Machine learning
and
Pattern Recognition

Project:

Gender Identification

Antonio Ferrigno [s316467]

Simone Varriale [s315962]

Index

Figures index

This project is intended to show a binary classification task on a dataset made of 12 continuous observations coming from face images embeddings. The embedding components does not have a physical interpretation and the samples belong to 3 different age groups, but no information about age is provided. Features represent points in the m-dimensional embedding space. This task does not have balanced classes for both training and evaluation set.

1. Dataset analysis

1.1. Training and evaluation sets

The datasets provided are:

- Training Set: 720 samples belonging to Male class (Label = 0) and 1680 samples belonging to Female class (Label = 1).
- Evaluation Set: 4200 samples belonging to Male class (Label = 0) and 1800 samples belonging to Female class (Label = 1).

We will use the Training Set to perform all the analysis and only once we have selected the most promising models, we will train these ones using the entire Training Set and final considerations will be done considering their behavior on the Evaluation Set.

1.2. Features analysis

All the features are contiguous and here it is the plot of the raw features through histograms that will show how the features are distributed:

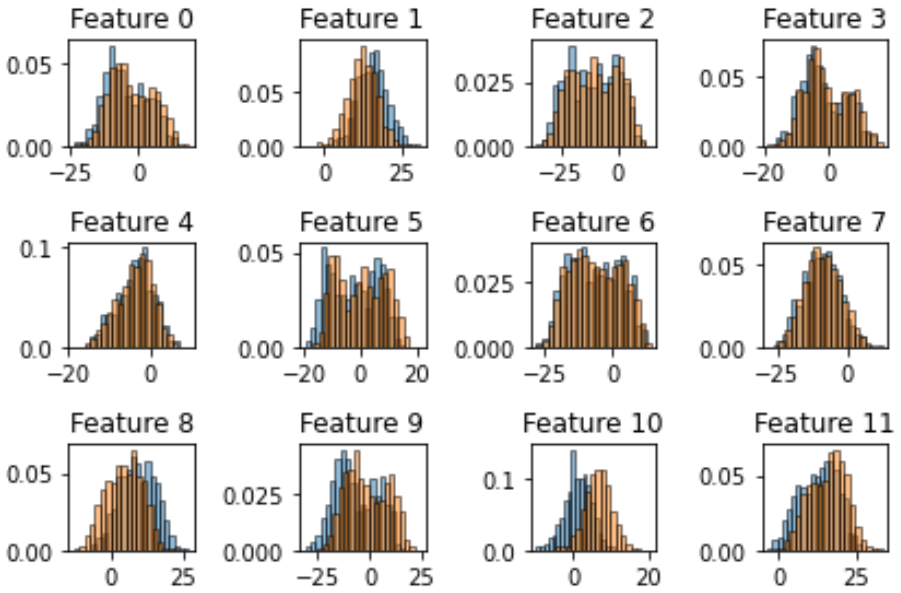

Figure 1: Raw features distribution

By plotting one histogram for each feature, separately for classes male and female, it is possible to show if the samples follow a Gaussian distribution and how well. Immediately, there is the possibility to say that the a large number of feature already follow a Gaussian distribution. Also, it is possible to notice how feature 10 can be the most discriminant feature based on how well it separates the data for the two classes.

To deal with more uniform data, possible preprocessing technique could be applied. For this reason, an explanation about Z-normalization and Gaussianization techniques are provided.

1.3. Z-normalization

A useful operation that can be applied to avoid dealing with numerical issues and to make data more uniform is to apply Z-normalization as a preprocessing step. We are going to transform each sample of our dataset as:

$$z_{i,j} = \frac{x_{i,j} - \mu_j}{\sigma_j} \forall x_i \in D$$

Where $z_{i,j}$ is the Z-normalized value corresponding to the feature j of sample i while μ_j and σ_j are, respectively, the mean and the variance computed over all the values for feature j.

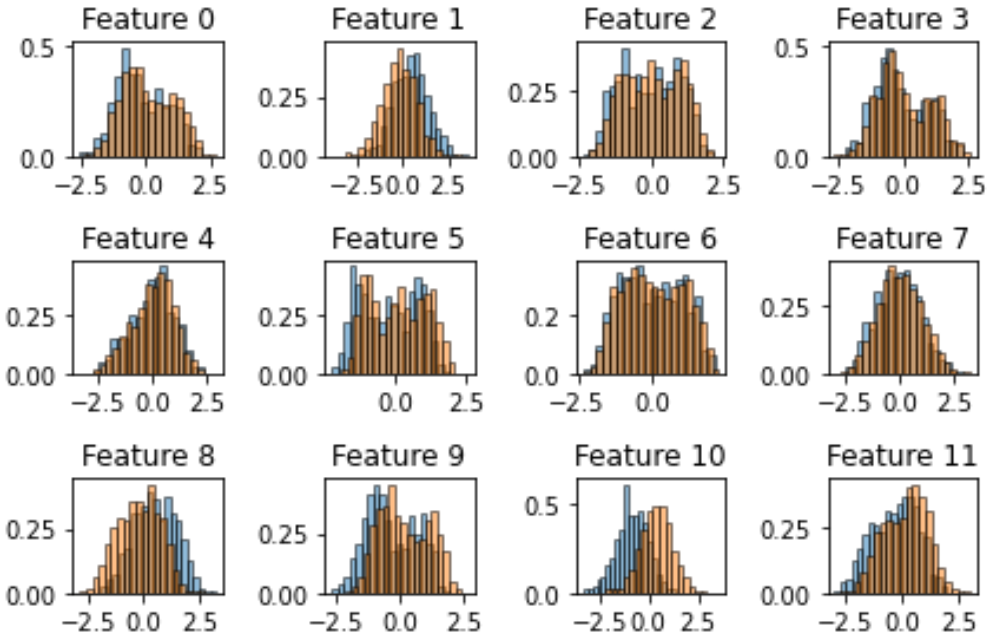

Figure 2: Z-normalized features distribution

We can notice from figure 2 that almost all the features are already well-distributed (w.r.t. the Gaussian distribution) except for features 2, 3, 5, 6, 9. Precisely, the feature 2-5-6 also reflects the three different age groups of the samples. We can thus apply an additional pre-processing step to make all the features following a Gaussian distribution: this step is called Gaussianization, and in this case it will be always applied after Z-Normalization preprocessing.

1.4. Gaussianization

Gaussianization is a pre-processing step that maps each feature to values whose empirical cumulative distribution is well approximated by a Gaussian cumulative distribution function. For each feature x to gaussianize the rank over the dataset must be computed:

$$r(x) = \frac{\sum_{i=1}^{N} \mathbb{I}[x_i < x] + 1}{N + 2}$$

where I is the indicator function (1 when the condition inside $[x_i < x]$ is true, 0 otherwise). We are counting how many samples in the dataset D have a greater value with respect to the feature we are computing the rank on. The next step is to compute the transformed feature as $y = \Phi^{-1}(r(x))$ where Φ is the inverse of the cumulative distribution function.

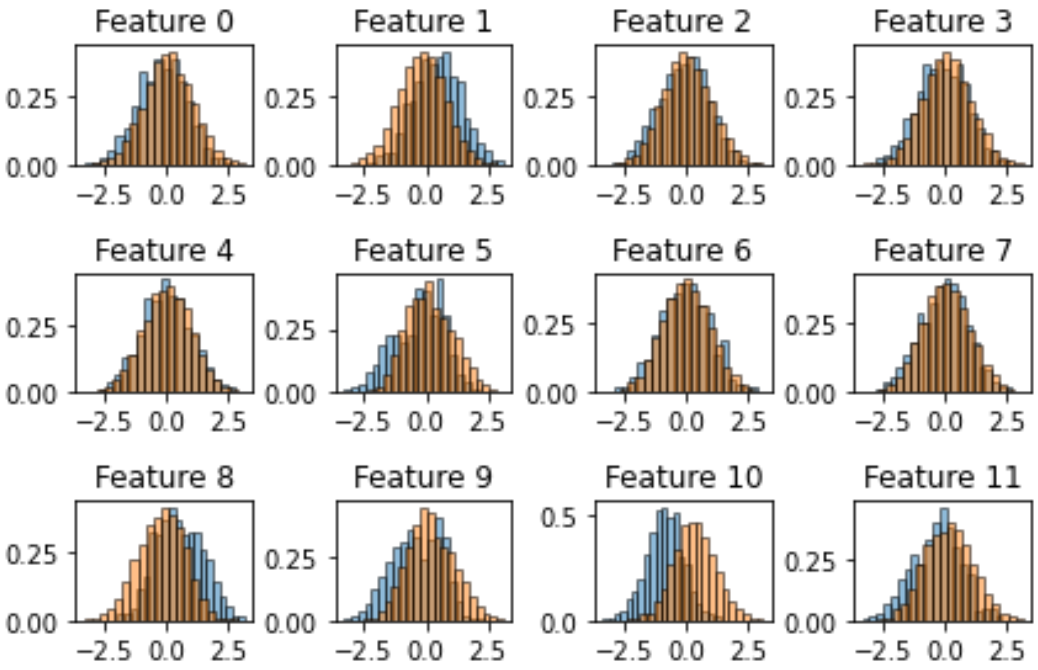

Figure 3: Gaussianized features distribution.

