GENDER CLASSIFICATION PROJECT REPORT

MACHINE LEARNING AND PATTERN RECOGNITION 2022/2023

Gabriele Quaranta , s318944

Introduction

Abstract

This report we will analyze a dataset consisting of low-level images of both males and females, utilizing various Machine Learning (ML) algorithms. The primary step is to gain insights into the dataset's structure, its distribution, and then to use different classifiers with the aim of creating a accurate gender classification system.

The families of models we will employ are the following:

- Gaussian Classifiers
- Discriminative Models (Logistic Regression)
- Non-probabilistic Models (Linear, Poly, RBF SVMs)
- Gaussian Mixture Models

Data Information

The project's goal is the creation of a gender classifier based on high-level features extracted from facial images. Such tools find applications in various fields, including gender-dependent face recognition models.

The dataset is composed by image embeddings, which are compact, low-dimensional representations of facial images.

The dataset exhibits an imbalance; the training set contains an excess of female samples, while the test set is unbalanced towards male samples. Additionally, the samples are categorized into three age groups, each potentially characterized by distinct embedding distributions, but age information is unavailable.

Models Performance and Evaluation

Our ultimate goal is to identify the most effective model that can address the challenges posed by the imbalanced dataset and deliver accurate results across diverse application scenarios.

To achieve this objective, we have implemented pre-processing techniques and employed a K-Fold cross-validation approach with K=5. This choice ensures a good amount of data for both training and validation, an important consideration given the significant class imbalance in our dataset.

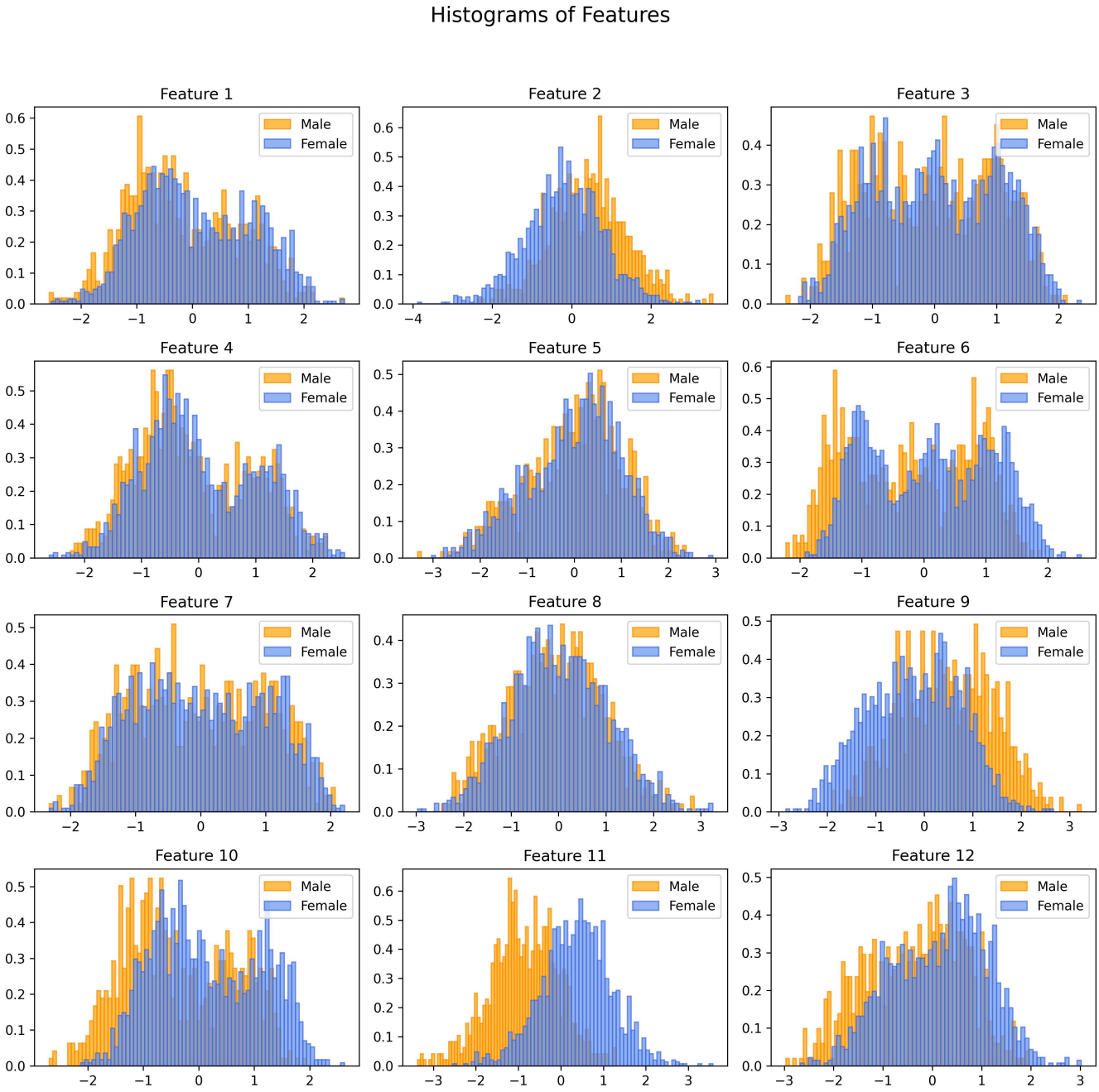
Our primary focus is on the target application defined by the triplet ($\tilde{\pi}$, Cfn, Cfp) = (0.5, 1, 1). However, we extend our analysis to other application scenarios ($\tilde{\pi}$, Cfn, Cfp) = (0.1, 1, 1) and ($\tilde{\pi}$, Cfn, Cfp) = (0.9, 1, 1).

To assess model performance effectively, we will consider the normalized minimum detection costs. This metric takes into account the costs associated with different error types, providing a comprehensive evaluation of a model's ability.

Dataset Features Analysis

Features distribution

We begin with the creation of histograms to visually show the distribution of each feature.

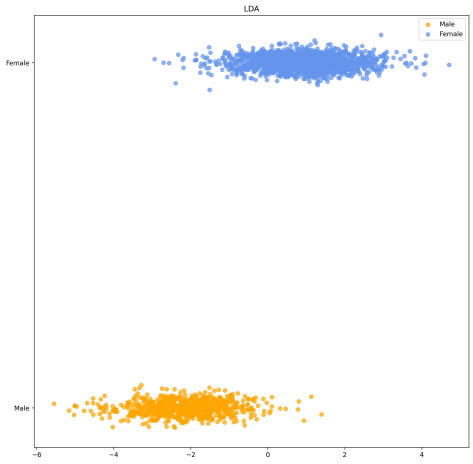


We observe that several histograms, particularly figures 3 and 6, exhibit a pattern reminiscent of a Gaussian distribution, with three distinct components. This observation corresponds with our prior knowledge of the dataset's division into three age groups. Numerous other histograms also display a Gaussian-like distribution, hinting at the potential suitability of Gaussian Models for this dataset.

