MLPR 2023 – Gender identification

Gabriele Lucca - Matteo Martini
(s314297- s314786)

Abstract

The objective of this report is to examine a dataset containing diverse low-level images of both males and females through the application of a range of Machine Learning algorithms. A gender classifier in machine learning is a model or algorithm that is trained to predict or classify the gender of an individual based on input data. Gender classification can be approached in various ways, and the choice of method often depends on the available data and the specific problem being addressed. Features or characteristics of individuals are extracted from the dataset that includes examples of individuals along with their gender labels. The model is trained using the labeled dataset, with the goal of learning patterns and relationships that can distinguish between different genders based on the chosen features. The trained model is tested on a separate dataset to assess its accuracy and performance. Once the model is trained and evaluated satisfactorily, it can be deployed for gender classification tasks on new, unseen data. In the beginning, an evaluation of the dataset's composition will be undertaken, trying to understand its distribution. Subsequently, we will move forward with an investigation of the different classifiers to develop a system capable of classifying our samples with maximum accuracy while minimizing costs.

1. Features analysis

In the training dataset there are 2400 samples, with 720 representing males and 1680 representing females. Consequently, there is a substantial bias towards females, accounting for 70% of the dataset. Each sample comprises 12 features and represents a low-dimensional image generated by mapping images onto a common low-dimensional manifold. On the other hand, the test dataset comprises 6000 samples, with 4200 being males and 1800 being females. As a result, it does not mirror the class proportions of the training dataset but retains all other relevant information.
We will start the dataset analysis by creating some plots, beginning with an examination of the distribution of our features.