1.5. Features correlation

We can show how much features are correlated by using a heatmap plot showing a darker color inside cells for which it exists a high correlation among feature i and feature j. We are going to use the Pearson correlation coefficient to compute the correlation among feature X and feature Y:

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

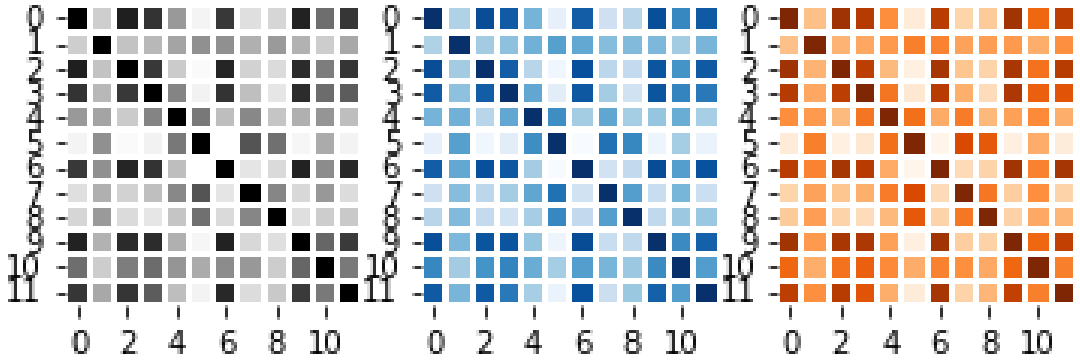

Figure 4: Z-normalized features correlation: grey the whole dataset, orange F class, blue M class

Features are moderately correlated among the dataset, only feature 5 seems to be slightly uncorrelated from the others. There are different features that are strongly correlated with the others like feature "0" due the high number of dark cells.

Continuing the analysis, the heatmap of the two classes are very similar to each other, so it suggests that Full Covariance Gaussian model and the Tied Gaussian one will have similar performance, while the Naïve Bayes assumption will lead to scarce results, since for both classes the feature are moderately correlated, and the latter model assume they are not.

From the the correlation of the feauture it is possible to understand that applying PCA would be meaningful for values of m not below 11 or 10 at most. For smaller values of m, we would probably lose important information coming from high-correlated features. This will be analyzed more in depth with PCA impact evaluation varying the value of m seeing how much the variance of data will change.

2. Dimensionality reduction

Before proceeding with the classification task we will spend some words on PCA dimensionality reduction technique.

2.1. PCA

As already anticipated, given the heatmap in figure 4 we can observe that PCA with reasonable values of m can be applied. PCA is a dimensionality reduction technique that, given a centered dataset $X = \{x_1, \dots, x_k\}$, it aims to find the subspace of R^n that allows to preserve most of the information, i.e. the directions with the highest variance.
Starting from the empirical covariance matrix

$$C = \frac{1}{K}\sum_i (x_i - \bar{x})(x_i - \bar{x})^T$$

we compute the eigen-decomposition of $C = U\Sigma U^T$ and project the data in the subspace spanned by the m columns of U corresponding to the m highest eigen values:

$$y_i = P^T(x_i - \bar{x})$$

where P is the matrix corresponding to the m columns of U associated the to the m highest eigenvalues of C. Then, to select the optimal value for m, it is possible to use cross-validation approach inspecting how much of the total variance of the data we are able to retain by using different values for m. In order to do that the fact that each eigenvalue corresponds to the variance along the corresponding axis and the eigenvalues are the elements of the diagonal of the matrix Σ is exploited. The percentage will be computed as the ratio between the sum of the first m eigenvalues and the sum of all the eigenvalues.

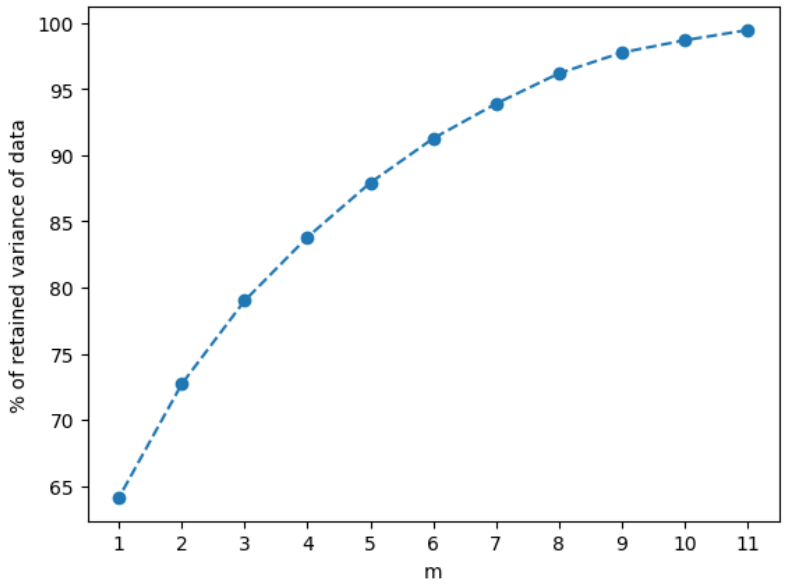

Figure 5: 5-fold cross validation for PCA impact evaluation

We can clearly understand from figure 5 that values of $m < 9$ would rapidly decrease the amount of retained variance of the data. Removing one or two features is probably the best choice because we will maintain 96%-97% of variance of data.

3. Premises

To understand which model is most promising and to assess the effects of the different pre-processing techniques, we will employ the K-Fold cross validation with k=5. This methodology has been chosen to have more data for training and validation.

Three types of applications are considered:

$$(\tilde{\pi}, C_{fp}, C_{fn}) = (0.1,1,1)$$
$$(\tilde{\pi}, C_{fp}, C_{fn}) = (0.5,1,1)$$
$$(\tilde{\pi}, C_{fp}, C_{fn}) = (0.9,1,1)$$

and the target application will be the balanced one:

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

The goal of the following analysis is to choose the model which performances promise to be the best. This performance will be measured in terms of normalized minimum detection costs. This technique would allow us to discriminate the top-performing model for our target application giving an assessment of how well the model detect the desired category.

It is defined in the following way:

$$DCF = \frac{DCF_u(\pi_T, C_{fn}, C_{fp})}{min\left(\pi_T, C_{fn}, (1-\pi_T C_{fp})\right)} = \frac{\pi_T C_{fn} P_{fn} + (1-\pi_T) C_{fp} P_{fp}}{min\left(\pi_T, C_{fn}, (1-\pi_T C_{fp})\right)}$$

and for minDCF computation we will look for the threshold:

$$t' = -\log\left(\frac{\tilde{\pi}}{1-\tilde{\pi}}\right)$$

that allows us to obtain the lowest possible DCF.

4. Classification models analysis

4.1. Gaussian models

The first class of models we are going to analyze are the generative Gaussian models. We assume that, given the dataset X, the sample x_t is a realization of the R.V. X_t. A simple model consists in assuming that our data, given the class, can be described by a Gaussian distribution:

$$(X_t \mid C_t = c) \sim (X \mid C = c) \sim \mathcal{N}(x_t \mid \mu_c, \Sigma_c)$$

Since we are dealing with a binary classification task we will assign a probabilistic score to each sample in terms of the class-posterior log-likelihood ratio:

$$\text{llr}\,(x_t) = \log\, r(x_t) = \log \frac{P(C = h_1 \mid x_t)}{P(C = h_0 \mid x_t)}$$

We can expand this expression by writing:

$$\log r(x_t) = \log \frac{f_{X|C}(x_t \mid h_1)}{f_{X|C}(x_t \mid h_0)} + \log \frac{\pi}{1-\pi}$$
$$\text{with } f_{X|C}(x_t \mid c) = \mathcal{N}(x \mid \mu_c, \Sigma_c)$$

The training phase will consist in estimating the model parameters of the Multivariate Gaussian distribution (μ_c, Σ_c), while the scoring phase consists in computing the log-likelihood ratio for each sample and then compare it with a threshold specific for each application (since it depends on the prior) to compute the minDCF. There are different Gaussian model and they are differentiated by how the model parameters are computed.

4.1.1. MVG Gaussian Classifier

The ML solution to the previous described problem is given by the empirical mean and covariance matrix for each class:

$$\mu_c^* = \frac{1}{N_c} \sum_{i=1}^{N} x_{c,i}$$
$$\Sigma_c^* = \frac{1}{N_c} \sum_{i=1}^{N} (x_{c,i} - \mu_c^*)(x_{c,i} - \mu_c^*)^T$$

4.1.2. Naive Bayes Classifier

The Naive Bayes assumption simplifies the MVG full covariance model stating that for each class the components are approximately independent, so the distribution X|C can be factorized over its components. The ML solution to this problem is:

$$\mu_{c,[j]}^* = \frac{1}{N_c} \sum_{i|c_i=c} x_{i,[j]}$$
$$\sigma_{c,[j]}^2 = \frac{1}{N_c} \sum_{i|c_i=c} (x_{i,[j]} - \mu_{c,[j]})^2$$

The density of a sample x can be expressed as $N(x|\mu_c,\Sigma_c)$ where μc is an array where each element $\mu_{c,[j]}$ is the the mean for each class for each component while Σ_c is a diagonal covariance matrix. The Naïve Bayes classifier corresponds to the MVG full covariance classifier with a diagonal covariance matrix.

4.1.3. Tied Gaussian Classifier

This model assumes that the covariance matrices of the different classes are tied, so that the covariance matrix is common to all classes, but each class still has its own mean. It assumes that:

$$f_{X|C}(x \mid c) = \mathcal{N}(x \mid \mu_c, \Sigma)$$

The ML solution to this problem is:

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

This model is strongly related to LDA (used as a linear classification model). By considering the binary log-likelihood ratio of the tied model we obtain a linear decision function:

$$\text{llr}(x) = \log \frac{f_{X|C}(x \mid h_1)}{f_{X|C}(x \mid h_0)} = x^T b + c$$

where b and c are functions of class means and (tied) covariance matrix. On the other hand, projecting over the LDA subspace is, up to a scaling factor k, given by:

$$w^T x = k \cdot x^T \Lambda(\mu_1 - \mu_0)$$

where Λ (μ1 – μ0) = b. The LDA assumption that all the classes have the same within class covariance matrix is related to the assumption done for the tied model.