The distribution of individual features seems quite similar in both gender classes. However, there are specific distributions that are more discriminative between the two classes, like Feature 11.

We can apply Linear Discriminant Analysis (LDA) to the entire dataset. This analysis aims to assess whether our features exhibit linear discriminability: it can helps us determine whether linear and/or Gaussian models might outperform non-Gaussian and/or non-linear models.

LDA identifies C-1 directions for projecting our features, with C representing the number of classes.

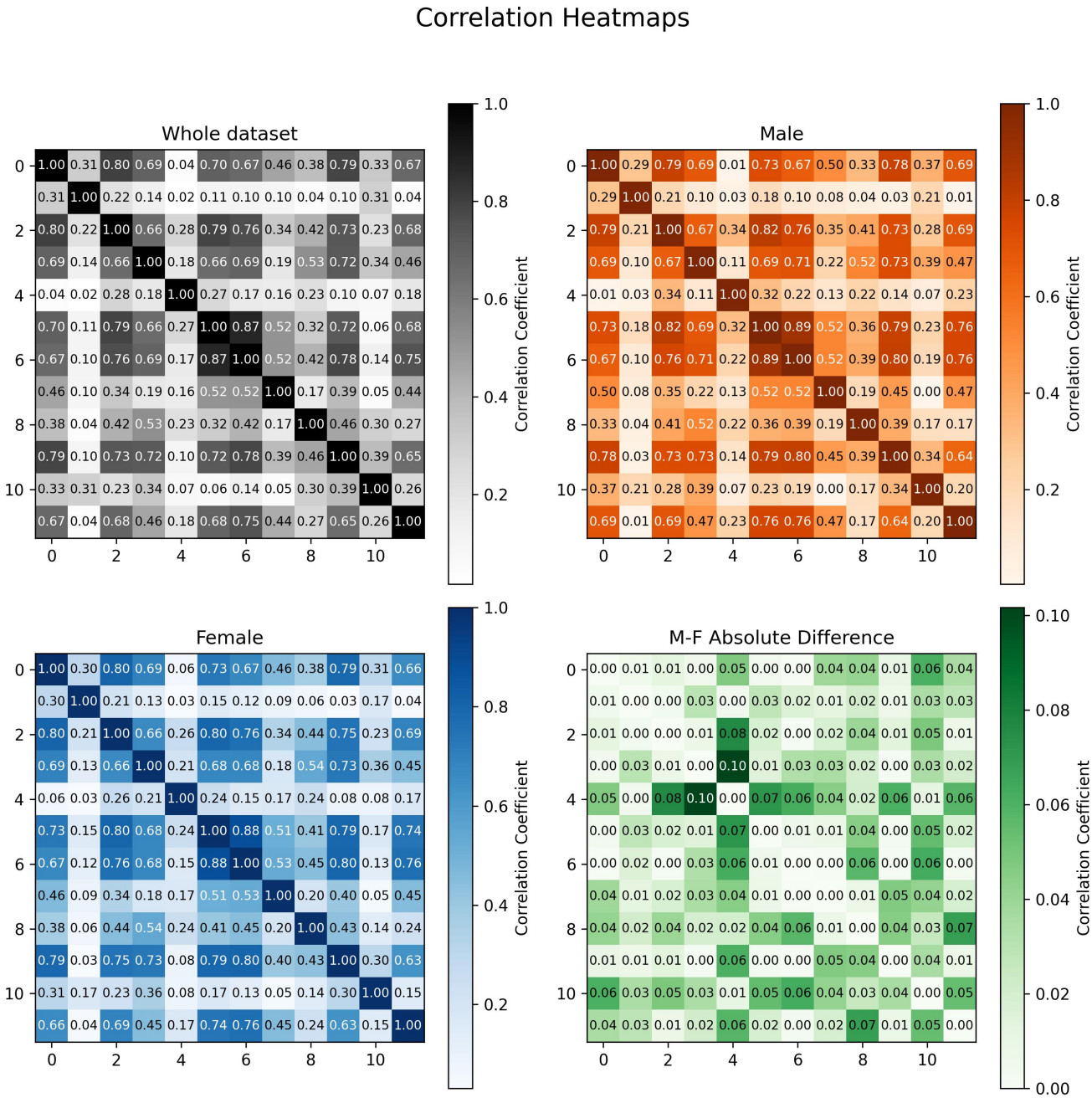


In our binary classification scenario, the application of Linear Discriminant Analysis (LDA) yields a one-dimensional transformation, which, when plotted on a 2D graph, may lack visual appeal and clarity. To enhance the readability we have introduced random jitter to the y-axes.

The classes exhibit a separation for the most part. However we note the presence of a overlapping region, where the potential for classification errors increases.

Features correlation

We will now explore the Pearson Correlation among our features.



We first consider the "Whole dataset" heatmap. We observe strong correlations between certain feature pairs, specifically (0, 2), (0, 9), (2, 5), and (5, 6). Most other features exhibit moderate correlations, with coefficients roughly averaging between 0.4 and 0.5.

This suggests that we might derive benefits from reducing the dimensionality of our data. We will use PCA to reduce the dimensions up to 9, but we will probability experience a some amount of loss.

Focusing on to the heatmaps for individual classes, it is clear that the two heatmaps have a striking resemblance to each other ("M-F Absolute Difference"). This similarity implies that Multivariate Gaussian Models and Tied Gaussian Models might exhibit comparable performance characteristics.

The presence of mild correlation leads us to assume that classifiers based on the Naïve hypothesis may not perform as effectively.

Gaussian Classifiers

Gaussian classifiers are a category of probabilistic machine learning algorithms specifically designed for classification tasks. They operate on the assumption that the features within the dataset follow a Gaussian (normal) distribution.

In Gaussian classifiers, the selection of the covariance matrix structure is very important as it directly influences the modeling of feature distributions for each class. Different covariance matrix structures can significantly impact the classifier's performance.

Below are the four types of gaussian classifers considered:

1. Full Covariance (Full-Cov) Gaussian Classifier:
   - The full covariance matrix contains variances and covariances between all pairs of features.
2. Diagonal Covariance (Diag-Cov) Gaussian Classifier:
   - The diagonal covariance matrix contains only the variances of individual features on the diagonal, and it assumes that features are uncorrelated.
3. Tied Full Covariance (Tied Full-Cov) Gaussian Classifier:
   - In a Tied Full Covariance Gaussian classifier, all classes share a single full covariance matrix.
4. Tied Diagonal Covariance (Tied Diag-Cov) Gaussian Classifier:
   - In a Tied Diagonal Covariance Gaussian classifier, all classes share a single diagonal covariance matrix.

