Report Machine Learning : Gender Identification

Authors:
Dalia Vincenzo s309864
Todaro Mario s308812

February 2024

Contents

Abstract

This report aims to examine a dataset comprising diverse low-level images of genders through the utilization of various Machine Learning (ML) algorithms. Initial focus involves analyzing the dataset structure to understand its distribution, followed by an analysis of multiple classifiers. The goal is to develop a system capable of optimal sample classification while minimizing costs. The project's significance lies in its potential applications, such as gender-dependent face recognition models.

1 Dataset

The dataset used consists of low-dimensional representations of images obtained by mapping face images to a common low-dimensional variety.

The samples, which have 12 features, belong to 3 different age groups, each characterized by different distributions for the embeddings, and were divided into two separate files :

Train.txt contains a total of 2400 samples (720 males and 1680 females) and is the part of the dataset that will be used for training various models, while Test.txt contains 6000 samples (4200 males and 1800 females) and will be used for testing and evaluation.

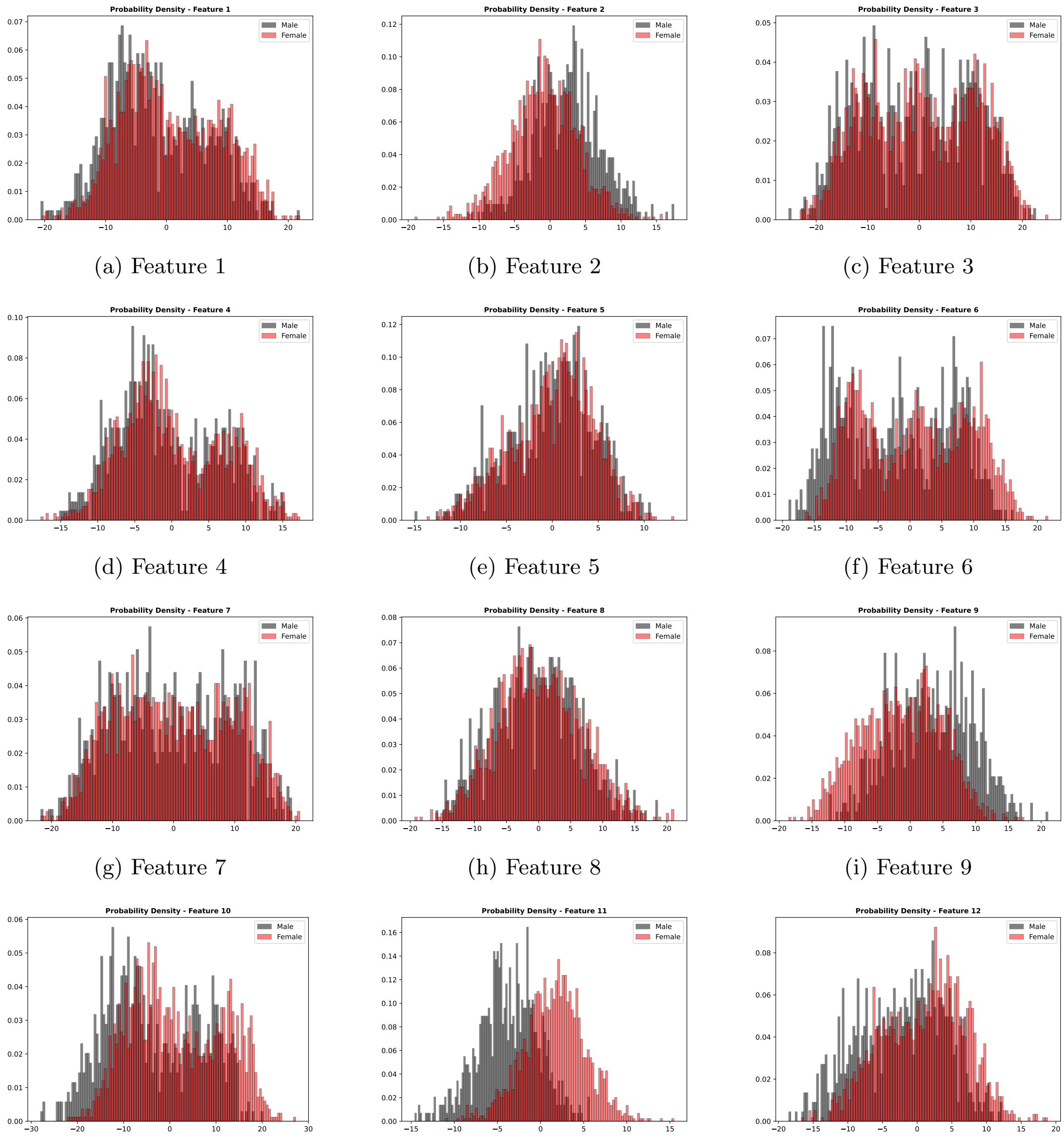
It is important to point out that the two datasets are very unbalanced, in fact while as far as the training set is concerned there is a female majority(labeled as 1), the situation is completely reversed in the test set, where male samples (labeled as 0) represent 70% of the total.

The working point is therefore defined by the triplet ($\pi_T = 0.5$,$C_{fn} = 1$, $C_{fp} = 1$), where π_T is the application prior, C_{fn} is the cost of false negative errors and C_{fp} instead is the cost of false positive errors. However, we will also analyze how the model behaves for different working points.

2 Feature Analysis

2.1 Histograms

In the following graphs, histograms on the features of the dataset have been plotted in order to deepen the analysis of the data and have a greater understanding on the individual variables (it should be specified that the data were first centered).



(a) Feature 1 (b) Feature 2 (c) Feature 3

(d) Feature 4 (e) Feature 5 (f) Feature 6

(g) Feature 7 (h) Feature 8 (i) Feature 9

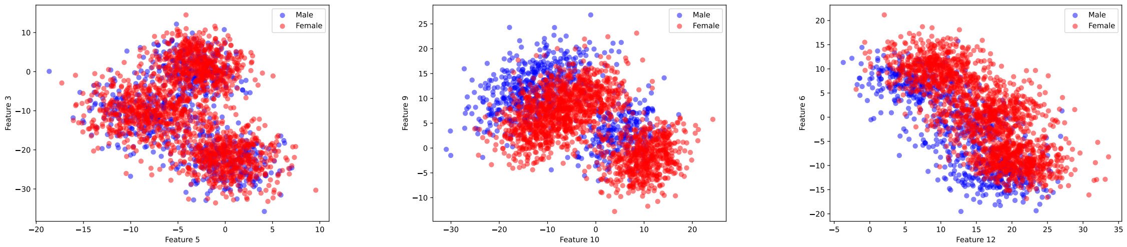
(j) Feature 10 (k) Feature 11 (l) Feature 12

Figure 1: Probability Density for each Feature
Black : Male - Red : Female

By analyzing the histograms it is possible for us to identify which features will be better to use for classification, selecting the graphs in which the distributions of the two classes are most distinct, such as the one for feature 11. While in features 2 and 9 the separation is present although very slight, for all the others the overlap is sharp, so they will not discriminate well between the two classes. Furthermore, it is evident how the histograms of features 3 and 6 resemble the pattern of 3 Gaussians, due to the division into 3 different age groups affecting the distribution

2.2 Scatter

Now consider the following scatter plots, which relate two different features on the abscissa and ordinate