4.1.4. Gaussian Models Comparison

Table 1: *Gaussian Models*

	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.9$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.9$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.9$
	Raw			Gaussianized			Z-Norm		
	No PCA								
Full Cov	0.297	0.113	0.350	0.298	0.122	0.353	0.297	0.113	0.350
Tied Cov	0.299	0.109	0.342	0.306	0.118	0.353	0.299	0.109	0.342
Naïve Bayes	0.771	0.463	0.777	0.763	0.466	0.759	0.771	0.463	0.777
	PCA (m=11)								
Full Cov	0.310	0.117	0.353	0.312	0.123	0.350	0.315	0.119	0.356
Tied Cov	0.294	0.117	0.357	0.309	0.119	0.354	0.299	0.117	0.358
Naïve Bayes	0.318	0.127	0.371	0.339	0.128	0.382	0.298	0.123	0.345
	PCA (m = 10)								
Full Cov	0.389	0.161	0.492	0.395	0.170	0.502	0.415	0.184	0.538
Tied Cov	0.389	0.162	0.478	0.429	0.167	0.505	0.429	0.182	0.534
Naïve Bayes	0.443	0.168	0.477	0.416	0.172	0.525	0.407	0.183	0.546

A graphical version of the table can be helpful in analyzing results:

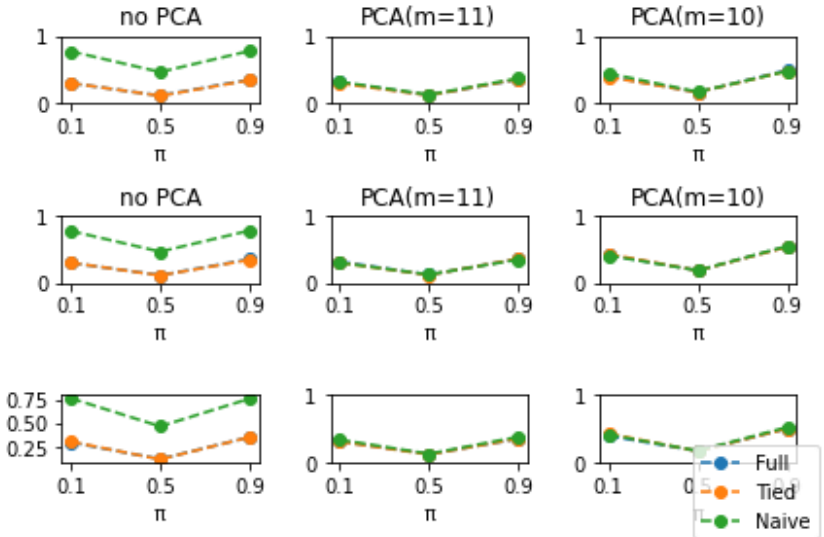

Figure 6: Top: Raw features, Center: Z-Normalized features, Bottom: Gaussianized features

- An initial comparison of the Full Covariance (Full Cov), Tied Covariance (Tied Cov) and Naive Bayes reveals what was expected from the analysis of the heatmap, hence the Full Cov and Tied Cov produced good result also without any kind of preprocessing step, instead the performance of Naïve Bayes are worst compared to the previous model on data without applying PCA.
- The Full Cov and Tied Cov models exhibit consistent results across all preprocessing techniques with a slight edge towards the Tied Cov model, especially when no PCA is applied. This likely indicates that the covariance matrices of the different classes are closely aligned. As such, this would suggest that the Tied Cov model provides slightly more robust estimates.
- The Naive Bayes model tends to perform less well, especially when PCA is not utilized. This could suggest that the Naive Bayes assumption, which assumes that the covariance matrix of each class is diagonal, does not hold well for the given dataset without PCA. When PCA is applied, the performance of the Naive Bayes model appears to improve, mainly for the target application, while the other pre-processing techniques do not provide other kind of improvement.
- PCA, when applied at m=11, does not significantly degrade the performance of Full Cov and Tied Cov models. This suggests that the models still possess good decision-making capabilities. However, as the value of m decreases, there's a notable increase of minDCF, implying that the models become less accurate in their decision making.
- In general, the Multi-Variate Gaussian (MVG) classifiers perform best when using the Tied Cov model with raw features (or with Z-normalized features) and without PCA. When PCA with m=11 is applied, performance worsens slightly, but remains comparable to the no-PCA version. This suggests that PCA at m=11 could be an option to potentially avoid overfitting and reduce computational resource needed to compute the results.
- It's also worth noting that Gaussianized preprocessing method does not significantly impact the results, indicating that the data is likely already well distributed according to Gaussian assumptions and in some cases, it worsens the results.

Best Gaussian Model analyzed:
- Tied Cov (Z-Normalization, no PCA)

4.2. Logistic Regression Classifier

Logistic Regression is a discriminative classification model which does not make any assumption over the distribution of the data but tries to find a hyperplane that maximizes the posterior probability. Starting from the results obtained from the Tied Gaussian classifier we consider the linear decision function obtained from the expression of the posterior log-likelihood ratio:

$$l(x) = \log \frac{P(C = h_1 \mid x)}{P(C = h_0 \mid x)} = \log \frac{f_{X|C}(x \mid h_1)}{f_{X|C}(x \mid h_0)} + \log \frac{\pi}{1-\pi} = \underbrace{w^T x + b}$$

where b considers all the prior information. Given w and b we can compute the expression for the posterior class probability:

$$P(C = h_1 \mid x, w, b) = \frac{e^{(w^T x + b)}}{1 + e^{(w^T x + b)}} = \sigma(w^T x + b)$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid function. LR assumes that the decision rules will be hyperplanes orthogonal to w.

4.2.1. Linear Logistic Regression (LLR)

We are going to look for the minimizer of the function:

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

where λ is an hyperparameter that represents the regularization term (or norm penalty), that has been introduced to make the problem solvable in case of linearly separable classes. The version of LLR used to plot the minDCF for different values of λ is not weighted for simplicity, then after the choice of candidate values of λ the analysis will proceed more in depth with a non-weighted version of LLR and a weighted version with π = 0.5 since it is the target of the main application.

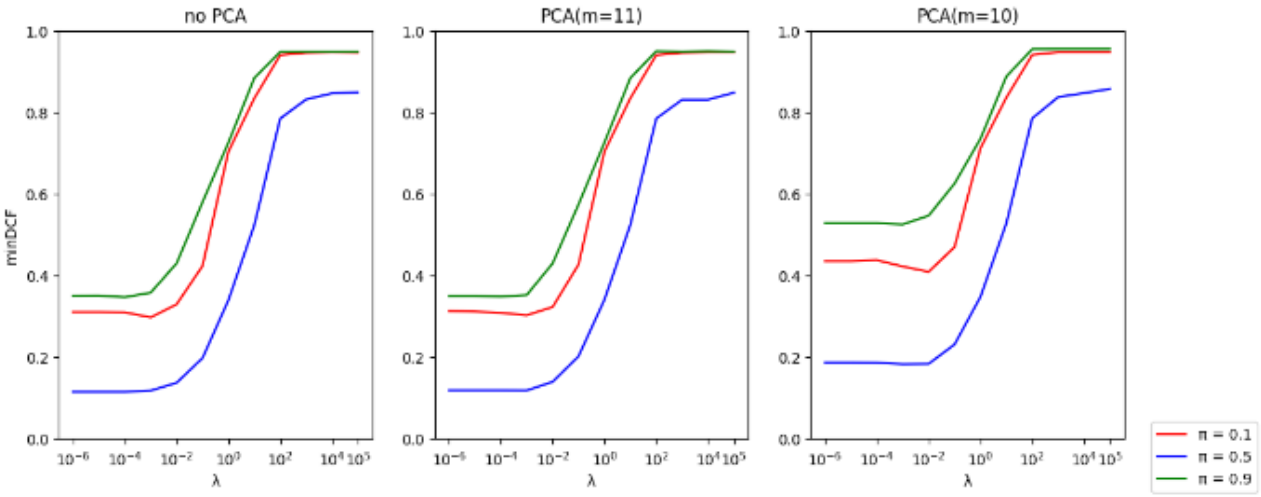

Figure 7: minDCF for different values of λ and different priors

Through the plot in figure 7 is possible to observe the significant influence of the lambda parameter across all configurations, particularly for imbalanced datasets. This observation is consistent across all π˜ values, hence for lambda values exceeding 10^-3, an important increase in the minDCF is noticeable for all applications. There is the need to remember that a bigger value of lambda can lead to excessive generalization, while a smaller value of lambda can lead to overfitting. For this reason, a more detailed table has been provided for values of lambda less than 10^-3 in order to do the best choice (for our training data).