GAUSSIAN	Prior = 0.5	Prior = 0.1	Prior = 0.9
RAW			
Full-Cov	0.113	0.297	0.350
Diag-Cov	0.463	0.771	0.777
Tied FC	0.109	0.299	0.342
Tied DC	0.457	0.770	0.781
PCA m = 11			
Full-Cov	0.122	0.312	0.358
Diag-Cov	0.124	0.312	0.349
Tied FC	0.118	0.299	0.356
Tied DC	0.123	0.294	0.355
PCA m = 10			
Full-Cov	0.187	0.407	0.538
Diag-Cov	0.184	0.435	0.546
Tied FC	0.183	0.428	0.535
Tied DC	0.179	0.421	0.543
PCA m = 9			
Full-Cov	0.220	0.500	0.578
Diag-Cov	0.208	0.486	0.597
Tied FC	0.212	0.476	0.577
Tied DC	0.210	0.482	0.589

The results align with our expectations.

For the raw data the Naïve Bayes consideration we did before regarding the features (due to the presence of moderate correlations among the features, as observed in the heatmaps) holds true.

However we can see this issue disappearing after applying Principal Component Analysis (PCA): it projects the samples into a lower dimensional space while preserving the highest variance (effectively diagonalizing the covariance matrix).

In most cases, Multivariate Gaussian (MVG) and Tied Multivariate Gaussian (TMVG) models exhibited comparable performance. MVG assumes two distinct covariance matrices for the classes, while TMVG, even considering the similarity of covariance matrices between two classes, demonstrated comparable results.

Reducing dimensionality beyond 11 components resulted in a loss of information and a deterioration in model performance. Based on these findings, we have chosen to continue our analysis using the raw data and PCA with 11 components.

Discriminative Models

Logistic Regression is a statistical model primarily used for binary classification, but it can be extended to multi-class problems.
The main concept behind logistic regression is in the use of the sigmoid function (logistic function) to model the relationship between input features and the probability of the input belonging to the positive class (class 1). It ensures that the predicted probabilities always fall within the (0, 1) range.

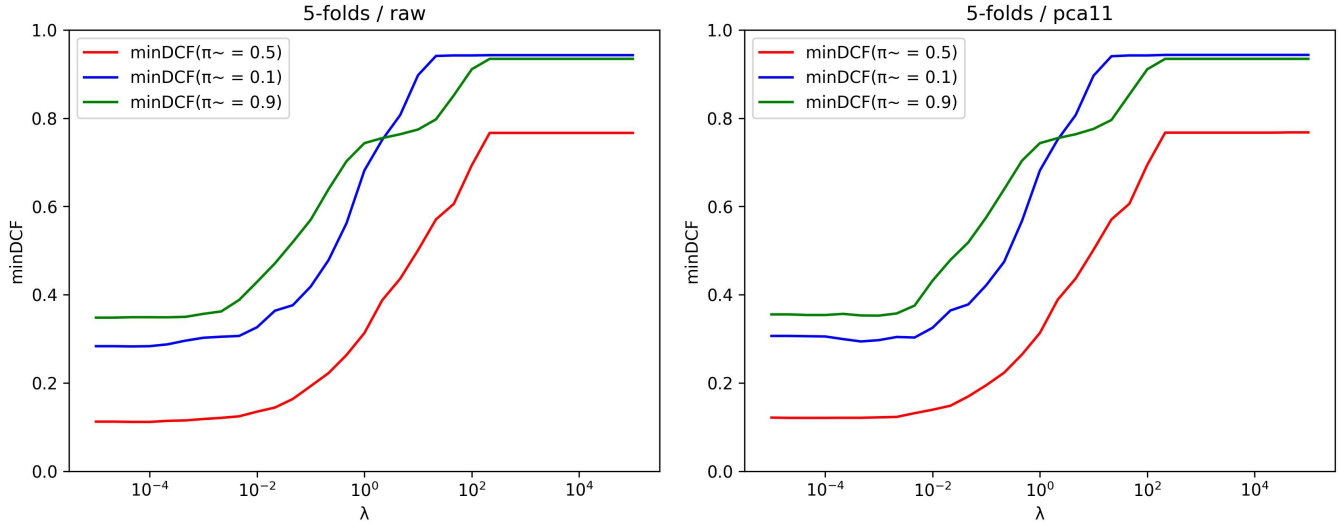
Given class imbalance, we adopt a regularized version of the objective function.

$$J(\omega, b) = \frac{\lambda}{2}||\omega||^2 + \frac{\pi_T}{n_T}\sum_{i=1}^{n}\log(1+e^{-z_i s_i}) + \frac{1-\pi_T}{n_F}\sum_{i=1}^{n}\log(1+e^{-z_i s_i})$$

$$\text{with } s_i = (\omega^T x_i + b)$$

This objective function includes model parameters (ω, b), a regularization term $\frac{\lambda}{2}||\omega||^2$, and λ, a regularization coefficient hyperparameter.

To select an appropriate value for λ, we plot the minDCF (minimum Detection Cost Function) graphs across a range of λ values.



In the context of our target application, (π̃, Cfn, Cfp = 0.5, 1, 1), we have identified a potential value for the hyperparameter λ (the regularization coefficient) as 10^{-4}.

The selection of λ is very important as it significantly influences model performance:

- When λ is too large, the model struggles to correctly classify samples. This is because excessive regularization simplifies the model too much.
- For small values of λ, the model excels in training data separation and generalizes well to unseen data. In such cases, regularization has minimal to no effect on improving model performance.

Here are the MinDCF obtained by considering the above factors:

LOGREG	Prior = 0.5	Prior = 0.1	Prior = 0.9
RAW			
LR(λ = 1.00e-04, πT = 0.5)	0.112	0.283	0.349
LR(λ = 1.00e-04, πT = 0.1)	0.123	0.296	0.371
LR(λ = 1.00e-04, πT = 0.9)	0.112	0.316	0.344
PCA m = 11			
LR(λ = 1.00e-04, πT = 0.5)	0.121	0.305	0.354
LR(λ = 1.00e-04, πT = 0.1)	0.127	0.302	0.373
LR(λ = 1.00e-04, πT = 0.9)	0.115	0.315	0.353

Different values for the threshold parameter πT do not significantly improve the model. There is a slight improvement when using πT = 0.9, but this is mainly due to the class imbalance in the dataset, with a bias towards class 1 (female).

Logistic Regression overall performs well, as we hypothesized during LDA analysis of the features.

Non Probabilistic models

Support Vector Machine (SVM) models seek to find a hyperplane that effectively separates different classes. SVM takes a distinct approach by aiming to identify the hyperplane that maximizes the "maximum margin hyperplane." This margin represents the widest possible gap between the classes, facilitating classification.