(a) Feature 4 - Feature 2

(b) Feature 9 - Feature 8

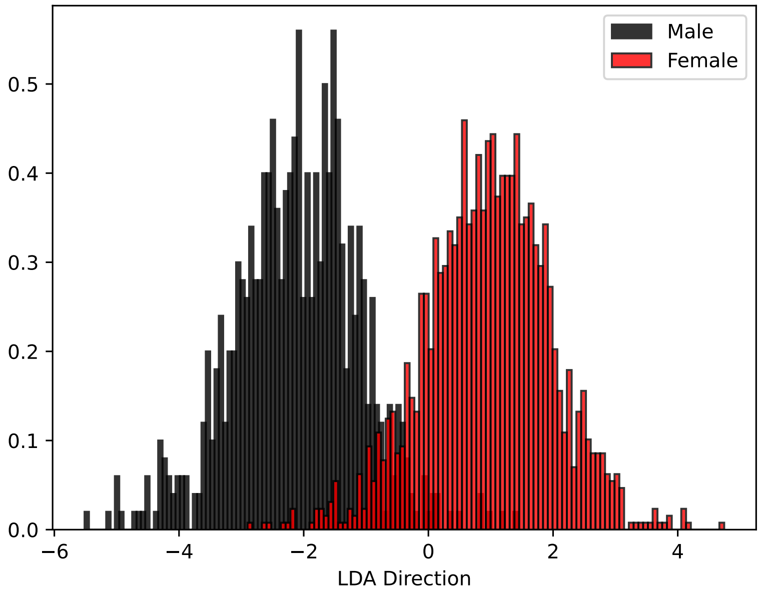
(c) Feature 11 - Feature 5

Figure 2: Representative Scatter Plots
Blue : Male - Red : Female

The analysis of some representative scatter plots further confirms the above considerations: the 3 different clusters are easily identifiable and it is evident how feature 9 manages to separate the classes better. Also, noting that the distributions resemble a Gaussian, it is possible to assume that Gaussian models will be particularly effective.

2.3 LDA

Linear Discriminant Analysis (LDA) is a statistical method used for dimensionality reduction and classification. Its primary goal is to find the linear combinations of features that best separate two or more classes in a dataset, which is a useful operation in our case.

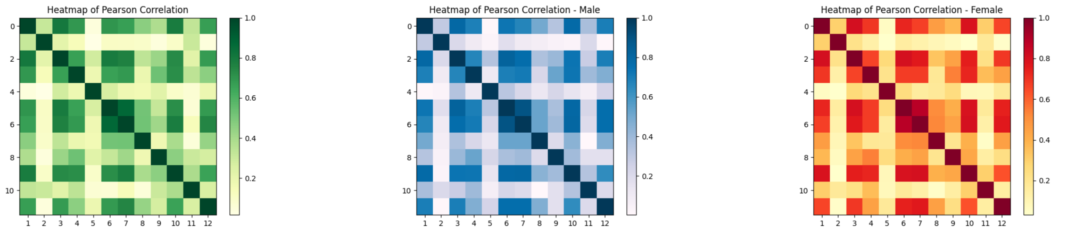


(a) Linear Discriminant Analysis
Black : Male - Red : Female

Applying the LDA allowed us to clearly separate the two classes, although there is still an overlap that, although minimal, can cause classification errors.

2.4 Pearson Correlation

Now we use Pearson correlation to evaluate the linear relationships between different features (variables) in a dataset. This can be helpful for tasks such as feature selection, identifying correlations, and gaining insights into the interdependence of variables.



(a) Heatmap of the dataset

(b) Heatmap for Males

(c) Heatmap for Females

Figure 4: Pearson Correlation Heatmaps

The heatmap of the entire dataset shows that some features are highly correlated, such as couples 6-7 and 1-9, while others like pairs 2-4 or 1-5, are not related to each other. It should be noticed that, in general, the features are on average related to each other, with an oscillation that varies between 0.4 and 0.6. Analyzing heatmaps for individual labels, the interdependence of the features remains evident, suggesting the usefulness of resorting to features selection methods to manage redundancy and keep only the most informative features.

2.5 PCA

Now we analyze the results of the Principal Component Analysis (PCA), a technique used to capture and retain the most significant information in the dataset while reducing its dimensionality. In particular, explained variance is crucial in this context because helps us understand how much information each principal component carries from the original dataset.

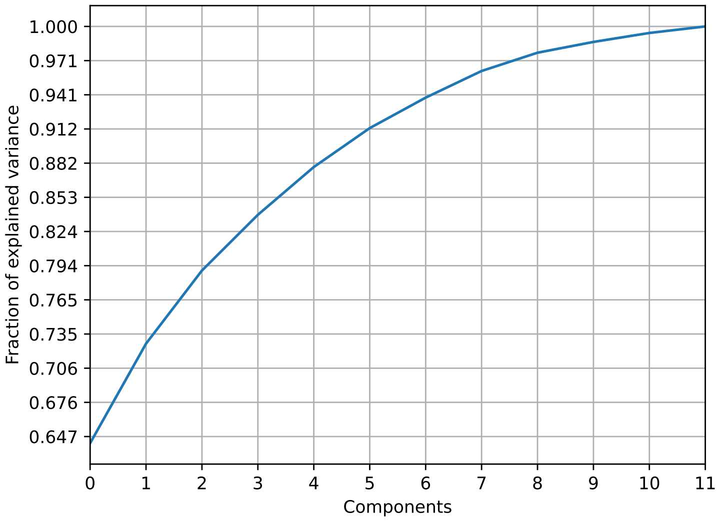


Figure 5: Explained Variance

As we can see from the plot, we can apply the PCA reducing up to 8 features while still maintaining over 97% of variance. Hence, it might be valuable to attempt the mapping of data from 12 dimensions to 8 dimensions. This exploration is motivated by the observation that by eliminating 4 dimensions, we can retain 97% of the data variance, thereby preserving a significant portion of the essential information. Moreover, the application of PCA (Principal Component Analysis) not only maintains the integrity of the data but also reduces the number of parameters required for classification estimation.

3 Model Training

For the training of the models we decided to proceed with the K-Fold Cross Validation in order to have more data available for both training and validation. In particular, we implemented a K-Fold approach with K=5, which is a good trade-off. In fact, given the dataset's relatively small size, we could have opted for a Leave-One-Out approach as well, which is a specialized case of K-Fold with K=N, where N represents the number of samples. However, implementing this approach necessitates training numerous models, resulting in a time-consuming and computationally expensive process. We will consider three different applications: a uniform prior application ($\pi_T = 0.5$, $C_{fn} = 1$, $C_{fp} = 1$) and two unbalanced applications where the prior is biased towards one of the two classes: ($\pi_T = 0.1$, $C_{fn} = 1$, $C_{fp} = 1$) and ($\pi_T = 0.9$, $C_{fn} = 1$, $C_{fp} = 1$), also using different pre-processing techniques such as PCA and ZNorm. For evaluating our models, we will utilize the normalized minimum detection cost function (minDCF) as the performance metric. This metric quantifies the cost we would pay if we possessed the optimal threshold in advance for the evaluation set, specifically referring to our validation set in this context.

3.1 Gaussian Classifier

We are going to focus the attention on Multivariate Gaussian Classifiers (MVG), in particular for those with the following covariance matrices: Full Covariances, Tied Covariance, Diagonal Covariances.In particular, all these models assume that we can use a Gaussian distribution to describe our data, given the class:

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

Given that histograms indicate an approximate Gaussian distribution for the features, it is anticipated that Generative Models will perform well on this dataset. Additionally, heatmaps reveal a considerable dispersion of correlation among the features. Hence, it is expected that models relying on Naïve Bayes assumptions will exhibit poor performance.

3.1.1 MVG

The following results relate to the MVG model both by considering the RAW model and by applying PCA.

	$\pi_\mathbf{T} = \mathbf{0.5}$	$\pi_\mathbf{T} = \mathbf{0.1}$	$\pi_\mathbf{T} = \mathbf{0.9}$
MVG NO PCA	0.117	0.303	0.353
MVG + PCA (m=11)	0.124	0.294	0.335
MVG + PCA (m=10)	0.167	0.410	0.488
MVG + PCA (m=9)	0.188	0.408	0.560
MVG + PCA (m=8)	0.195	0.470	0.563
MVG + PCA (m=7)	0.272	0.652	0.713

Table 1: MVG Results with and without PCA

As we expected MVG gets good results. The PCA has been applied in various configurations (m variable from 11 to 7, with m indicating the number of selected features) while for m=11 similar results are obtained to the NO PCA model, further decreasing the number of features the performance is of lower quality. This happens because by removing features we are also reducing the amount of information available to the algorithm.