Table 2: *Linear Logistic Regression - 5-fold cross validation*

	No prior			Prior = 0.5		
	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.9$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.9$
	Z-normalized features - no PCA					
LLR ($\lambda = 10^{-3}$)	0.297	0.117	0.358	0.302	0.117	0.356
LLR ($\lambda = 10^{-4}$)	0.310	0.115	0.347	0.285	0.112	0.348
LLR ($\lambda = 10^{-5}$)	0.310	0.115	0.351	0.284	0.112	0.344
LLR ($\lambda = 10^{-6}$)	0.310	0.115	0.350	0.284	0.112	0.341
	Z-normalized features - PCA(m=11)					
LLR ($\lambda = 10^{-3}$)	0.302	0.118	0.352	0.300	0.122	0.354
LLR ($\lambda = 10^{-4}$)	0.308	0.118	0.348	0.305	0.119	0.357
LLR ($\lambda = 10^{-5}$)	0.311	0.118	0.350	0.304	0.120	0.363
LLR ($\lambda = 10^{-6}$)	0.312	0.118	0.350	0.303	0.120	0.364
	Z-normalized features - PCA(m=10)					
LLR ($\lambda = 10^{-3}$)	0.422	0.183	0.528	0.410	0.182	0.534
LLR ($\lambda = 10^{-4}$)	0.437	0.186	0.528	0.418	0.183	0.535
LLR ($\lambda = 10^{-5}$)	0.435	0.186	0.528	0.419	0.183	0.537
LLR ($\lambda = 10^{-6}$)	0.435	0.186	0.528	0.420	0.183	0.537

- The application of PCA has not shown to improve results; in fact, in some cases, it leads to higher minDCF values, especially when considering Z-normalized features with PCA(m=10).
- Seeing the result in the table, it appears clearly that the choice of lambda should be 10^-4 since it provides the best result with smaller values of lambda, but at the same time it reduces the risk of overfitting.
- The prior-weighted version of the LR provides result comparable with the non-prior weighted version, in some cases it provides slight improvements for our main application like it happens for the Z-normalized features without applying PCA.

To conclude, the LLR model's performance is closely tied to the choice of the lambda parameter, the use of PCA does not seem to improve the results, and the presence of a prior doesn't lead to significant performance differences.

Selected LLR Model:
• Z-normalized features, λ = 10^-4, no PCA, prior-weighted version to 0.5

4.2.2. Quadratic Logistic Regression (QLR)

Now we are going to train a Quadratic LR model by performing features expansion. By looking instead at the separation surface obtained through the MVG Gaussian classifier we have:

$$\log \frac{P(C = h_1 \mid x)}{P(C = h_0 \mid x)} = x^T A x + b^T x + c = s(x, A, b, c)$$

This expression is quadratic in x but it's linear in A and b. We could rewrite it to obtain a decision function that is linear for the expanded features space but quadratic in the original features space. Features expansion is defined as:

$$\Phi(x) = \begin{bmatrix} \text{vec}\,(xx^T) \\ x \end{bmatrix}, w = \begin{bmatrix} \text{vec}\,(A) \\ b \end{bmatrix}$$

where vec(X) is the operator that stacks the columns of X. In this way the posterior log-likelihood is expressed as:

$$s(x, w, c) = s^T \phi(x) + c$$

We are now going to train the Linear Logistic Regression model using features vectors φ(x).

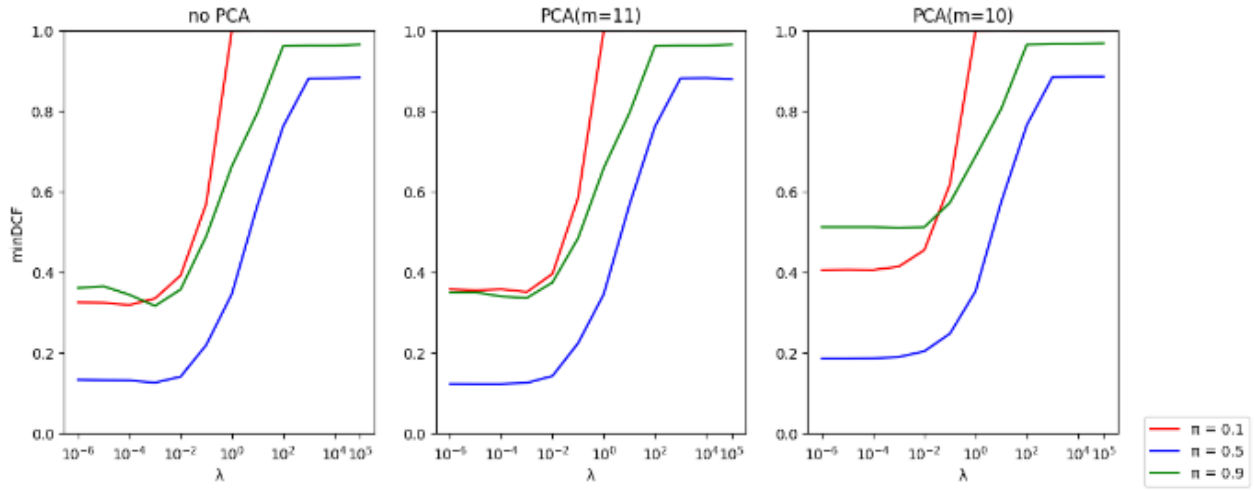

Figure 8: minDCF for different values of λ and different priors

From the analysis of the plot in figure 8, the values of lambda that has been chosen are again the one less than 10^-3 and for this values the analysis would proceed more in depth.

Table 3: *Quadratic Logistic Regression - 5-fold cross validation no Prior / Prior = 0.5*

	No prior			Prior = 0.5		
	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.9$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.9$
Z-normalized features - no PCA						
QLR ($\lambda = 10^{-3}$)	0.335	0.126	0.315	0.335	0.125	0.334
QLR ($\lambda = 10^{-4}$)	0.319	0.133	0.344	0.323	0.133	0.347
QLR ($\lambda = 10^{-5}$)	0.323	0.132	0.361	0.324	0.133	0.358
QLR ($\lambda = 10^{-6}$)	0.330	0.132	0.362	0.325	0.133	0.356
Z-normalized features - PCA(m=11)						
QLR ($\lambda = 10^{-3}$)	0.351	0.124	0.336	0.359	0.124	0.342
QLR ($\lambda = 10^{-4}$)	0.358	0.122	0.340	0.369	0.129	0.344
QLR ($\lambda = 10^{-5}$)	0.357	0.123	0.350	0.363	0.128	0.347
QLR ($\lambda = 10^{-6}$)	0.357	0.124	0.351	0.365	0.128	0.347
Z-normalized features - PCA(m=10)						
QLR ($\lambda = 10^{-3}$)	0.414	0.189	0.510	0.419	0.193	0.515
QLR ($\lambda = 10^{-4}$)	0.405	0.186	0.512	0.420	0.188	0.513
QLR ($\lambda = 10^{-5}$)	0.407	0.185	0.512	0.420	0.188	0.508
QLR ($\lambda = 10^{-6}$)	0.405	0.185	0.512	0.419	0.188	0.508

- Quick analysis of the results reflects that Quadratic Logistic Regression (QLR) outcomes are considerably lower in efficiency compared to their linear counterpart. The QLR version does not perform as well as LLR when no feature expansion is applied, also the application of Principal Component Analysis (PCA) fails to provide significant enhancements to the model performance.
- In the context of the target application, worse outcomes are achieved as with the linear logistic regression version, but the unbalanced applications appear more penalized.
- The incorporation of prior doesn't seem to drastically alter the model's performance, suggesting the model's robustness to different priors.

To conclude, with respect to Gaussian models, comparable performances are achieved for λ = 10^−4 and no PCA considering the LLR model. The quadratic model performs worse than the linear model so it will not be considered.

4.3. SVM Classifier

4.3.1. Linear SVM

Support Vector Machines are linear classifiers that look for maximum margin separation hyperplanes. The primal formulation of the soft-margin SVM problem consists in minimizing the function:

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

where N is the number of training samples and C is an hyperparameter.
We are also going to consider the dual formulation to solve the problem that consists in maximizing the function:

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

where $\mathbf{1}$ is a n-dimensional vector of ones and H is the matrix, whose elements are $H_{ij} = z_i z_j x_i^T x_j$.
From the constraints of the Lagrangian problem that allows to introduce the dual formulation we can obtain that:

$$w^* = \sum_{i=1}^{n} \alpha_i^* z_i x_i$$

and the optimal bias b can be computed considering a sample xi that lies on the margin: $z_i(w^{*T}x_i + b^*) = 1$. To be able to computationally solve the problem (due to L-BFGS with the second constraint of the dual formulation) we need to modify the primal formulation as:

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

where $\hat{x}_i = \begin{bmatrix} x_i \\ 1 \end{bmatrix}$ and $\hat{w} = \begin{bmatrix} w \\ b \end{bmatrix}$.

The scoring rule $\hat{w}^T \hat{x}_i = w^T x + b$ has the same form of the original formulation but we are also regularizing the norm of $\hat{w}$: $\| \hat{w} \|^2 = \| w \|^2 + b^2$ and we use a mapping $\hat{x}_i = \begin{bmatrix} x_i \\ K \end{bmatrix}$ to mitigate the fact that by regularizing the bias term we could obtain sub-optimal results.
There is also the possibility to implement the prior-weigthed version of the model for which the value of C depend on the classes. For a binary case $C_i = \frac{\pi_T}{\pi_T^{emp}}$ or $C_i = \frac{\pi_F}{\pi_F^{emp}}$ depending if the sample is male or female in this case, where π_T^{emp} and π_F^{emp} are the empirical prior evaluated on the training set.

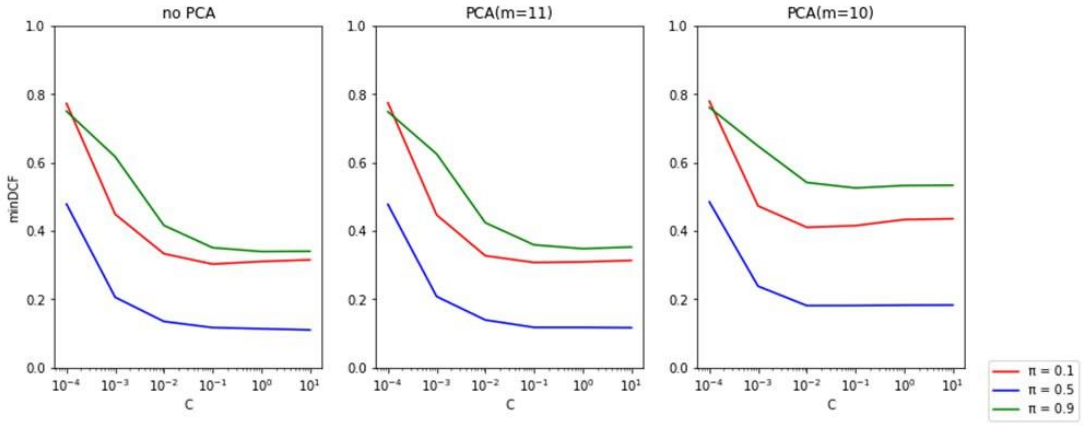

Figure 9: Linear SVM (K=0, Z-Normalized features) – minDCF for different values of C and different prior

The only preprocessing step that has been considered is Z-normalization. As we can notice from figure 9 better results are achieved for higher values of the hyperparameter C so the choice of it is crucial. The results start to decrease from values of C greater than 0.1 and for this reason a more in-depth analysis has been conducted on C=0.1 and C=1.