To solve the SVM problem we can consider the dual formulation, which is easier to optimize (its complexity depends only on the number of samples), and it allows us to compute non-linear hyperplanes without the need to explicitly expand
the features :

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

The objective is subject to certain constraints, including $0 \leq \alpha_i \leq C$ for all i in the range from 1 to n, where C is a hyperparameter, and $\sum_i \alpha_i \cdot z_i = 0$, where z_i represents the class labels of the training samples.

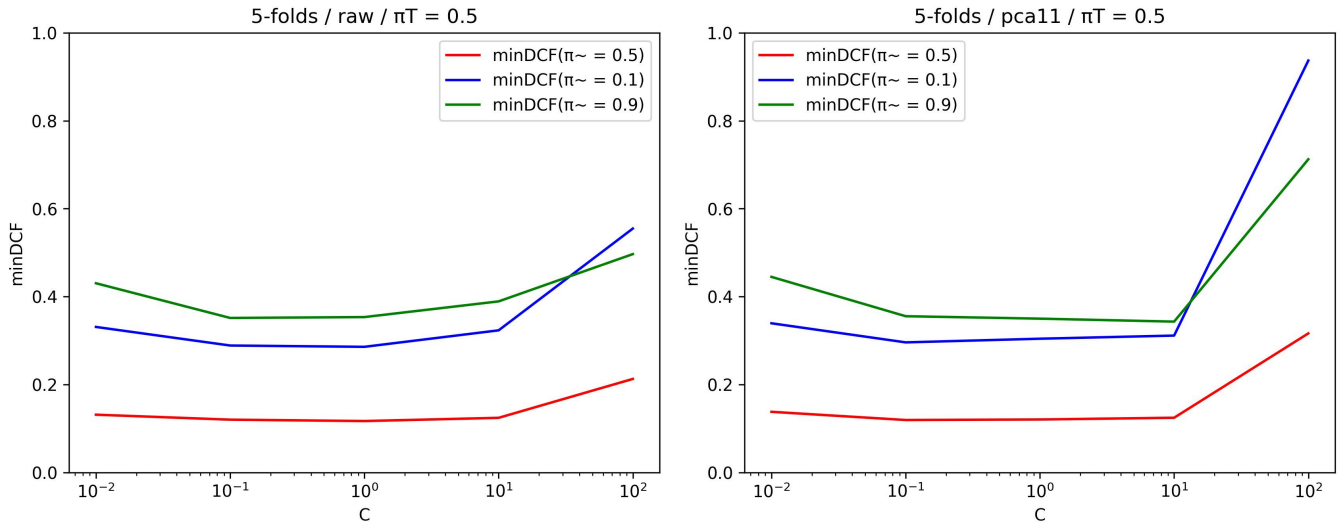
In our exploration of SVM, we consider a balanced version. This approach involves adjusting the hyperparameter C for each class individually in the dual formulation's box constraint.

C is a hyperparameter that requires careful tuning. We explore a range of values for C spanning from 10^{-2} to 10^2. As we have observed previously, we generally anticipate that a linear decision rule will outperform the quadratic one.

The hyperparameter C plays a the role of a tradeoff factor. It balances between achieving low training errors and maximizing the margin between classes:

- When C tends towards infinity, the model places a strong emphasis on minimizing training errors. This can result in poor generalization and overfitting.
- When C approaches zero, the model prioritizes a wide margin over the training data, ignoring training errors in favor of maximizing separation.

Linear SVM



From these plots we decided for a suitable value of C to be C = 1. for both raw and PCA11 data as the graphs are quite similar.

Here are the minDCF values obtained for a linear SVM model:

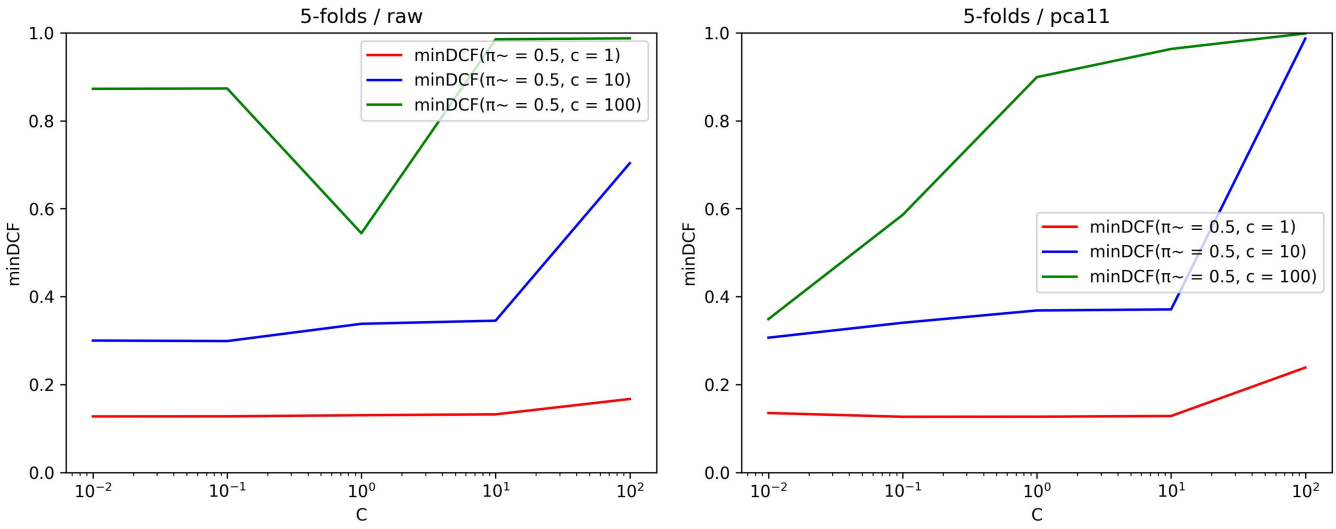
LINEAR SVM	Prior = 0.5	Prior = 0.1	Prior = 0.9
RAW			
Linear SVM(C = 1, πT = 0.5)	0.117	0.286	0.353
Linear SVM(C = 1, πT = 0.1)	0.128	0.308	0.379
Linear SVM(C = 1, πT = 0.9)	0.112	0.318	0.334
PCA m = 11			
Linear SVM(C = 1, πT = 0.5)	0.120	0.304	0.350

LINEAR SVM	Prior = 0.5	Prior = 0.1	Prior = 0.9
Linear SVM(C = 1, πT = 0.1)	0.132	0.310	0.382
Linear SVM(C = 1, πT = 0.9)	0.122	0.333	0.354

The results are promising and reinforce our assumption regarding the effectiveness of a linear decision rule. Again the model exhibits good performance also in the 0-9 unbalanced application.

Polynomial kernel SVM

We will explore the polynomial SVM with a degree 2 i.e. Quadratic SVM.