3.1.2 Tied MVG

In the Tied Gaussian Classifier we assume that the two classes have the same covariance matrix:

$$(X|C = c) \sim \mathcal{N}(\mu_c, \Sigma) \tag{2}$$

So, each class has its own mean μ_c, but the covariance matrix Σ is the same for all classes.

	$\pi_{\mathbf{T}} = \mathbf{0.5}$	$\pi_{\mathbf{T}} = \mathbf{0.1}$	$\pi_{\mathbf{T}} = \mathbf{0.9}$
Tied NO PCA	0.116	0.286	0.337
Tied + PCA (m=11)	0.120	0.290	0.356
Tied + PCA (m=10)	0.164	0.384	0.475
Tied + PCA (m=9)	0.183	0.386	0.550
Tied + PCA (m=8)	0.189	0.418	0.561
Tied + PCA (m=7)	0.268	0.638	0.700

Table 2: Tied Gaussian Results with and without PCA

As we can see from the results, MVG and the Tied Gaussian get very similar results. This happens because MVG assumes that the two classes have very distinct covariance matrices, while as we have seen from the analysis of the Heatmap, in the dataset examined the correlation between the various features is very similar for both classes.

3.1.3 Naïve Bayes Gaussian Classifier

The Naive Bayes Classifier assumes the statistic independence of the variables of a class, in order to simplify the model. As a result the covariance matrices will all be diagonal.

	$\pi_{\mathbf{T}} = \mathbf{0.5}$	$\pi_{\mathbf{T}} = \mathbf{0.1}$	$\pi_{\mathbf{T}} = \mathbf{0.9}$
NB NO PCA	0.466	0.782	0.772
NB + PCA (m=11)	0.137	0.299	0.356
NB + PCA (m=10)	0.173	0.434	0.465
NB + PCA (m=9)	0.195	0.475	0.544
NB + PCA (m=8)	0.196	0.480	0.548
NB + PCA (m=7)	0.282	0.642	0.679

Table 3: Naïve Bayes Gaussian Results with and without PCA

Applying this model without preprocessing results much worse than MVG, due to the correlation between some of the features present. Nevertheless, by applying the PCA the results improve significantly, while still being worse than those of the MVG model or Tied MVG.

3.1.4 Tied Naive Gaussian Classifier

We can merge the Naive Bayes assumption with the tied covariance constraint to create the Naive Tied Gaussian model. This model involves a singular diagonal covariance matrix.

	$\pi_{\mathbf{T}} = \mathbf{0.5}$	$\pi_{\mathbf{T}} = \mathbf{0.1}$	$\pi_{\mathbf{T}} = \mathbf{0.9}$
TiedNB NO PCA	0.461	0.778	0.780
TiedNB + PCA (m=11)	0.128	0.292	0.367
TiedNB + PCA (m=10)	0.174	0.423	0.467
TiedNB + PCA (m=9)	0.187	0.450	0.542
TiedNB + PCA (m=8)	0.194	0.462	0.557
TiedNB + PCA (m=7)	0.271	0.643	0.694

Table 4: Tied Naïve Bayes Gaussian Results with and without PCA

3.2 Logistic Regression

Logistic Regression, which is a discriminative approach for classification, unlike generative models, relies on assumptions about the separation rule rather than the distribution of the data. In this context, we will explore both linear and quadratic logistic regression, employing a version of the model that incorporates prior weights. Our objective function will be:

$$J(w,b) = \frac{\lambda}{2}\|w\|^2 + \frac{1}{2}\sum_{i=1|c_i=1}^{n} \frac{\pi_T}{n_T} \log\left(1+e^{-z_i(w^T x_i+b)}\right) + \frac{1}{2}\sum_{i=1|c_i=0}^{n} \frac{(1-\pi_T)}{n_F} \log\left(1+e^{-z_i(w^T x_i+b)}\right) \tag{3}$$

The symbol λ represents the regularization coefficient, serving as a hyperparameter crucial for optimizing the classifier's performance. It is essential to carefully select λ to strike a balance in the regularization term, promoting simpler solutions and mitigating the risk of overfitting. The regularization term encourages a model that generalizes well to unseen data. If λ is excessively small, the solution may exhibit effective separation on the training set but might perform poorly on new data. Conversely, if λ is too large, the solution may struggle to adequately distinguish between classes. The proper choice of λ is pivotal for achieving a well-balanced and effective classifier.
It is also possible to define non-linear separation rules by building a certain expanded feature space using, instead of x, feature vectors $\phi(x)$, defined as:

$$\phi(x) = \begin{bmatrix} \mathbf{x}\mathbf{x}^T \\ x \end{bmatrix} \tag{4}$$

It allows computing linear separation rules for $\phi(x)$, and this corresponds to estimate quadratic separation surfaces in the original space.

3.2.1 Linear Logistic Regression

First of all we make a comparison between various configurations of the model to identify the best value for the hyperparameter λ.

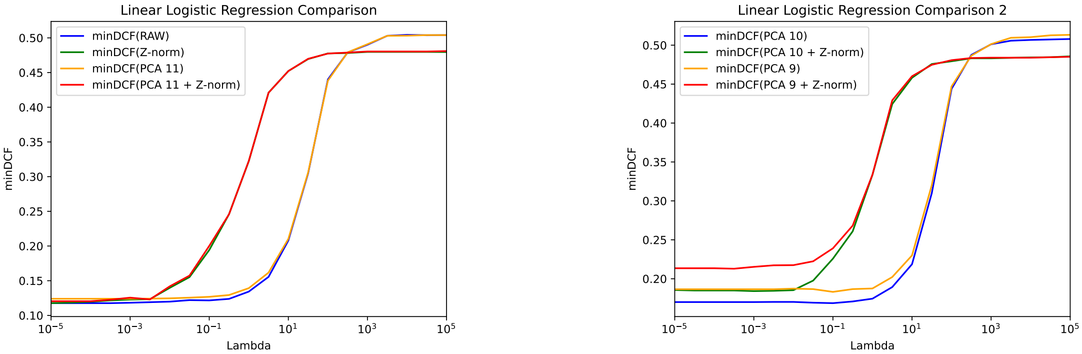
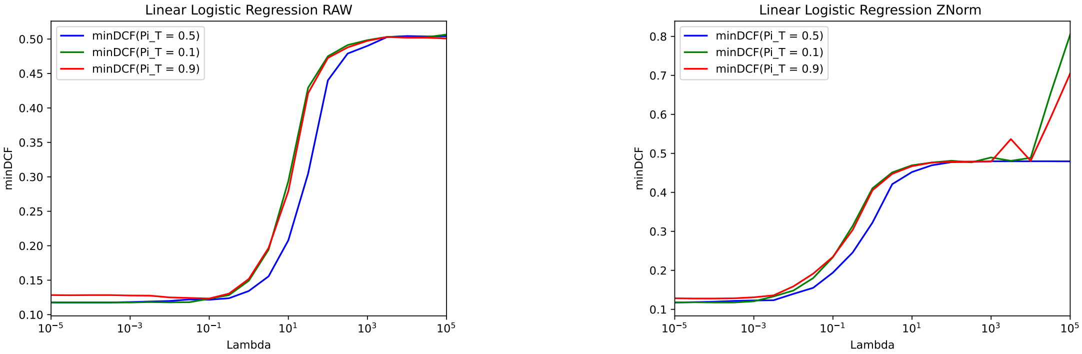


Figure 6: Comparison between different values of λ in different configurations

This comparison graph shows that the two models that, for different lambda values, get better values in the calculation of the minDCF are the RAW model and the one on which we apply the ZNorm as pre-processing. So, let's deepen the study of these models to choose as accurately as possible the value of lambda, also investigating for different values of π_T.



(a) RAW Logistic Regression

(b) ZNorm Logistic Regression

Figure 7: Raw Logistic Regression and ZNorm Logistic Regression with different Π_T

By analyzing the graphs, the best lambda values to select are 0.01 and 0.0001 for the RAW model and ZNorm model respectively. Choosing this value helps to alleviate the risk of overfitting the model to the training data and enhances its capacity to generalize effectively to novel, unseen data. Here are the results obtained using the selected value of lambda considering also different values of π_T.