Table 4: *Linear SVM - 5-fold cross validation*

	No prior			Prior = 0.5		
	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.9$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.9$
Z-normalized features - no PCA						
Linear SVM ($C = 0.1$)	0.289	0.117	0.354	0.291	0.121	0.357
Linear SVM ($C = 1$)	0.298	0.111	0.351	0.322	0.134	0.406
Z-normalized features - PCA(m=11)						
Linear SVM ($C = 0.1$)	0.295	0.126	0.353	0.287	0.127	0.357
Linear SVM ($C = 1$)	0.285	0.117	0.355	0.324	0.137	0.396
Z-normalized features - PCA(m=10)						
Linear SVM ($C = 0.1$)	0.417	0.180	0.550	0.398	0.182	0.548
Linear SVM ($C = 1$)	0.427	0.182	0.540	0.392	0.194	0.574

- Optimum performances on the target application are observed for C = 1 combined with Z-normalized features with no PCA.
- The PCA technique does not contribute significantly to reducing the minDCF in any of the tested applications. Nonetheless, with PCA(m=11), there is no substantial performance drop for the target application.
- Moreover, applying the target prior does not improve performance. In some cases, such as Linear SVM (C = 1) with Z-normalized features and no PCA, the prior can degrade performance.

Selected Linear SVM Model:
Z-normalized features - K=0, C=1 - no PCA, no prior

4.3.2. Kernel SVM

SVMs allow for non-linear classification through the kernel trick. In contrast with Quadratic Logistic Regression classifier, there is no explicit expansion of the features space, it is sufficient to be able to compute the scalar product between the expanded features: $k(x_1, x_2) = \phi(x_1)^T \phi(x_2)$ where k is the kernel function. We must replace $\hat{H}$ with $\hat{H} = z_i z_j k(z_1, x_2)$. We are going to implement two types of kernels: polynomial and rbf.

- Polynomial kernel of degree d:

$$k(x_1, x_2) = (x_1^T x_2 + c)^d$$

- Radial Basis Function kernel:

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

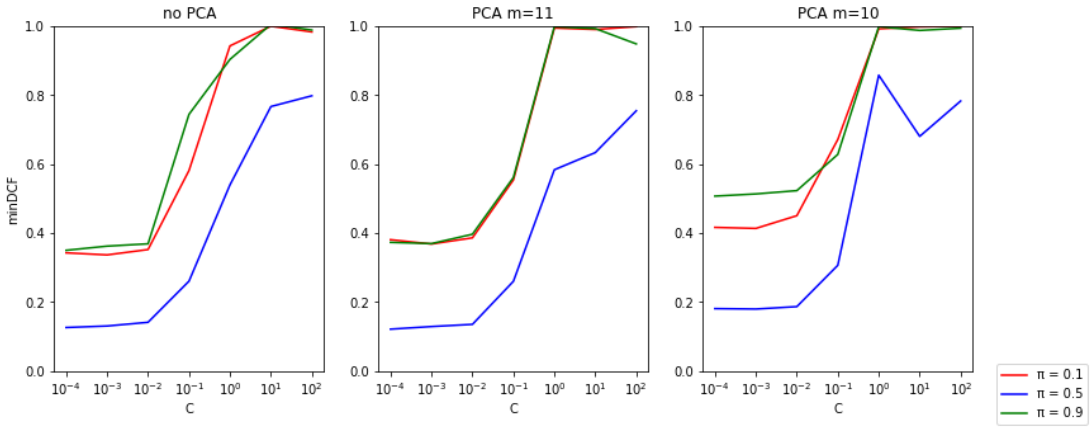

Figure 10: Polynomial SVM (K=1, c=1, d=2, raw features) - minDCF for different values of C

The Polynomial Kernel SVM model was trained with raw features since the Z-Normalization step led to very poor results for almost all the values of the hyperparameter C. By looking at figure 10 we can realize that the choice of C is again crucial and best performances are obtained for $C \leq 10^{-2}$. For this reason, we have chosen $C = 10^{-4}$ to show more detailed results.

Table 5: *Polynomial Kernel SVM – $C = 10^{-4}$ – 5-fold cross validation*

	No prior			Prior = 0.5		
	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.9$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.9$
Raw features – no PCA						
Poly SVM (c=0, d=2)	0.343	0.126	0.350	0.351	0.129	0.371
Poly SVM (c=1, d=2)	0.335	0.127	0.346	0.347	0.128	0.363
Poly SVM (c=1, d=3)	0.487	0.189	0.484	0.488	0.189	0.484
Raw features – PCA(m=11)						
Poly SVM (c=0, d=2)	0.381	0.122	0.373	0.373	0.132	0.375
Poly SVM (c=1, d=2)	0.373	0.123	0.369	0.365	0.132	0.368
Poly SVM (c=1, d=3)	0.588	0.201	0.491	0.576	0.196	0.491
Raw features – PCA(m=10)						
Poly SVM (c=0, d=2)	0.417	0.179	0.506	0.410	0.177	0.483
Poly SVM (c=1, d=2)	0.396	0.181	0.497	0.411	0.175	0.496
Poly SVM (c=1, d=3)	0.593	0.257	0.652	0.526	0.224	0.582

- The Polynomial SVM still manages to perform well, with the model using c = 0 and d = 2 providing the best results (among all Polynomial SVM). In general, the introduction of the prior seems to slightly impact the minDCF, but not in a significant way. Although the performance is not too bad, they are not comparable with previous models that has been chosen as candidates.
- PCA(m=11), like before, slightly improve the results across the board. The impact of the prior here is more pronounced, further decreasing the model's performance.
- As expected, the performance of this model is in line with the performance of the QLR due to quadratic separation rule.

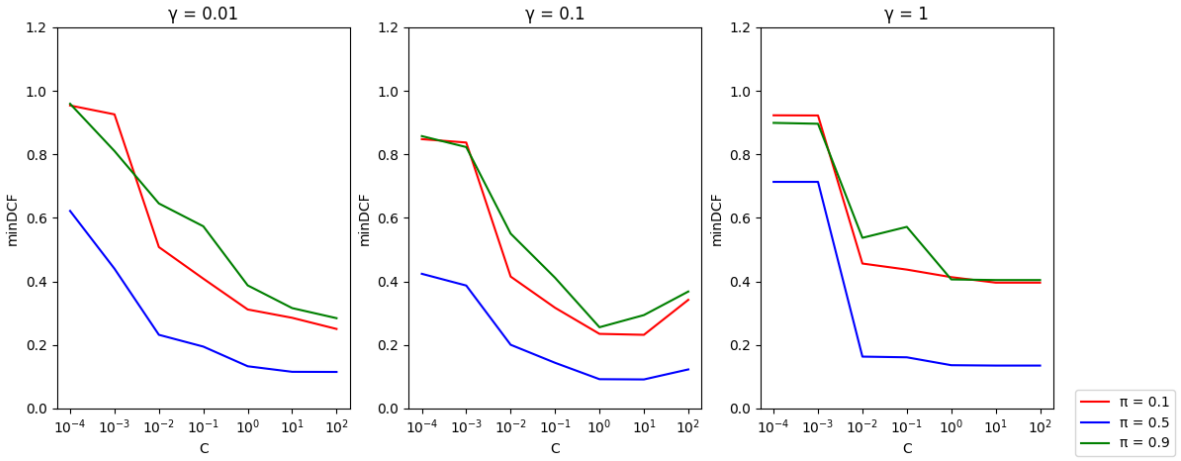

Figure 11: RBF SVM (K = 1, γ ∈{ 0.01,0.1,1}) – minDCF for different values of C

The other kernel that was employed is the RBF kernel and these are the results achieved. In this case it was again used Z-normalization as pre-processing step. The hyperparameter γ must be tuned so we trained the model for different values of γ to analyze the performances and here are showed the results for γ = 0.01, γ = 0.1 and γ = 1. From figure 11 we again realize that the value of the hyperparameter C is critical and a value C ≥1 has to be chosen.

Table 6: *RBF Kernel SVM - C=10*

	No prior			Prior = 0.5		
	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.9$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.9$
Z-Normalized features - no PCA						
RBF SVM (γ = 1)	0.396	0.135	0.405	0.396	0.135	0.405
RBF SVM (γ = 0.1)	0.232	0.092	0.294	0.246	0.093	0.287
RBF SVM (γ = 0.01)	0.286	0.116	0.316	0.274	0.114	0.335
Z-Normalized features - PCA(m=11)						
RBF SVM (γ = 1)	0.414	0.134	0.416	0.414	0.134	0.416
RBF SVM (γ = 0.1)	0.261	0.095	0.277	0.273	0.095	0.289
RBF SVM (γ = 0.01)	0.297	0.114	0.324	0.280	0.114	0.319
Z-Normalized features - PCA(m=10)						
RBF SVM (γ = 1)	0.446	0.163	0.547	0.446	0.163	0.547
RBF SVM (γ = 0.1)	0.355	0.126	0.342	0.361	0.121	0.377
RBF SVM (γ = 0.01)	0.401	0.184	0.500	0.401	0.180	0.489

- RBF SVM performs well for the target application when PCA is not applied. The best minDCF for the target application was achieved with gamma = 0.1 and C=10.
- The use of PCA does not significantly improve the minDCF values, which confirms the previous conclusion about the limited effectiveness of PCA in these cases.
- Interestingly, the introduction of the prior seems to slightly alter the minDCF, with most of the changes being a little bit worse than before.