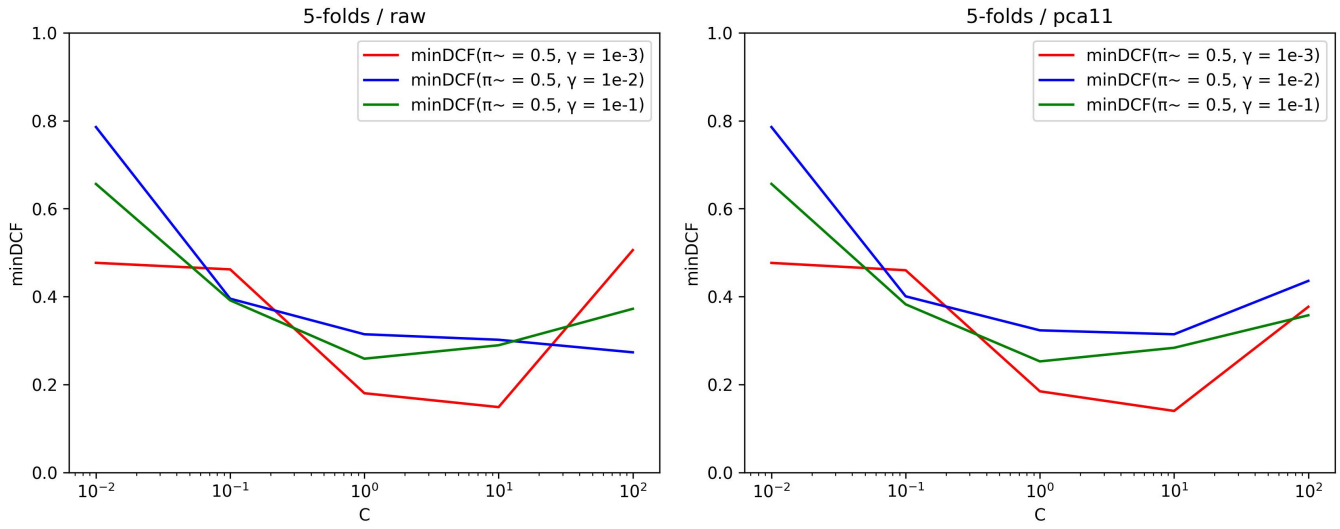
From these plots we decided for a suitable value of C to be C = 0.1.

Here are the minDCF values obtained for a quadratic SVM model:

POLY SVM	Prior = 0.5	Prior = 0.1	Prior = 0.9
RAW			
Poly SVM(C = 0.1, c = 1, d = 2, πT = 0.5)	0.128	0.317	0.348
Poly SVM(C = 0.1, c = 1, d = 2, πT = 0.1)	0.143	0.365	0.444
Poly SVM(C = 0.1, c = 1, d = 2, πT = 0.9)	0.139	0.404	0.317
PCA m = 11			
Poly SVM(C = 0.1, c = 1, d = 2, πT = 0.5)	0.127	0.338	0.340
Poly SVM(C = 0.1, c = 1, d = 2, πT = 0.1)	0.157	0.385	0.446
Poly SVM(C = 0.1, c = 1, d = 2, πT = 0.9)	0.133	0.407	0.330

We can see similar but slightly worse performance with respect to the linear model, which is what we expected.

RBF kernel SVM



From these plots we decided for a suitable value of the hyper-parameter to be C = 10 and γ = 0.1

Here are the minDCF values obtained for a RBF SVM model:

RBF SVM	Prior = 0.5	Prior = 0.1	Prior = 0.9
RAW			
RBF SVM(C = 10, γ = 0.1, πT = 0.5)	0.095	0.275	0.284
RBF SVM(C = 10, γ = 0.1, πT = 0.1)	0.115	0.279	0.367
RBF SVM(C = 10, γ = 0.1, πT = 0.9)	0.110	0.338	0.262
PCA m = 11			
RBF SVM(C = 10, γ = 0.1, πT = 0.5)	0.095	0.275	0.284
RBF SVM(C = 10, γ = 0.1, πT = 0.1)	0.115	0.279	0.367
RBF SVM(C = 10, γ = 0.1, πT = 0.9)	0.110	0.338	0.262

Notably we can observe RBF SVM in the target application gives us the best score so far.

Gaussian Mixture models

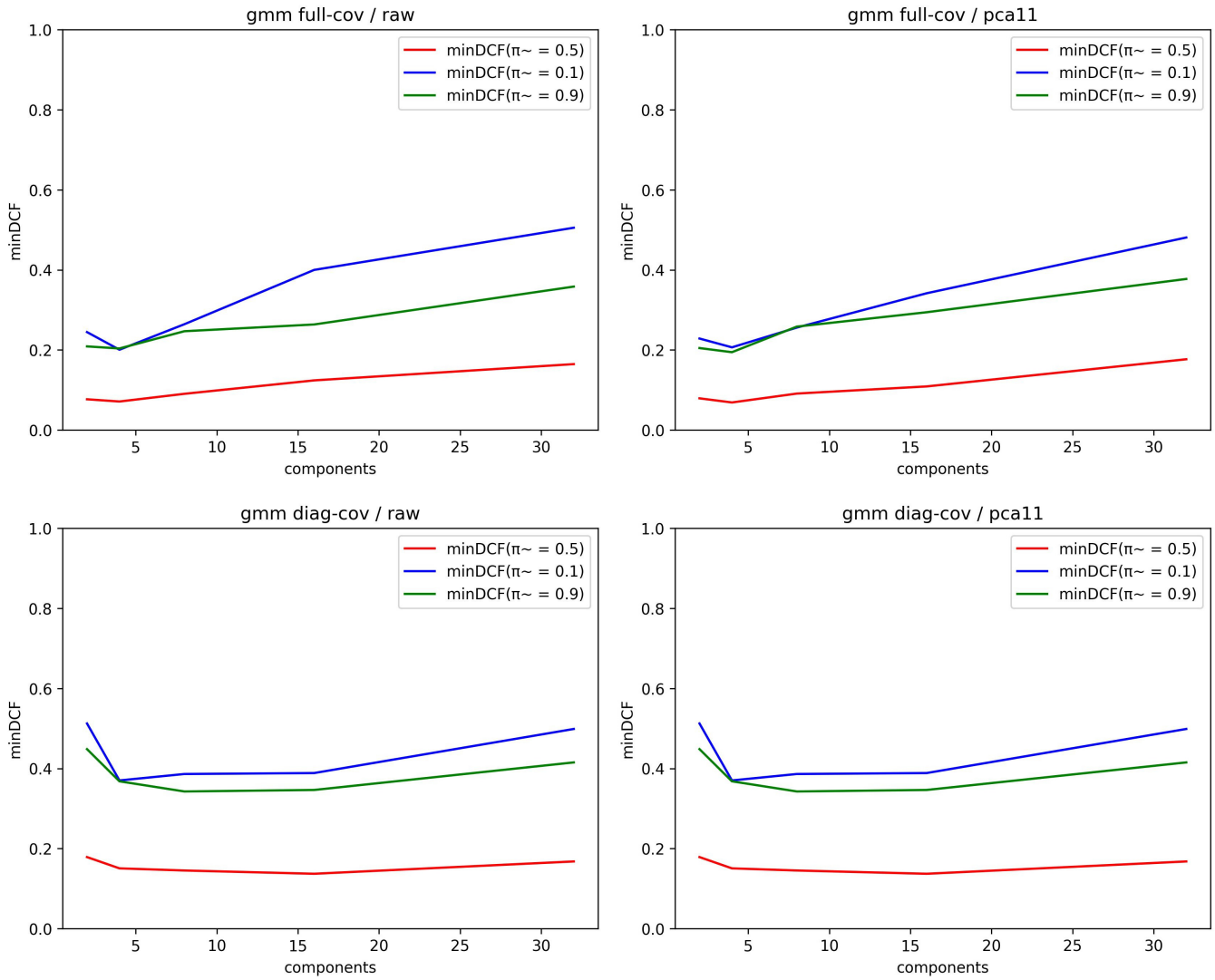
Finally we turn our attention to Gaussian Mixture Models (GMMs). These models are primarily utilized for density estimation tasks and operate on the fundamental assumption that our data can be effectively characterized as a combination of Gaussian distributions, which may include multiple components.