$\mathbf{RAW}, \lambda = \mathbf{0.01}$	$\pi = \mathbf{0.5}$	$\pi = \mathbf{0.1}$	$\pi = \mathbf{0.9}$
$\pi_{\mathbf{T}} = \mathbf{0.5}$	0.120	0.282	0.334
$\pi_{\mathbf{T}} = \mathbf{0.1}$	0.118	0.311	0.330
$\pi_{\mathbf{T}} = \mathbf{0.9}$	0.125	0.304	0.368

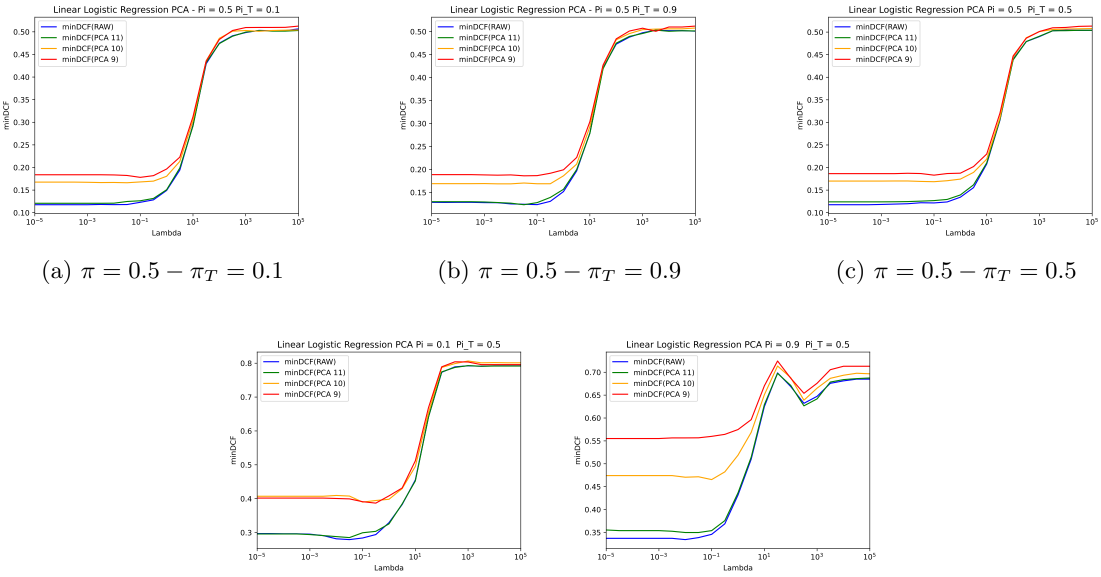
Table 5: RAW Logistic Regression with different combinations of π e π_T

$\mathbf{ZNorm}, \lambda = \mathbf{0.0001}$	$\pi = \mathbf{0.5}$	$\pi = \mathbf{0.1}$	$\pi = \mathbf{0.9}$
$\pi_{\mathbf{T}} = \mathbf{0.5}$	0.120	0.286	0.337
$\pi_{\mathbf{T}} = \mathbf{0.1}$	0.118	0.308	0.324
$\pi_{\mathbf{T}} = \mathbf{0.9}$	0.128	0.304	0.368

Table 6: ZNorm Logistic Regression with different combinations of π e π_T

As we can see the results of the two models are very similar, and in particular the best score is identical. As the Z-Score is a linear transformation, the performances achieved on both raw and Z-Scored data remain unchanged. Modifying the values of π_T does not significantly enhance the model. While there are slight improvements when attempting $\pi_t = 0.9$, this is attributed to the dataset's imbalance towards class 1 (female class).

For completeness, the model graphs are shown below applying the PCA as a pre-processing with different combinations of π and π_T.



(a) $\pi = 0.5 - \pi_T = 0.1$ (b) $\pi = 0.5 - \pi_T = 0.9$ (c) $\pi = 0.5 - \pi_T = 0.5$

(d) $\pi = 0.1 - \pi_T = 0.5$ (e) $\pi = 0.9 - \pi_T = 0.5$

Figure 8: PCA Linear Logistic Regression with different combinations of π and π_T

3.2.2 Quadratic Logistic Regression

As with Linear Logistic Regression, we compare different model configurations and preprocessing methods to find the best λ value for Quadratic Logistic Regression as well.

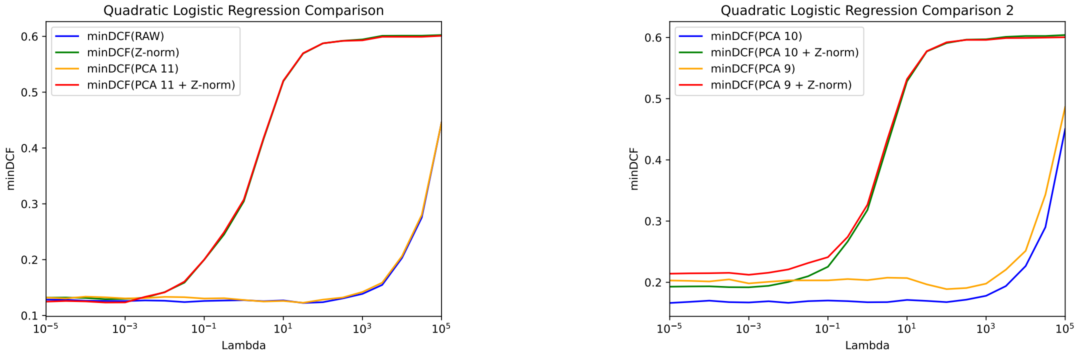


Figure 9: Comparison between different values of λ in different configurations

Also in this case the best results are obtained with the RAW model and the pre-processed model with ZNorm and the selected values of λ are 100 and 0 respectively. Now let's look at the results.

$\mathbf{RAW}, \lambda = \mathbf{100}$	$\pi = \mathbf{0.5}$	$\pi = \mathbf{0.1}$	$\pi = \mathbf{0.9}$
$\pi_\mathbf{T} = \mathbf{0.5}$	0.124	0.288	0.352
$\pi_\mathbf{T} = \mathbf{0.1}$	0.130	0.326	0.323
$\pi_\mathbf{T} = \mathbf{0.9}$	0.128	0.298	0.410

Table 7: RAW Quadratic Logistic Regression with different combinations of π e π_T

$\mathbf{ZNorm}, \lambda = \mathbf{0}$	$\pi = \mathbf{0.5}$	$\pi = \mathbf{0.1}$	$\pi = \mathbf{0.9}$
$\pi_\mathbf{T} = \mathbf{0.5}$	0.133	0.364	0.332
$\pi_\mathbf{T} = \mathbf{0.1}$	0.136	0.331	0.362
$\pi_\mathbf{T} = \mathbf{0.9}$	0.141	0.406	0.382

Table 8: ZNorm Quadratic Logistic Regression with different combinations of π e π_T

In contrast to the linear case, the outcomes for Raw features and Z-Scored differ in the non-linear scenario. This discrepancy arises due to a feature expansion operation that computes the dot product within an alternative embedding space. As a result, the model exhibits sensitivity to data transformations. Moreover, the quadratic model performs worse since the class distribution does not fit in quadratic models.

3.3 SVM

Let's shift our focus to Support Vector Machines (SVMs), a non-probabilistic model that generates scores without a probabilistic interpretation. Unlike Logistic Regression, which aims to maximize class probabilities, SVMs choose the hyperplane that maximizes the margin − the distance from the selected hyperplane to the closest point separating the classes. We consider a balanced version by multiplying the hyperparameter C by two factors :

$$C_i = C\frac{\pi_T}{\pi_{\text{T emp}}} \qquad C_i = C\frac{\pi_F}{\pi_{\text{F emp}}} \tag{5}$$

In this model, we explore two distinct formulations:

- The **Primal Formulation** is utilized when dealing with linearly separable classes and is applied in the context of the linear Support Vector Machine (SVM) model. The associated objective function that needs to be minimized is:

$$\hat{J}(\hat{\mathbf{w}}) = \frac{1}{2}\|\hat{\mathbf{w}}\|^2 + C\sum_{i=1}^{n} \max(0, 1 - z_i \cdot (\hat{\mathbf{w}}^\mathbf{T} \cdot \hat{\mathbf{x}}_\mathbf{i})) \tag{6}$$

  where:

$$\hat{x}_i = \begin{bmatrix} \mathbf{x_i} \\ K \end{bmatrix} \qquad \hat{\omega} = \begin{bmatrix} \omega \\ b \end{bmatrix} \tag{7}$$

  We center our attention on an essential hyperparameters: C, which plays a crucial role in balancing the trade-off between maximizing the margin and minimizing training error. In fact, when the value of C is low, the model excels on the training set but tends to be overly specific. On the other hand, a high value of C ensures generalization, yet it may lead to non-optimal values of min_{DCF}.