Selected Kernel SVM Models:
- RBF SVM: Z-Normalized features, K = 1, C = 10, γ = 0.1, no PCA, no prior

4.4. GMM Classifier

The last model we are going to consider is a generative model. The GMM problem is related to the estimation of a population distribution and can be also applied to the classification task. We resort the class-posterior probability used for the Gaussian classifiers:

$$f_X(x) = \sum_{c=1}^{K} f_{X|C}(x \mid c) \cdot P(C = c) = \sum_{c=1}^{K} \mathcal{N}(x \mid \mu_c, \Sigma_c) \cdot \pi_c$$

More in general we can introduce a weight term instead of the prior probability that will be one of the model's parameters of the GMM:

$$X \sim \text{GMM}\,(M, \mathcal{S}, w) \Rightarrow f_X(x) = \sum_{g=1}^{M} w_g \cdot \mathcal{N}\big(x \mid \mu_g, \Sigma_g\big)$$

so, the GMM density is the sum of M Gaussians where $M = [\mu_1, \dots, \mu_M], \mathcal{S} = [\Sigma_1, \dots, \Sigma_M], w = [w_1, \dots, w_M]$ are the model parameters. Gaussian components can be seen as clusters the samples belong to (in a hard or in a soft way) and the cluster label is an unobserved latent random variable. If we define

$$f_{X_i,G_i}(x_i, g) = w_g \mathcal{N}\big(x_i \mid \mu_g, \Sigma_g\big) \text{ and}$$
$$f_{X_i}(x_i) = \sum_{g=1}^{M} f_{X_i,G_i}(x_i, g)$$

we can introduce a term called responsibility that represents the posterior probability that a sample

belongs to a certain cluster:

$$\gamma_{g,i} = P(G_i = g \mid X_i = x) = \frac{f_{X_i,G_i}(x_i, g)}{f_{X_i}(x_i)} = \frac{w_g \mathcal{N}(x_i \mid \mu_g, \Sigma_g)}{\sum_{g'} w_{g'} \mathcal{N}(x_i \mid \mu_{g'}, \Sigma_{g'})}$$

Then assign the sample to the cluster label for which the responsibility is maximum and then re-estimate the model parameters given the cluster assignments. An evident problem is the creation of hard clusters not admitting that a sample could belong to more than one cluster. To handle soft-clusters:

$$\boldsymbol{\mu}_c = \frac{\sum_i \gamma_{c,i} \mathbf{x}_i}{\sum_i \gamma_{c,i}} = \frac{\mathbf{F}_c}{N_c} \qquad \boldsymbol{\Sigma}_c = \frac{1}{N_c} \sum_i \gamma_{c,i} \mathbf{x}_i \mathbf{x}_i^T - \boldsymbol{\mu}_c \boldsymbol{\mu}_c^T = \frac{\mathbf{S}_c}{N_c} - \boldsymbol{\mu}_c \boldsymbol{\mu}_c^T \qquad w_c = \frac{N_c}{N} = \frac{\sum_i \gamma_{c,i}}{\sum_{c=1}^{K} \sum_i \gamma_{c,i}}$$

Where Nc, Fc and Sc are respectively zero-order, first-order and second-order statistics.
In this way we will be able to apply the Expectation-Maximization algorithm:

- Expectation step: estimate the responsibility (given the model parameters (M_t,S_t ,w_t))
- Maximization step: estimate the new model parameters by using the statistics mentioned above.

The estimation goes on starting from an initial value of the model parameters until a certain criterion is met. The EM algorithm thus require an initial estimate for the GMM parameters, so we employ the LBG algorithm to incrementally construct a GMM with 2G components from a GMM with G components. The starting point will be (1, μ, C) so we use the empirical mean and covariance matrix of the dataset.Then it builds a 2-componenents model starting from one and from each of the new components other new 2 components are generated and so on and so forth.
GMM has also its Diagonal and Tied version: the diagonal variant consists of a model where each component has a diagonal covariance matrix (this does not correspond with the Naive Bayes assumption of the Gaussian models). The Tied covariance model assumes that the covariance of a single GMM $\Sigma_{g,c} = \Sigma_c$ are the same but each GMM of each class has a difference covariance matrix Σ_c (again, this is different from the Tied MVG model). Now, given the fact that almost all the features are already well distributed according to the Gaussian hypothesis probably there is no need of many gaussian components for the model. Moreover, since the data are divided into three different age group particular attention should be put on GMM model with more than 2 components. Tied GMM model is supposed to outperform the other two models as already seen in the Gaussian classifiers.

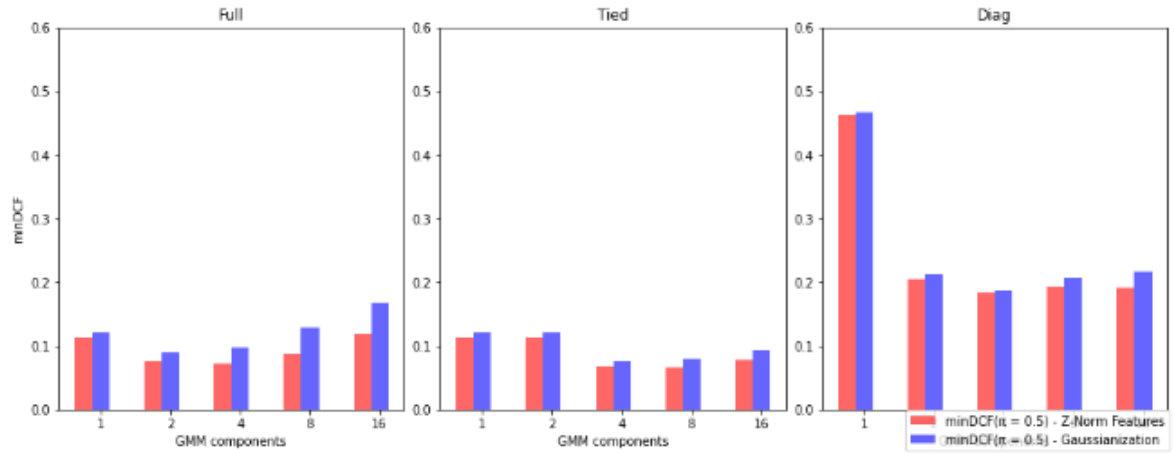

Figure 12: GMM - minDCF for different number of components

- Examining the performance with a reduced number of components (4), it's apparent that all model types deliver substantial results. The diagonal covariance assumption showing comparatively lower efficiency when fewer components are used, with a small increase for a greater number of components.
- Of the models analyzed, the Tied model emerges as the best performer, closely followed by the Full model. This performance pattern aligns with the initial hypothesis based on our observations from Gaussian models.
- Regarding the preprocessing steps, the Gaussianization process doesn't seem to aid in improving model results. As expected, this suggests that the initial data distribution was already optimized for this classification task, making the Gaussianization step redundant.

Now it will follow a more in depth analysis for the Full and Tied GMM model with 4 components since they are the best performer.

Table 7: *GMM - 5-fold cross validation*

	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.9$
Z-normalized features - no PCA			
GMM Full (4 comp.)	0.201	0.071	0.205
GMM Tied (4 comp.)	0.237	0.067	0.222
GMM Diag (4 comp.)	0.462	0.183	0.474
Z-normalized features - PCA(m=11)			
GMM Full (4 comp.)	0.206	0.069	0.194
GMM Tied (4 comp.)	0.202	0.070	0.229
GMM Diag (4 comp.)	0.371	0.147	0.375
Z-normalized features - PCA(m=10)			
GMM Full (4 comp.)	0.268	0.111	0.276
GMM Tied (4 comp.)	0.315	0.100	0.316
GMM Diag (4 comp.)	0.465	0.199	0.491
Gaussianized features - no PCA			
GMM Full (4 comp.)	0.283	0.098	0.252
GMM Tied (4 comp.)	0.208	0.077	0.236
GMM Diag (4 comp.)	0.500	0.187	0.513
Gaussianized features - PCA(m=11)			
GMM Full (4 comp.)	0.290	0.091	0.239
GMM Tied (4 comp.)	0.182	0.074	0.223
GMM Diag (4 comp.)	0.386	0.159	0.387
Gaussianized features - PCA(m=10)			
GMM Full (4 comp.)	0.373	0.134	0.312
GMM Tied (4 comp.)	0.259	0.106	0.298
GMM Diag (4 comp.)	0.476	0.198	0.481

- Principal Component Analysis (PCA), much like Gaussianization, does not substantially improve the results. PCA with m=11 has results comparable to performance without PCA, while PCA with m=10 worsens a lot the performance. It is noteworthy that Gaussianization and PCA(m=11) produced a minor improvement for the ~pi = 0.5 application.

The GMM Tied (4 components) trained with Z-Normalization and no PCA is the one that achieves the best results among all the models evaluated up to now.

Selected GMM Model:

- GMM Tied (4 components), Z-Normalization, no PCA.