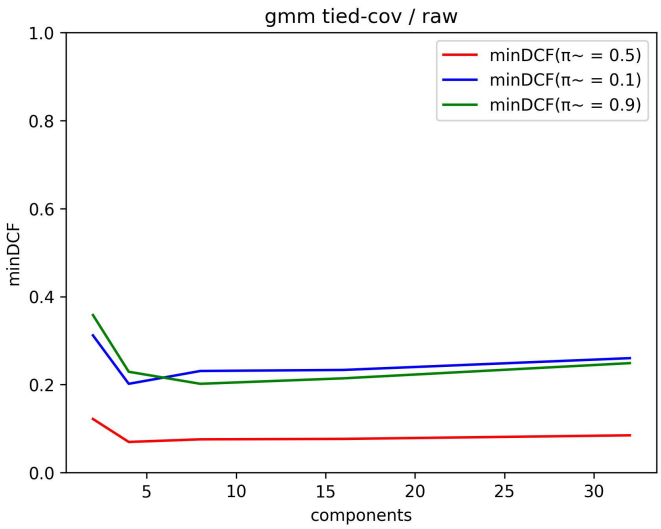
GMMs operate under the premise that each data sample arises from a mixture of Gaussian distributions, each with its own set of parameters. The specific count of these distributions is a hyperparameter we will need to evaluate.

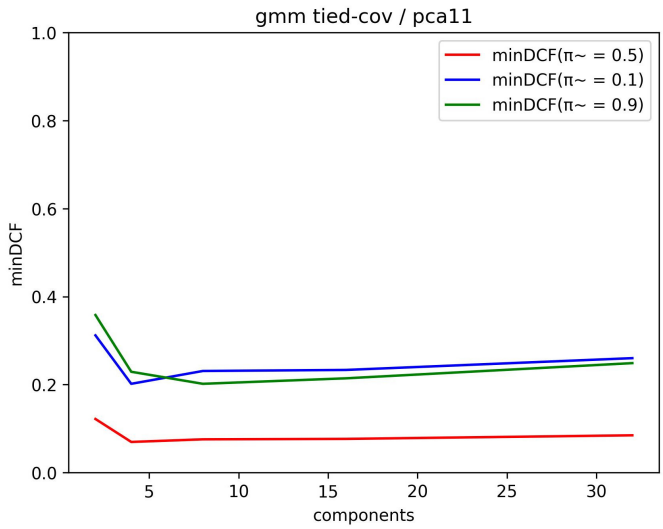
Our earlier findings on Gaussian models, along with the dataset analysis, suggest that GMMs, particularly Tied GMMs, have the potential to perform well.

We anticipate that the most favorable results for Gaussian Mixture Models will be achieved with 2 or 4 components. This choice aligns with our dataset analysis and prior information on age groups, which revealed that our training set exhibits a distribution reminiscent of a Gaussian with 3 components or clusters. The nearest whole numbers to this distribution's characteristics are 2 and 4 (GMMs are typically trained with a number of clusters that are powers of 2).

The hyperparameter we need to tune is the number of components for each Gaussian distribution, to do so we again consider the minDCF plots.







From these plots we landed on the following number of components:

- Full Covariance : 4 components
- Diagonal Covariance : 16 components
- Tied Covariance : 4 components

Here are the minCF values obtained using these values:

	Prior = 0.5	Prior = 0.1	Prior = 0.9
RAW			
GMM Full (4 components)	0.071	0.201	0.204
GMM Diag (16 components)	0.197	0.501	0.474
GMM Tied (4 components)	0.068	0.237	0.222
PCA m = 11			
GMM Full (4 components)	0.069	0.207	0.194
GMM Diag (16 components)	0.137	0.389	0.346
GMM Tied (4 components)	0.070	0.202	0.229

GMMs give us the new best modeloverall : GMM Tied with 4 components on RAW data. This results align with all previous considerations.

Score calibration

The best classifiers at this point are:

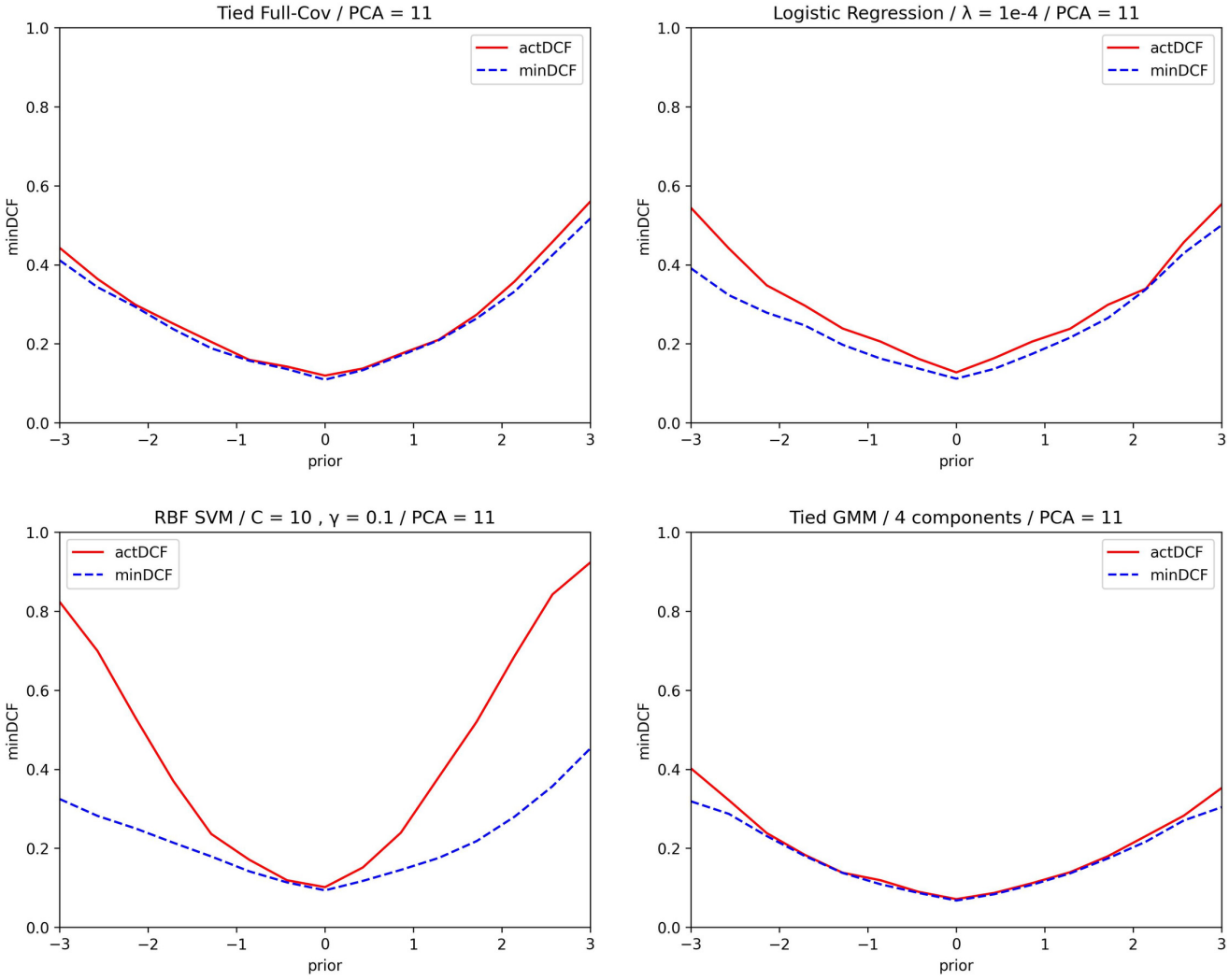
	Prior = 0.5	Prior = 0.1	Prior = 0.9
RAW Tied FC	0.109	0.299	0.342
RAW LR(λ = 1.00e-04, πT = 0.5)	0.112	0.283	0.349
PCA11 RBF SVM(C = 10, γ = 0.1, πT = 0.5)	0.095	0.275	0.284
PCA11 GMM Tied (4 components)	0.068	0.237	0.222

We will consider the dataset with reduced dimensionality PCA11 for all models moving forward.

Up until now, our primary metric for evaluating various models has been the minimum Detection Cost Function (minDCF). However, it's important to acknowledge that the actual cost depends on the effectiveness of the threshold used for class assignment. To address this we now introduce the concept of the actual Detection Cost Function (actDCF).

The key distinction between actDCF and minDCF lies in how classification decisions are made. In actDCF, classification is performed using the threshold corresponding to the target value π, rather than testing all possible thresholds and selecting the one with the lowest DCF.

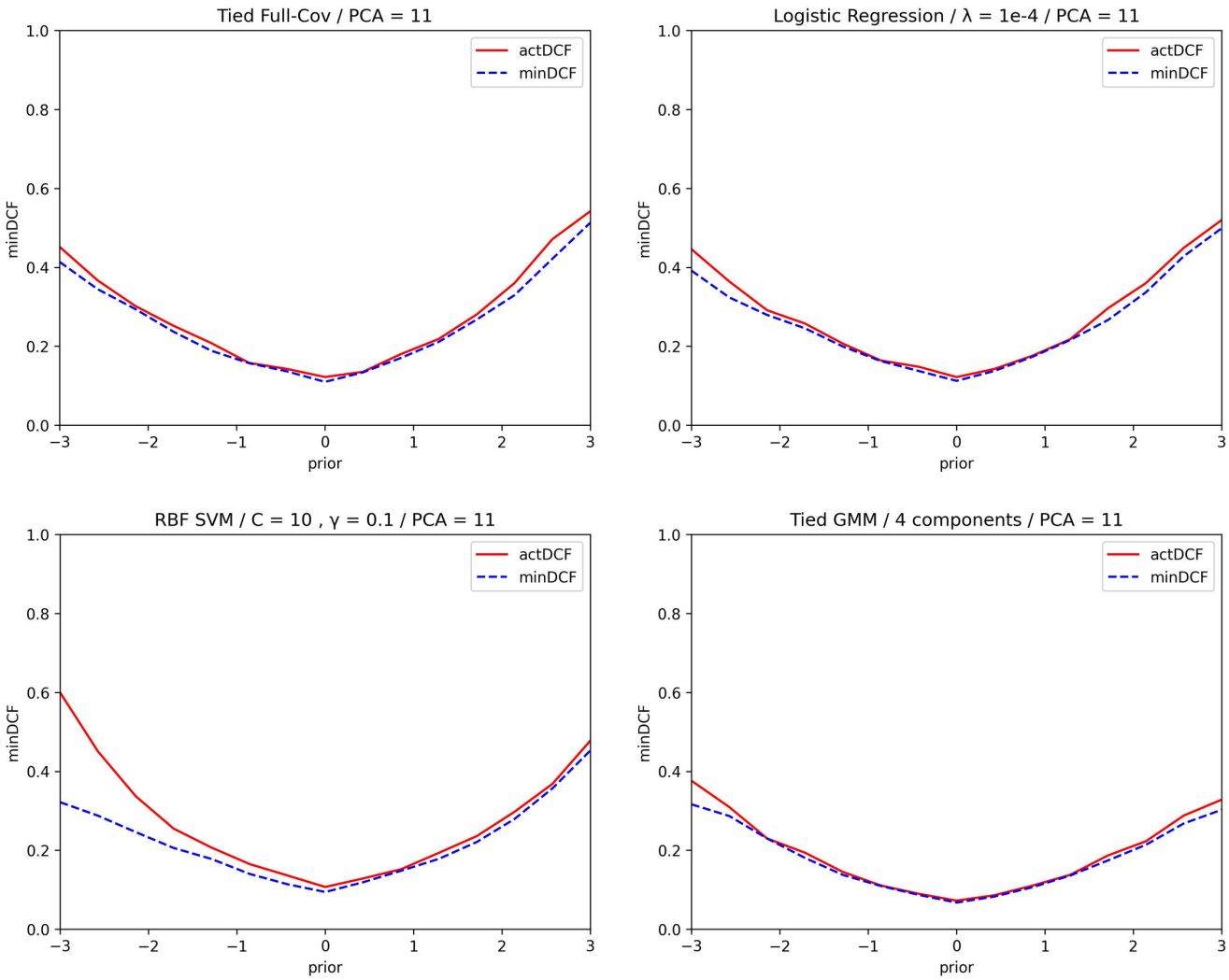
The Bayes Error Plots provided below illustrate the differences between minDCF and actDCF for our selected models non calibrated. A larger gap in these plots indicates a greater loss due to score mis-calibration when using the selected threshold.