- As we have done for the Quadratic Logistic Regression, also for SVM it is possible to **expand the feature space** in order to obtain non-linear decision functions. Achieving this is facilitated by the utilization of kernel functions, which permit the computation of dot products between matrices in the expanded space without requiring an actual transformation of the dataset.

3.3.1 Linear SVM

First we perform the hyperparameter tuning C considering K=1 and several pre-processing operations applied to the database.

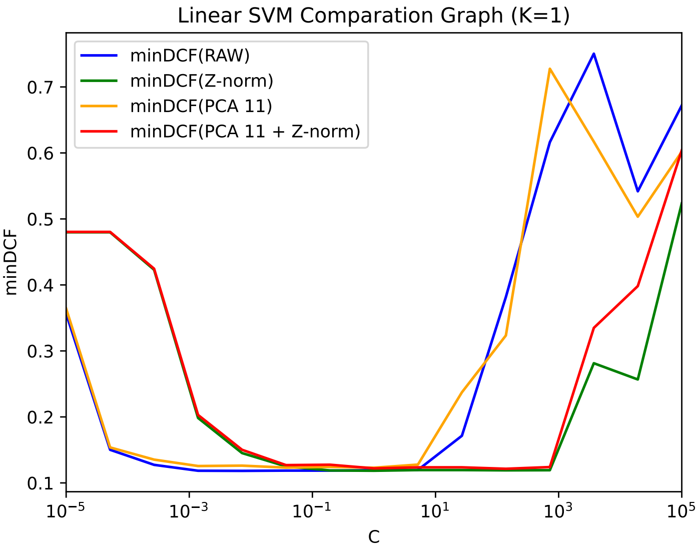


Figure 10: Comparison between different SVM configurations

As the graph shows, the best configurations seem to be the one without pre-processing and the one with ZNorm. Now we tune C.

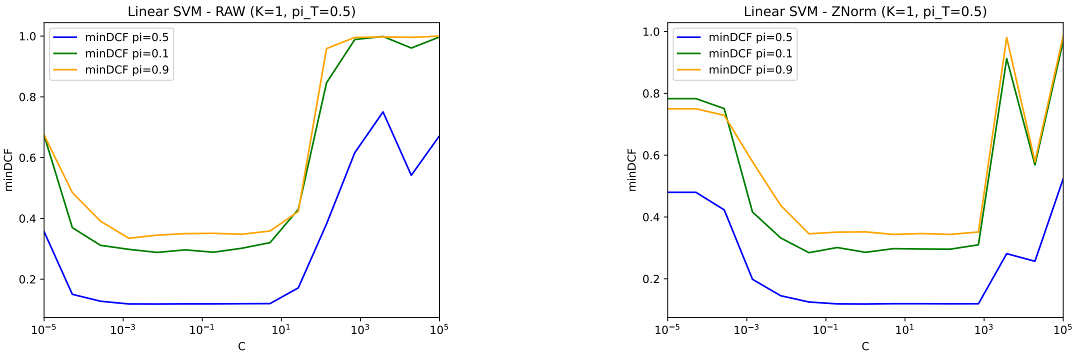


Figure 11: Tuning of C in different Linear SVM configurations

As we can see, they get good results with C=10. We then deepen the study by visualizing the results.

RAW, C=10	$\pi = \mathbf{0.5}$	$\pi = \mathbf{0.1}$	$\pi = \mathbf{0.9}$
$\pi_{\mathbf{T}} = \mathbf{0.5}$	0.122	0.312	0.356
$\pi_{\mathbf{T}} = \mathbf{0.1}$	0.125	0.301	0.370
$\pi_{\mathbf{T}} = \mathbf{0.9}$	0.134	0.327	0.381

Table 9: Linear SVM without pre-processing, C=10 and with different combinations of π e π_T

ZNorm, C=10	$\pi = \mathbf{0.5}$	$\pi = \mathbf{0.1}$	$\pi = \mathbf{0.9}$
$\pi_{\mathbf{T}} = \mathbf{0.5}$	0.119	0.295	0.346
$\pi_{\mathbf{T}} = \mathbf{0.1}$	0.121	0.311	0.321
$\pi_{\mathbf{T}} = \mathbf{0.9}$	0.134	0.299	0.370

Table 10: Linear ZNorm SVM, C=10 and with different combinations of π e π_T

So, with ZNorm, performance gets a boost (because data is centered) but it's not that big compared to Linear SVM without pre-processing.

3.3.2 Polynomial SVM

For this model that uses dual formulation, we have the following kernel function:

$$k(x_1, x_2) = (x_1^T x_2 + c)^d \tag{8}$$

Here, c and d are two hyperparameters signifying the polynomial coefficient and its degree, respectively. We will utilize a polynomial SVM with d=2 and c=1 , essentially creating a quadratic SVM. Next, we aim to develop C considering the two best configurations previously identified. The selected value, as suggested by the following graphs, will be C=0.001 for the RAW model and C=10 for the other.

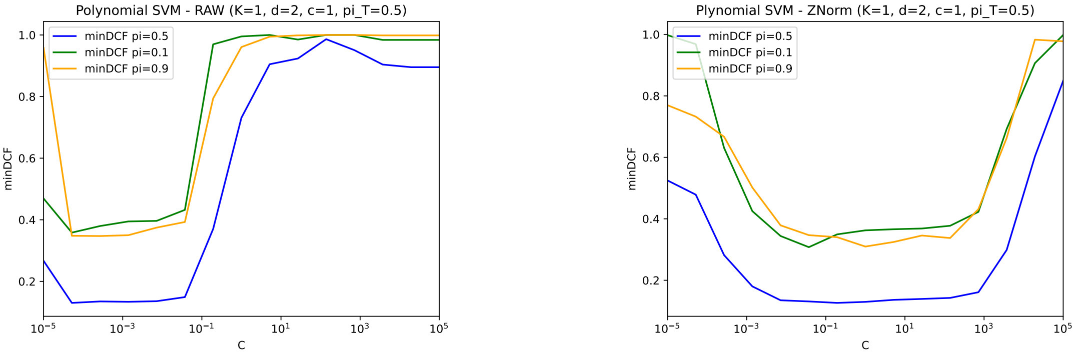


Figure 12: Tuning of C in different Polynomial SVM configurations

RAW, C=0.001	$\pi = \mathbf{0.5}$	$\pi = \mathbf{0.1}$	$\pi = \mathbf{0.9}$
$\pi_\mathbf{T} = \mathbf{0.5}$	0.135	0.406	0.347
$\pi_\mathbf{T} = \mathbf{0.1}$	0.144	0.392	0.384
$\pi_\mathbf{T} = \mathbf{0.9}$	0.154	0.402	0.439

Table 11: Polynomial SVM without pre-processing, C=10 and with different combinations of π e π_T

ZNorm, C=10	$\pi = \mathbf{0.5}$	$\pi = \mathbf{0.1}$	$\pi = \mathbf{0.9}$
$\pi_\mathbf{T} = \mathbf{0.5}$	0.138	0.37	0.337
$\pi_\mathbf{T} = \mathbf{0.1}$	0.15	0.32	0.383
$\pi_\mathbf{T} = \mathbf{0.9}$	0.153	0.46	0.405

Table 12: ZNorm Polynomial SVM, C=10 and with different combinations of π e π_T

3.3.3 Radial Basis Function SVM

We now consider another SVM model that uses the same dual formulation, but with different kernel function:

$$k(x_1, x_2) = e^{-\gamma \|x_1 - x_2\|^2} \tag{9}$$

The parameter γ plays a crucial role in determining the width of the kernel. A smaller γ leads to a broader kernel, causing a support vector to influence a larger number of points. On the other hand, a larger γ results in a narrower kernel, where a support vector has a more limited impact on points that are farther away. Therefore, our goal is to optimize the value of γ through tuning.