Table 8: *Best models analyzed up to now.*

	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.9$
Gaussian Models			
Tied Cov (Z-Norm, no PCA)	0.299	0.109	0.342
Logistic Regression Models			
LRR (Z-Norm, $\lambda = 10^{-4}$, no PCA, prior = 0.5)	0.310	0.115	0.347
SVM Models			
Linear SVM ($K = 0$, $C = 1$, no prior)	0.298	0.111	0.351
RBF SVM (Z-Norm, $\gamma = 0.1$, no PCA, no prior)	0.232	0.092	0.294

5. Score Calibration

5.1. Calibration Analysis on Selected Models

These are the selected candidates to be the best model and for which the analysis about the calibration of the score is provided.

- Gaussian model: Tied-Cov (Z-Normalization, no PCA)
- Logistic Regression: Linear LR (Z-Normalization, λ = 10^−4, no PCA, prior = 0.5)
- SVM: Linear SVM (K = 0, C = 1, no prior)
- SVM: RBF SVM (Z-Normalization, no PCA, no prior)
- GMM: Tied GMM with 4 components (Z-normalization, no PCA)

5.2. Calibrating Scores for Selected Models

The metric used to select the best model was the minDCF, however the evaluation of the the cost that is paid on a possible decision was based on theoretical threshold $t = -\log \frac{\tilde{\pi}}{1-\tilde{\pi}}$, but there is no guarantee that this threshold is still the optimal one and for this reason an analysis on the score is performed.

We assume that the function f that maps the uncalibrated score to calibrated has the form:

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

and f (s) can be interpreted as log-likelihood ratio for the two classes hypotheses:

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

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

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

By interpreting scores s as samples of a dataset (where each sample has 1 feature) a prior weighted logistic regression model can be employed to learn the model parameters over our calibration set. Letting:

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

a Logistic Regression model is obtained where α and β′ are the model parameters to be learnt. A prior $\tilde{\pi}$ needs to be specified and will be the one of the target application, but there will be improvements in terms of calibration even on the unbalanced applications. To obtain calibrated scores we will have to compute:

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

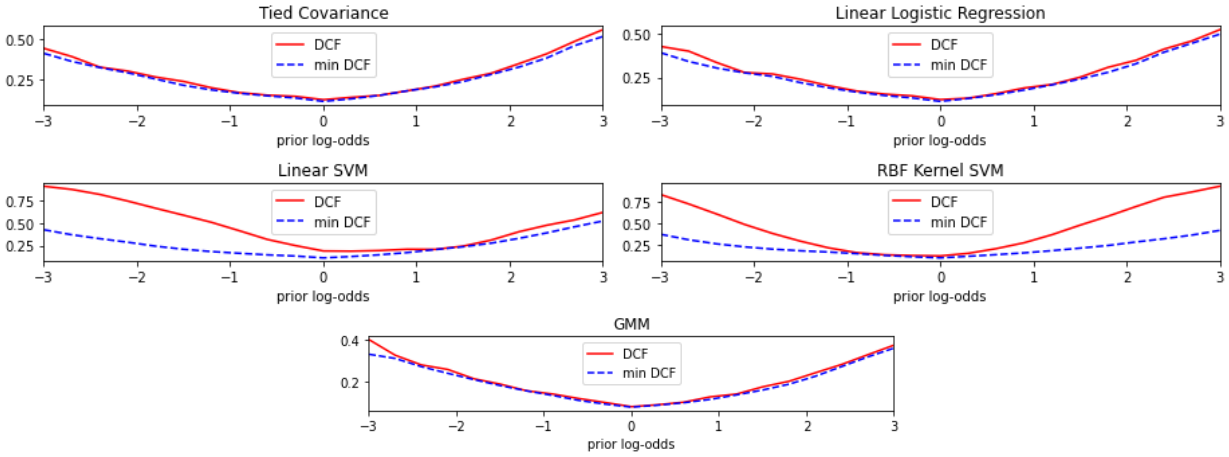

Figure 13: Bayesian Error Plots for each of the selected models (Without score calibration)

As the Figure 13 shows the Tied Gaussian, Linear Logistic Regression and Tied GMM models are already well calibrated for all values of the prior log-odds. The SVM models instead are not well calibrated for almost every considered value of the prior log-odds.

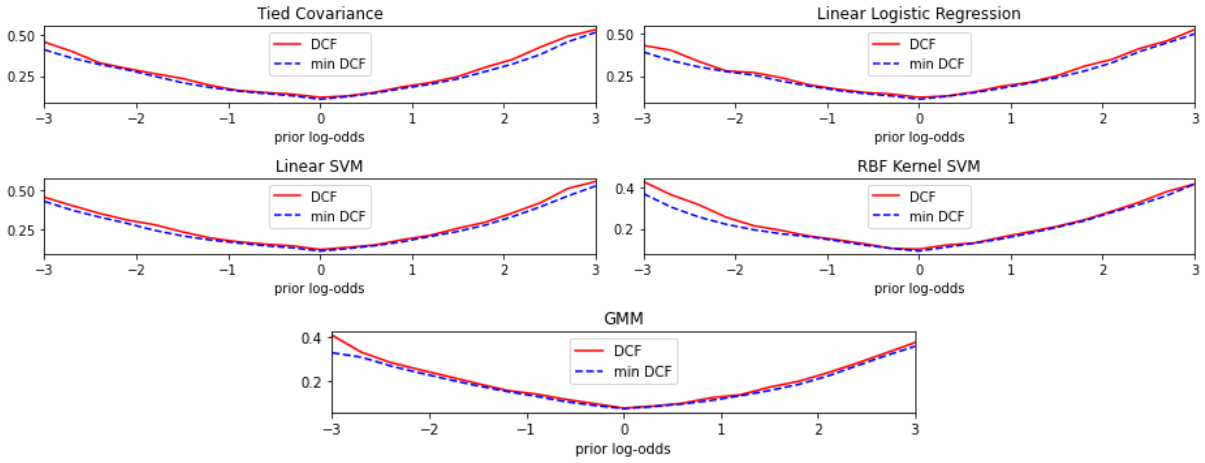

Figure 14: Bayesian Error Plots for each of the selected models (With score calibration)

Table 9: *Calibrated vs not calibrated scores for best selected Models*

	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.9$
Uncalibrated scores [minDCF / DCF]			
Tied Gaussian	0.299/0.311	0.109/0.119	0.342/0.371
Linear LR	0.285/0.299	0.112/0.122	0.348/0.365
Linear SVM	0.298/ 0.770	0.111/ 0.188	0.351/ 0.435
RBF SVM	0.232/0.517	0.092/0.115	0.294/0.742
Tied GMM	0.237/0.249	0.067/0.071	0.222/0.244
Calibrated scores [minDCF / DCF]			
Tied Gaussian	0.299/0.306	0.109/0.121	0.342/0.377
Linear LR	0.285/0.301	0.112/0.124	0.348/0.364
Linear SVM	0.298/0.317	0.111/0.121	0.351/0.368
RBF SVM	0.232/0.276	0.092/0.101	0.294/0.301
Tied GMM	0.237/0.242	0.067/0.073	0.222/0.231

We can easily notice that for models where scores where already well calibrated the score calibration didn't provide effective benefits, in some cases it lead to slightly worse results. For SVM model great benefits are obtained for both the unbalanced applications and target application.

6. Experimental Results

Now the best performing model will be trained on the entire training set and the evaluation set will be used for the first time to assess if the results obtained so far are consistent and the best models selected are confirmed. Then, after the evaluation of the top-performing models, an analysis will be done also for models with a different set of hyperparameters.

Table 10: *minDCF over evaluation set for all models trained over the whole training set (Z-Normalized features, no PCA)*

	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.9$
Gaussian Models (Z-normalized features, no PCA)			
Tied Cov	0.301	0.116	0.308
LR Models (Z-Normalized features, no PCA)			
LLR ($\lambda = 10^{-4}$, prior = 0.5)	0.300	0.120	0.306
SVM Models (Z-Normalized features, no PCA)			
Linear SVM ($K = 0, C = 1$, no prior)	0.308	0.117	0.310
RBF SVM ($K = 1, C = 10, \gamma = 0.01$, no prior)	0.275	0.090	0.234
GMM Models (Z-Normalized features, no PCA)			
GMM Tied (4 comp.)	0.178	0.058	0.160

Table 11: *minDCF over evaluation set for all models trained over the whole training set (Z-Normalized features, PCA(m=11))*

	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.9$
Gaussian Models (Z-normalized features, PCA(m=11))			
Tied Cov	0.314	0.124	0.313
LR Models (Z-Normalized features, PCA(m=11))			
LLR ($\lambda = 10^{-4}$, prior = 0.5)	0.316	0.124	0.310
SVM Models (Z-Normalized features, PCA(m=11))			
Linear SVM ($K = 0, C = 1$, no prior)	0.310	0.124	0.313
RBF SVM ($K = 1, C = 10, \gamma = 0.01$, no prior)	0.279	0.090	0.218
GMM Models (Z-Normalized features, PCA(m=11))			
GMM Tied (4 comp.)	0.148	0.058	0.155

The results obtained by training the models on the entire training set and by evaluating them on the entire evaluation set are coherent with the results previously obtained and with the observations made. Precisely, the evaluation has been made also using PCA with m=11 to see if there were any changes, since during the training phase different models were improved by this pre-processing technique. In this case, the results obtained are in line with the expectations with GMM Tied as best perfoming model overall that improved a little bit on the unbalanced applications.

The ROC curve in figure 15 shows and confirms that the best performing model is the Tied GMM with 4 components, immediately followed by RBF SVM and then the other three models also perform well but slightly worse than these two.

