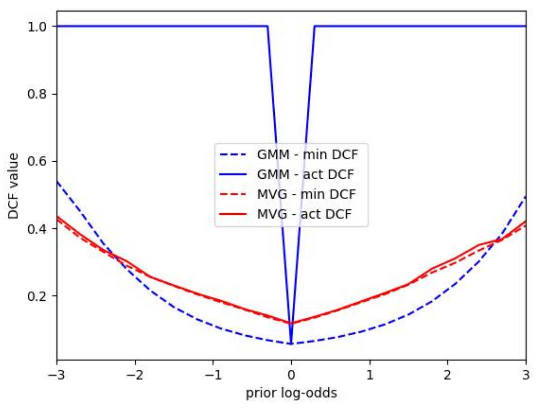
Here are reported the evaluation results for GMM:

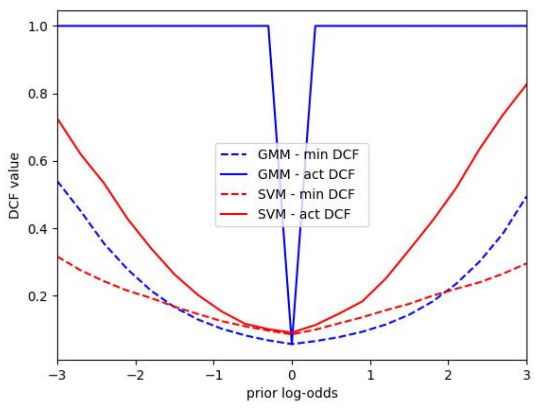
Components	1	2	4	8
		RAW		
FULL	0.123	0.078	0.062	0.086
DIAG	0.445	0.215	0.183	0.182
TIED - FULL	0.117	0.124	0.056	0.068
TIED -DIAG	0.445	0.215	0.183	0.182
		Gauss		
FULL	0.143	0.099	0.080	0.098
DIAG	0.427	0.223	0.192	0.186
TIED - FULL	0.136	0.142	0.071	0.080
TIED -DIAG	0.435	0.218	0.192	0.191

In this case, the Tied-Full version is better than the full version, although the latter is still better than the validation phase

3.4 Score Calibration

Similar to the validation phase, it is important to compare the models in terms of minDCF and actDCF using Bayes Error Plots. In this case we compare the GMM with 4 components and Tied Full Covariance Matrix, SVM with a kernel RBF and MVG with Tied Full Covariance Matrix, only the SVM model used z-score preprocessing, the other uses raw data.

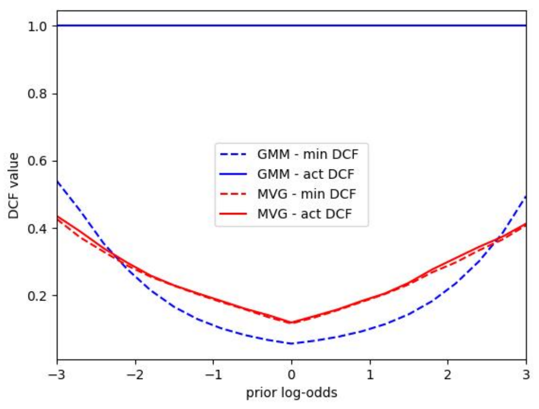


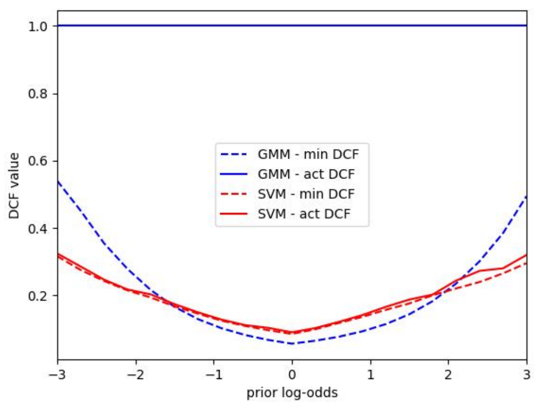


Again, the results are quite similar to the validation phase, where the MVG scores are calibrated instead those of the other two models are not at all.

Calibrated

Calibrated





after calibration as expected, SVM scores come out calibrated, instead GMM scores not at all.

3.5 Score Fusion

Finally, as in the validation phase, we tried to merge the scores of the best models

	$\pi = 0.5$		$\pi = 0.1$		$\pi = 0.9$	
	minDCF	actDCF	minDCF	actDCF	minDCF	actDCF
GMM - MVG	0.117	0.119	0.303	0.314	0.308	0.315
GMM - SVM	0.086	0.090	0.223	0.229	0.238	0.245
GMM - MVG - SVM	0.084	0.086	0.215	0.227	0.236	0.246

The results show again this time that fusing the scores does not benefit performance, but that merging them results in a well-calibrated model.

4 Conclusion

In conclusion, our approach, consists on the choice of the GMM - 4 components with Tied Full Covariance Matrix. Even if we choice another model in validation phase (GMM Full with 4 components), turns out that in evaluation phase that the model with Tied Covariance matrix outperforms it. We are able to achieve a DCF cost of ≈ 0.05 on the test for our main application ($\pi = 0.5$) but we don't get good results in unbalanced version, with imbalanced prior $\pi = 0.1$ (DCF cost of ≈ 0.3) and for the one with prior $\pi = 0.9$ (DCF cost of ≈ 0.2) .