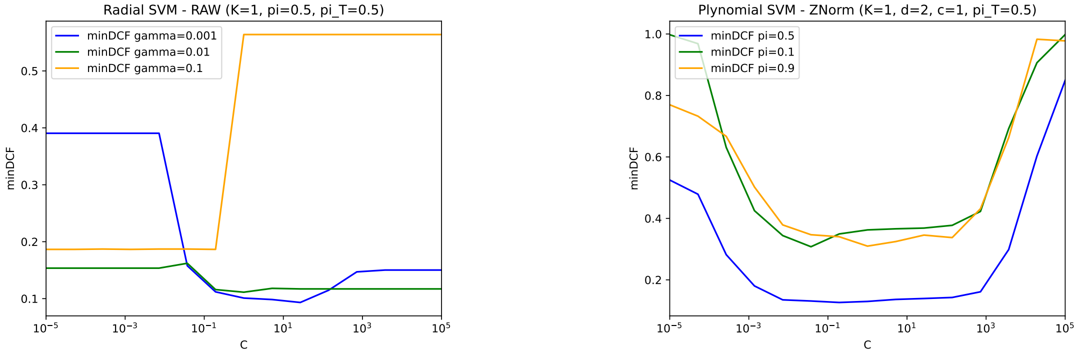


Figure 13: Tuning of λ and C in different Radial SVM configurations

After graph analysis, we have chosen $\lambda = 0.001$ and C=5 for the RAW model and $\lambda = 0.1$ and C=10 for the one with ZNorm.

RAW, C=5	$\pi = \mathbf{0.5}$	$\pi = \mathbf{0.1}$	$\pi = \mathbf{0.9}$
$\pi_\mathbf{T} = \mathbf{0.5}$	0.095	0.255	0.263
$\pi_\mathbf{T} = \mathbf{0.1}$	0.100	0.327	0.235
$\pi_\mathbf{T} = \mathbf{0.9}$	0.118	0.293	0.351

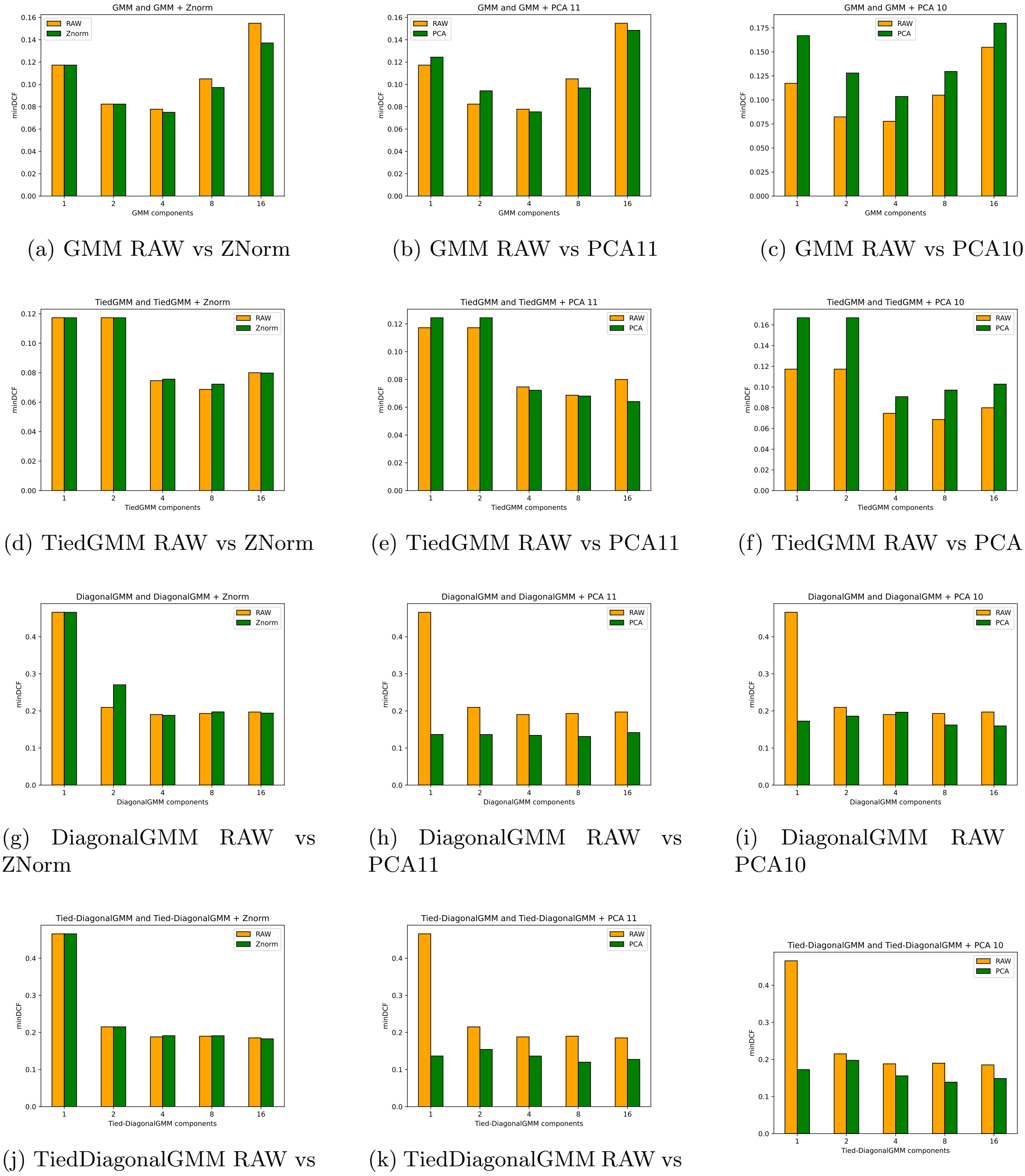
Table 13: Radial Basis SVM without pre-processing, C=10 and with different combinations of π e π_T

ZNorm, C=10	$\pi = \mathbf{0.5}$	$\pi = \mathbf{0.1}$	$\pi = \mathbf{0.9}$
$\pi_\mathbf{T} = \mathbf{0.5}$	0.100	0.274	0.276
$\pi_\mathbf{T} = \mathbf{0.1}$	0.110	0.325	0.262
$\pi_\mathbf{T} = \mathbf{0.9}$	0.111	0.298	0.332

Table 14: ZNorm Radial Basis SVM, C=10 and with different combinations of π e π_T

3.4 GMM

GMM assumes that data distributions can be modeled effectively using multiple Gaussian distributions. This assumption is of significant importance to us, as our analysis of the dataset suggests that some features adhere to this structure. As explained above, the dataset includes data from three different age groups. Anticipating the results of the Gaussian Model, we expect the GMM Standard and Tied GMM to show favorable performance, while Diagonal GMM and Tied GMM are likely to produce suboptimal results due to previous considerations regarding the correlations of characteristics in the dataset.



(a) GMM RAW vs ZNorm

(b) GMM RAW vs PCA11

(c) GMM RAW vs PCA10

(d) TiedGMM RAW vs ZNorm

(e) TiedGMM RAW vs PCA11

(f) TiedGMM RAW vs PCA10

(g) DiagonalGMM RAW vs ZNorm

(h) DiagonalGMM RAW vs PCA11

(i) DiagonalGMM RAW vs PCA10

(j) TiedDiagonalGMM RAW vs ZNorm

(k) TiedDiagonalGMM RAW vs PCA11

(l) GMM RAW vs PCA10

Figure 14: comparison of different GMM models with different configurations

As expected, analyzing the charts the best models are the GMM and the Tied GMM, which are able to have significantly better performance than the DiagonalGMM and TiedDiagonalGMM. Therefore we deepen the study of these models with various configurations that include eventual pre-processing with PCA or ZNorm.