To address the problem of threshold-dependent cost assessment, we employ Logistic Regression. Logistic Regression acts as a posterior log-likelihood ratio, allowing us to recover the calibrated score by simply subtracting the theoretical threshold.

To estimate the parameters of the calibration function, we will adopt a K-Fold Approach. This approach is particularly suited for our task due to the limited number of available samples.

Here are the calibrated plots:



As we can see the LogReg and RBF SVM benefited more from the calibration while gaussian models were already well calibrated.

This are the actCF and minDCF for the calibrated models:

minDCF	Prior = 0.5	Prior = 0.1	Prior = 0.9
RAW			
Tied Full-Cov	0.110	0.299	0.339
LogReg (λ = 1e-4, πT = 0.5)	0.112	0.284	0.348
RBF SVM (C = 10, γ = 0.1, πT = 0.5)	0.094	0.251	0.288
GMM Tied	0.067	0.236	0.220
PCA11			
Tied Full-Cov (PCA m = 11)	0.119	0.303	0.355
LogReg (λ = 1e-4, πT = 0.5)	0.121	0.306	0.353
RBF SVM (C = 10, γ = 0.1, πT = 0.5)	0.095	0.264	0.281
GMM Tied (4 components)	0.070	0.206	0.228

actDCF	Prior = 0.5	Prior = 0.1	Prior = 0.9
RAW			
Tied Full-Cov	0.122	0.301	0.372
LogReg (λ = 1e-4, πT = 0.5)	0.122	0.289	0.365
RBF SVM (C = 10, γ = 0.1, πT = 0.5)	0.107	0.352	0.306
GMM Tied (4 components)	0.072	0.236	0.224

actDCF	Prior = 0.5	Prior = 0.1	Prior = 0.9
PCA11			
Tied Full-Cov	0.120	0.318	0.369
LogReg (λ = 1e-4, πT = 0.5)	0.122	0.317	0.382
RBF SVM (C = 10, γ = 0.1, πT = 0.5)	0.106	0.358	0.289
GMM Tied (4 components)	0.072	0.255	0.244

We can observe improvement in performance for many of the best models so we will use them for the final step, the classification on the validation dataset.

Final Evaluation

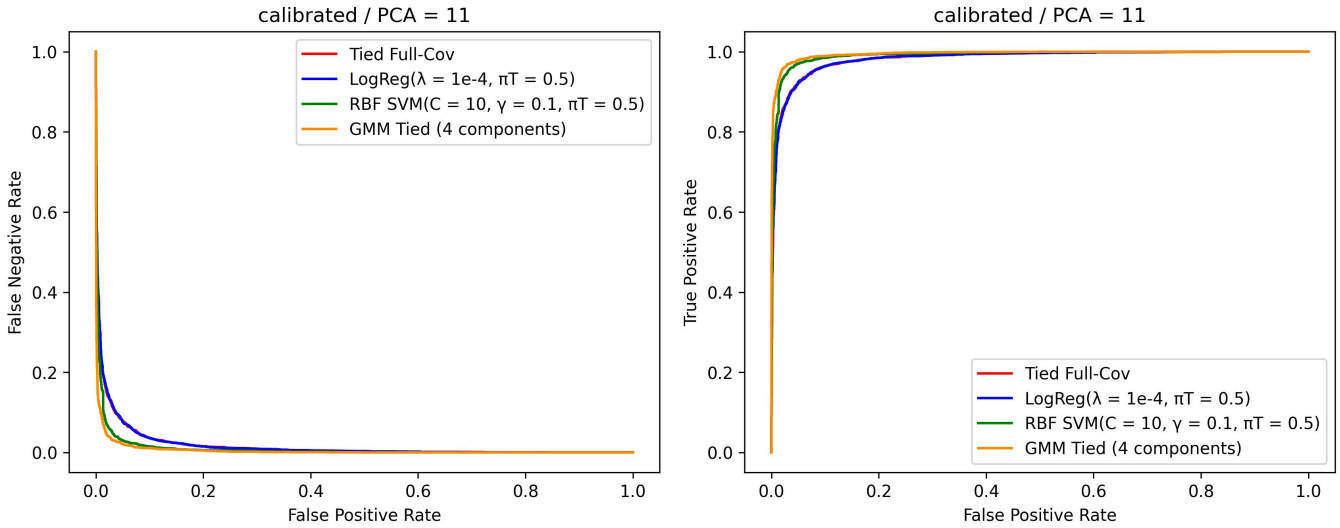
During the evaluation phase, we will only consider the models that showcased the most promising performance during the training phase.

Our objective is to assess their quality using the evaluation data. For this we continue to rely on minDCFs and actDCFs.

Below, are the values obtained on the evaluation data, which has been pre-processed with PCA11. Additionally, we plotted the DET and ROC curves for these models:

minDCF	Prior = 0.5	Prior = 0.1	Prior = 0.9
Tied Full-Cov	0.124	0.311	0.335
LogReg (λ = 1e-4, πT = 0.5)	0.124	0.314	0.326
RBF SVM (C = 10, γ = 0.1, πT = 0.5)	0.076	0.225	0.220
GMM Tied (4 components)	0.061	0.169	0.186

actDCF	Prior = 0.5	Prior = 0.1	Prior = 0.9
Tied Full-Cov	0.126	0.315	0.342
LogReg (λ = 1e-4, πT = 0.5)	0.127	0.318	0.327
RBF SVM (C = 10, γ = 0.1, πT = 0.5)	0.087	0.285	0.231
GMM Tied (4 components)	0.061	0.180	0.200



The results are consistent with the ones obtained during the previous analysis over the training set with the GMM Tied with 4 component showing the best performance on detecting the gender on the evaluation data.

Conclusion

The strategies and choices employed during the training phase have proven to be highly effective when applied to the test data, even considering the unbalance between training and valuation data.

Gaussian Mixture Model (GMM) with tied covariance on PCA11 data and 4 components has demonstrated its effectiveness in producing well-tuned scores across various application scenarios.

In our primary application (with $\tilde{\pi}$ = 0.5), we achieved an low DCF cost of 0.06. Additionally, for unbalanced scenarios with $\tilde{\pi}$ values of 0.1 and 0.9, we still obtained good values of DCF costs of 0.169 and 0.186, respectively.

This results mirror all the assumptions we made starting from the feature analysis of the dataset to the training phase, so we can consider this analysis to have proven correct.