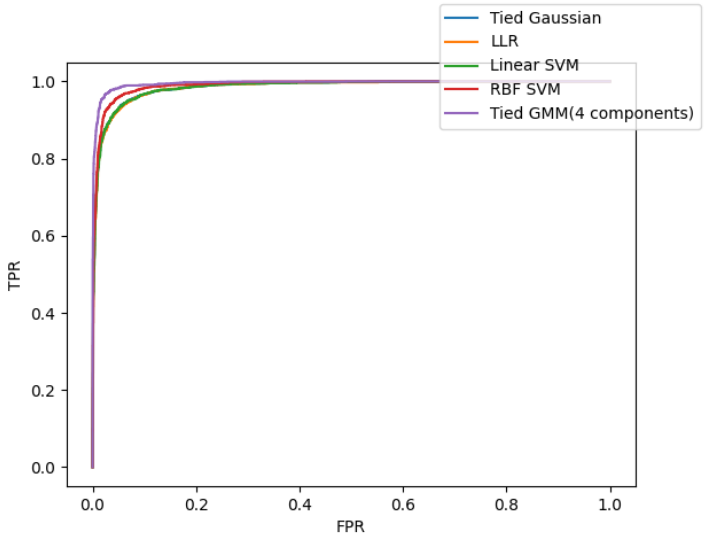

Figure 15: ROC curves of selected models

6.1. Calibration on evaluation score

After the evaluation of the test set, here it follows a check on the score calibration to see if there was a mis-calibration on the score.

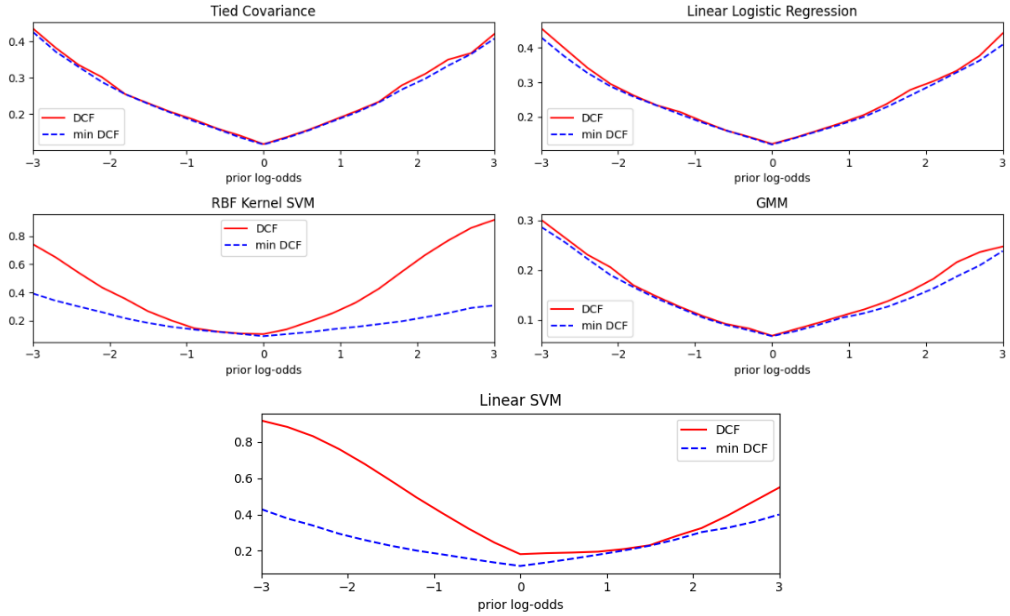

Figure 16: Bayesian Error Plots for each of the selected models on test set (Without score calibration)

As expected from the calibration made during the training phase, a good level of calibration is achieved by the probabilistic model. The two models belonging to the SVM family do not perform well for a large number of applications. Precisely, the linear SVM is not calibrated for negative values of the prior log-odds, while the RBF SVM is not calibrated for some positive values too. To conclude, SVM models could benefit of the calibration so the analysis of the calibrated score will follow as well (for all the models):

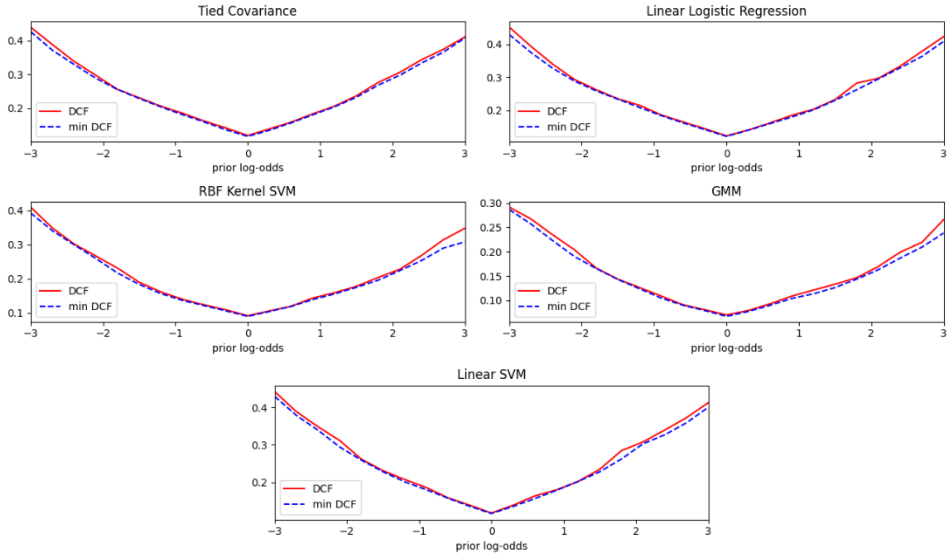

Figure 17: Bayesian Error Plots for each of the selected models on test set (With score calibration)

As the figure 17 shows also the models belonging to the SVM family now obtain a good level of calibration for all kind of applications. A table is provided to see a comparison between the non-calibrated/calibrated scores.

Table 12: *Calibrated vs not calibrated scores for models on evaluation*

	$\tilde{\pi}$ = 0.1	$\tilde{\pi}$ = 0.5	$\tilde{\pi}$ = 0.9
Uncalibrated scores [minDCF / DCF]			
Tied Gaussian	0.301/0.309	0.116/0.118	0.308/0.323
Linear LR	0.300/0.316	0.120/0.122	0.306/0.320
Linear SVM	0.308/0.789	0.117/0.182	0.310/0.340
RBF SVM	0.275/0.472	0.090/0.104	0.234/0.702
Tied GMM	0.178/0.182	0.058/0.059	0.160/0.182
Calibrated scores [minDCF / DCF]			
Tied Gaussian	0.301/0.310	0.116/0.119	0.308/0.314
Linear LR	0.300/0.313	0.120/0.122	0.306/0.313
Linear SVM	0.308/0.315	0.117/0.118	0.310/0.315
RBF SVM	0.275/0.281	0.090/0.091	0.234/0.238
Tied GMM	0.178/0.189	0.058/0.059	0.160/0.183

From table 12 it is possible to notice that where scores were already well calibrated the score calibration didn't provide effective benefits and in some cases the calibration also obtained slightly worse results as it happens for Tied Gaussian and Tied GMM model. It is instead a totally different situation for SVM models that had a great decrease for actual DCF values for unbalanced application.

6.2. Considerations

After seeing the performance of the model with the chosen hyperparameters, the analysis will proceed more in depth comparing the result of the same model with other set of parameters including also other 2 model that were discarded like Full gaussian model and the Full GMM model (that provided good results that were slightly worse than the tied model and that were discarded to choose just one model for each family). These results are computed on Z-normalized feature.

Table 13: *Results on other analyzed models with different parameters*

	$\tilde{\pi}$ = 0.1	$\tilde{\pi}$ = 0.5	$\tilde{\pi}$ = 0.9
Gaussian Models (Z-normalized features, no PCA)			
Full Cov	0.312	0.120	0.312
LR Models (Z-Normalized features, no PCA)			
LLR ($\lambda = 10^{-4}$, no prior)	0.310	0.120	0.320
LLR ($\lambda = 10^{-4}$, prior=0.1)	0.294	0.122	0.338
LLR ($\lambda = 10^{-4}$, prior=0.9)	0.333	0.123	0.276
SVM Models (Z-Normalized features, no PCA)			
Linear SVM ($K = 0$, $C = 0.1$, no prior)	0.296	0.116	0.321
Linear SVM ($K = 0$, $C = 1$, no prior)	0.308	0.117	0.310
RBF SVM ($K = 1$, $C = 10$, $\gamma = 0.001$, no prior)	0.321	0.124	0.312
RBF SVM ($K = 1$, $C = 10$, $\gamma = 1$, no prior)	0.445	0.129	0.400
GMM Models (Z-Normalized features, no PCA)			
Full GMM (4 comp.)	0.152	0.060	0.157
Full GMM (8 comp.)	0.205	0.075	0.185
Tied GMM (8 comp.)	0.184	0.064	0.163

As it is possible to extract from table 13, the results for the model with other set of hyperparameters are slightly worse than the results obtained with best model. An honorable mention must be done for the Full GMM model with 4 components that behaves in the same way of the Tied Model reflecting the similarity between the covariance matrix of the two analyzed classes.

7. Conclusions

We can conclude by saying that in general linear models have better performances on these data with respect to the non-linear ones. Due to a moderate level of correlation among features the PCA mostly didn't help in achieving better results, although with PCA(m=11) in some cases better performance were provided while with PCA(m=10) there was always a downgrade in performance. Naive hypothesis didn't work well in Gaussian models due to the reasons explained before. A great result was reached with a minDCF ≈ 0.058 for the target application ($\tilde{\pi} = 0.5$) but also for the unbalanced applications we were able to achieve good results (minDCF ≈ 0.148 for $\tilde{\pi} = 0.1$ and minDCF ≈ 0.155 for $\tilde{\pi} = 0.9$). The choices made on the training set via k-fold cross-validation proved to be coherent with the results obtained on the entire training set and proved to be successful when applied to the evaluation set.