Model / Components	$\pi = 0.5$	$\pi = 0.1$	$\pi = 0.9$
GMM / 2 Comp	0.082	0.257	0.189
GMM + ZNorm / 2 Comp	0.082	0.257	0.189
GMM / 4 Comp	0.078	0.238	0.203
GMM + ZNorm / 4 Comp	0.075	0.237	0.200
GMM (pca 11) / 4 Comp	0.074	0.227	0.198
GMM (pca 10) / 4 Comp	0.102	0.301	0.234
GMM (pca 9) / 4 Comp	0.106	0.325	0.269
Tied GMM / 8 Comp	0.069	0.251	0.179
Tied GMM + ZNorm / 8 Comp	0.072	0.256	0.196
Tied GMM (pca 11)/ 8 Comp	0.077	0.264	0.179

Table 15: GMM results for different configurations

As we can see, the Gaussian Mixture models give us good results. The TiedGMM with 8 components is the best model and it makes sense, given that reflects the distribution of our data as we have seen in the scatter plots.

3.5 Selected Models

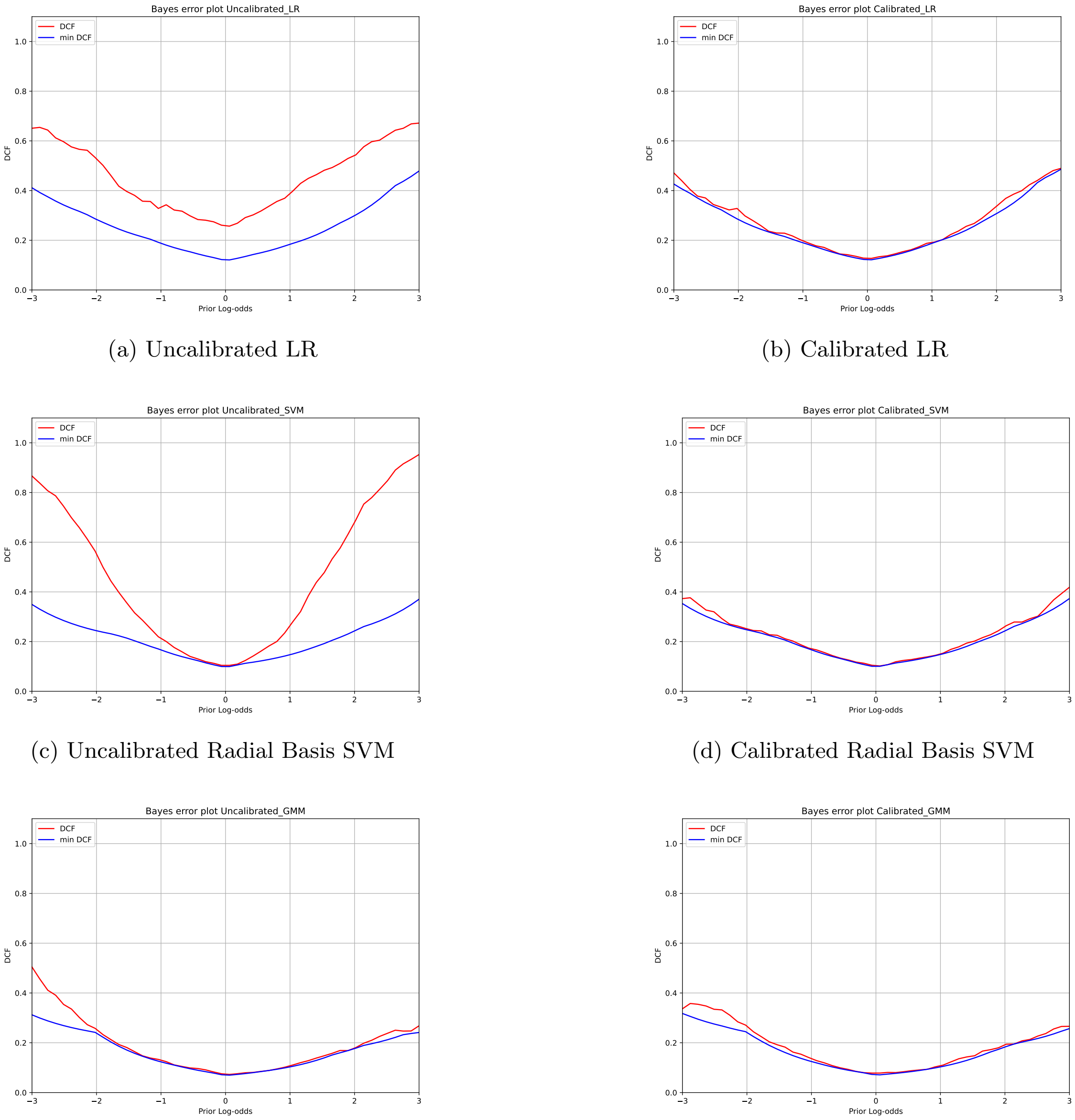
To sum up, our best models are:

Model	$\pi = \mathbf{0.5}$	$\pi = \mathbf{0.1}$	$\pi = \mathbf{0.9}$
RAW LogReg , $\lambda = \mathbf{0.01}$	0.118	0.311	0.330
Radial Basis SVM, C=5, $\pi_T = \mathbf{0.5}$	0.095	0.255	0.263
Tied GMM / 8 Comp	0.069	0.251	0.179

The data compares the selected models and we will deepen the study, highlighting how TiedGMM is the most promising.

4 Calibration

Up to this point, our assessment of model performance has centered around the minDCF metric, accounting for the cost associated with having knowledge of the optimal threshold in advance. Expanding our evaluation criteria, we now incorporate another metric called actualDCF (actDCF). Our current emphasis lies in examining the disparity between actDCF and minDCF, which signifies the loss attributed to inaccurately calibrated scores.



(a) Uncalibrated LR

(b) Calibrated LR

(c) Uncalibrated Radial Basis SVM

(d) Calibrated Radial Basis SVM

(e) Uncalibrated TiedGMM

(f) Calibrated TiedGMM

Figure 15: Comparison between the uncalibrated and calibrated version of the 3 chosen models

While the models Linear Logistic Regression and Radial Basis SVM are very uncalibrated, and you notice it from the distances of the curves in the graph, the TiedGMM is already very well calibrated even without additional operations. For the sake of completeness, the calibrated TiedGMM graph has also been added, so as to attention the slightest changes compared to the original template.

5 Evaluation

In the following tables are analyzed the results, for different applications, of the 3 models selected on the test set.

Model	min_dcf	act_dcf
LR	0.119	0.126
Radial SVM	0.097	0.1
TiedGMM	0.07	0.075

(a) Train set, $\pi = 0.5$

Model (Test)	min_dcf	act_dcf
LR	0.122	0.127
Radial SVM	0.107	0.111
TiedGMM	0.064	0.065

(b) Test set, $\pi = 0.5$

Model	min_dcf	act_dcf
LR	0.338	0.379
Radial SVM	0.266	0.288
TiedGMM	0.199	0.2

(c) Train set, $\pi = 0.9$

Model	min_dcf	act_dcf
LR	0.282	0.292
Radial SVM	0.312	0.323
TiedGMM	0.165	0.176

(d) Test set, $\pi = 0.9$

Model	min_dcf	act_dcf
LR	0.312	0.321
Radial SVM	0.261	0.27
TiedGMM	0.255	0.295

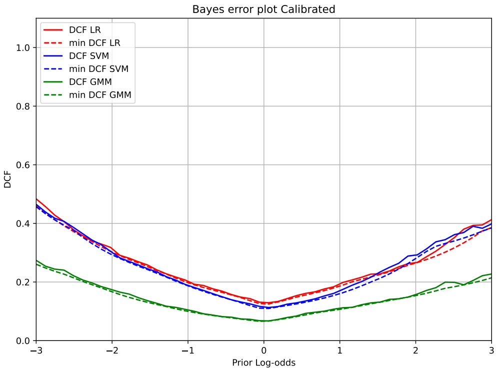
(e) Train set, $\pi = 0.1$

Model	min_dcf	act_dcf
LR	0.328	0.335
Radial SVM	0.32	0.338
TiedGMM	0.185	0.189

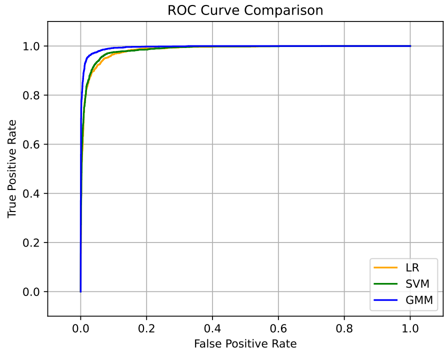
(f) Test set, $\pi = 0.1$

Figure 16: Analysis of the results of the 3 models on the Test set

All models generally get good results, but once again the TiedGMM is confirmed as the best model. For each of the π values it can achieve even better performance than those obtained on the Test Set. The following graphs further confirm the data and in particular: The ROC (Receiver Operating Characteristic) curve displays the relationship between sensitivity (True Positive Rate) and specificity (1 - False Positive Rate) over a range of classification thresholds. An ideal model will have a ROC curve that is as close as possible to the point (0,1) on the diagonal of the graph, indicating 100% sensitivity and 100% specificity. The Bayes Error Plot depicts the difference between minDCF and actDCF after calibration (as mentioned above) and, although calibration is effective for all models, TiedGMM results remain the best.



(a) Bayes Error Comparison



(b) Roc Comparison

5.1 Logistic Regression Evaluation

Just like in the training phase, we illustrate the varying minDCF values with changes in the parameter λ across the identical range of values experimented with during the training phase.

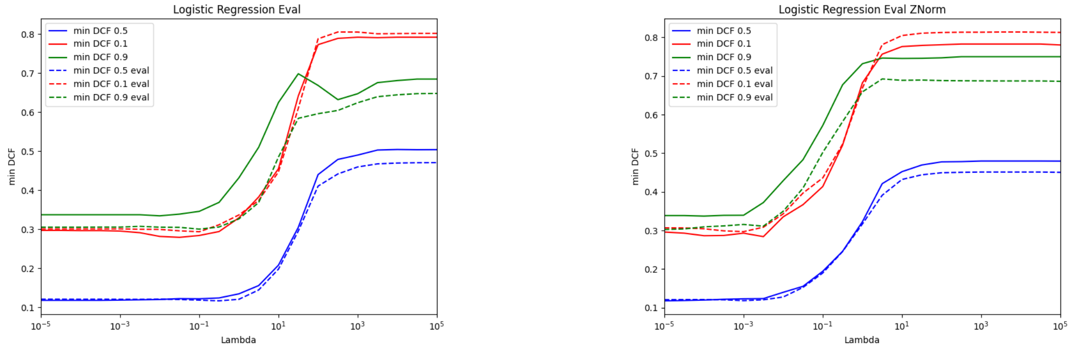


Figure 18: Linear Logistic Regression Evaluation RAW and with ZNorm and $\pi_T = 0.5$

As observed, when lambda takes on low values, the outcomes on both the training and test sets are nearly identical, indicating the effectiveness of the adopted parameters.

5.2 Radial Basis SVM Evaluation

The second classifier whose behavior we want to analyze is the SVM. It aligns with Logistic Regression in the case of linear SVM, while showcasing even more promising results with the application of the Radial Basis Kernel.

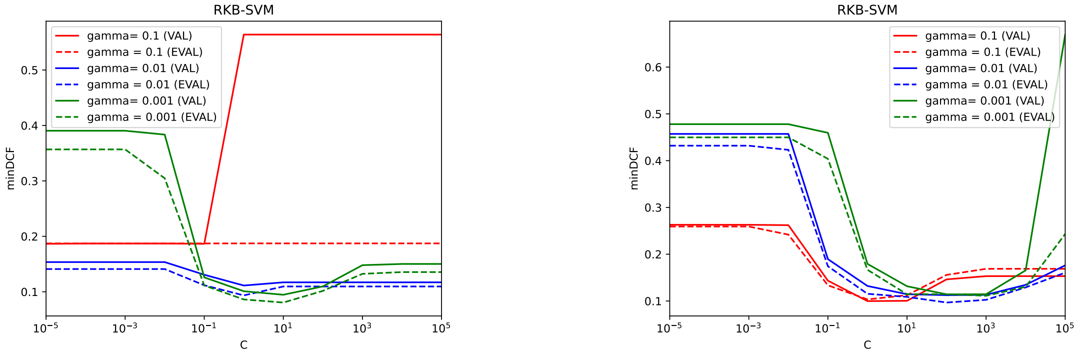


Figure 19: Radial Basis Kernel SVM Evaluation RAW and with ZNorm and $\pi_T = 0.5$

During the training phase, our selection for the best model was the RKB-SVM with $\gamma = 0.1$, C $= 10$, Znorm features, and $\pi_T = 0.5$. As evident, this choice remains suitable for the evaluation, although with a slight decrease in performance. Conversely, when examining RAW features, the hyperparameters chosen during training ($\lambda = 0.001$ and C $= 10$) prove highly effective on the evaluation set, achieving a minDCF of about 0.08. It is apparent that our solution is suboptimal.

5.3 Tied GMM Evaluation

Now we focus on the evaluation of GMM models and in particular of TiedGMM with 8 components, which during the training phase has proven to be the most promising model. Nevertheless we will also consider the Tied GMM with a different number of components and the version with ZNorm, which during the training has achieved very similar results to the selected model.

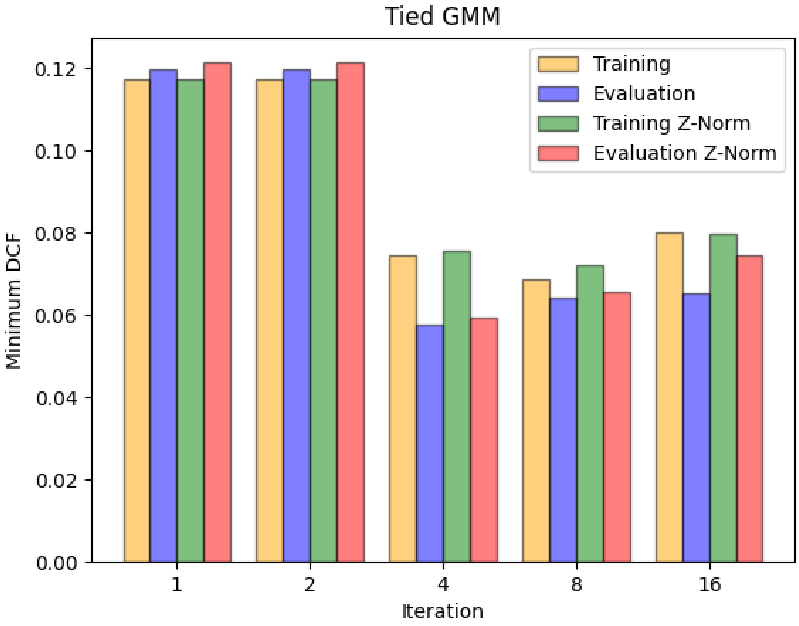


Figure 20: Tied GMM Evaluation RAW and with ZNorm and $\pi_T = 0.5$

Again the difference between the RAW model and the ZNorm model is minimal, with the former once again performing better than the latter. It should be noticed that the best results are obtained by TiedGMM with 4 components, where the difference between training results and evaluation is much wider.

6 Conclusions

In conclusion, it can be asserted that the decisions made during the training phase proved to be highly precise even during the evaluation phase. This can likely be attributed to the similarity in the characteristics of the training and test sets. The best model chosen in the training phase was the Tied GMM with 8 components which obtained a value of minDCF on the evaluation set approximately equal to 0.06, though the TiedGMM with 4 components gets slightly better performance evaluation. Overall the selected model lends itself very well and this outcome is expected, as even in the initial stages of our investigation, the dataset used to train the algorithms exhibited features that naturally inclined towards such a model.