Histogram and 2D scatter plots of the dataset features

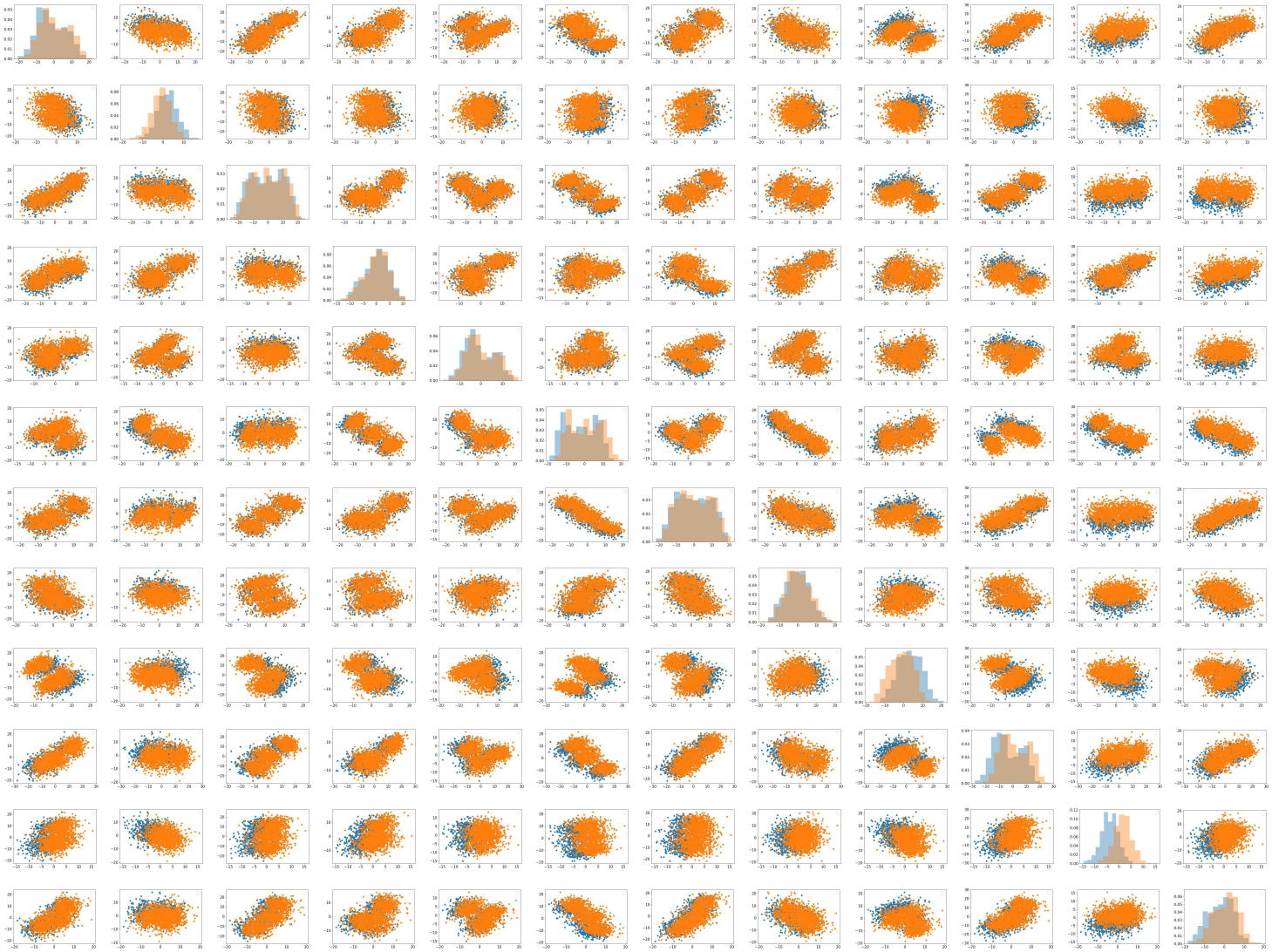

Fig. 1- Histogram and 2D scatter plots of the dataset features, where the male class are represented by blue and the female one are in orange

Some plotters exhibit characteristics reminiscent of Gaussian distributions with three distinct components, which correspond to the three age groups of the samples in the dataset. Furthermore, there are many histograms similar to Gaussian density. In general, the distribution of individual features appears similar for both classes. However, there are specific distributions that enable us to differentiate one class from another.

One potential preprocessing technique we can employ is Linear Discriminant Analysis (LDA). This step enables us to determine whether the features are linearly separable. Basically, it helps us to say whether a Gaussian model might outperform non-Gaussian models, taking into account both linear and non-linear models. LDA identifies C-1 directions in which to project our features, with C representing the number of classes. In our case, as we are dealing with binary classification, we obtain only one dimension.

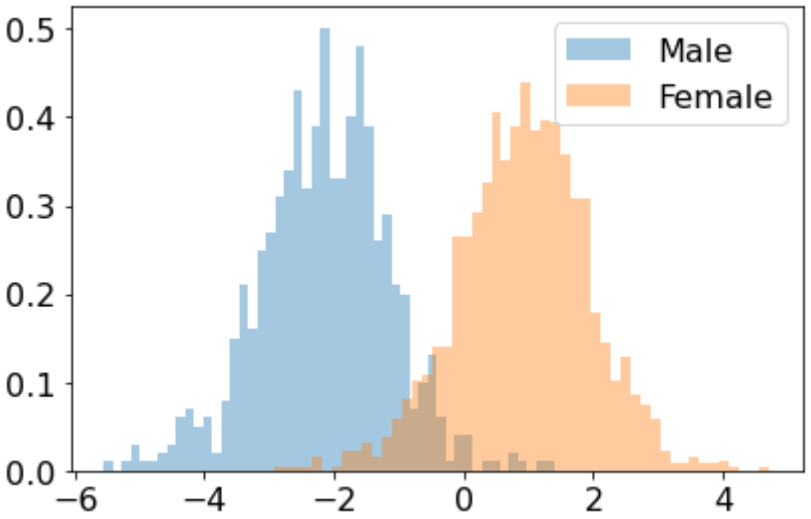

Fig. 2- Histogram showing the dataset features after applying LDA.

As illustrated in the plot (fig. 2), it's evident that the classes are generally distinguishable, although there exists a small region where errors can occur. When we examine the scatter plots for our dataset, we observe Gaussian-like distributions. This suggests that Gaussian models could yield satisfactory performance. It appears that there are multiple Gaussian distributions, each including several components or subsets. This concept is linked to the dataset containing three distinct age groups, which accounts for the observed phenomenon.

Moving our focus to the correlation analysis of our dataset, we will now examine the Pearson Correlation among our features.

Fig. 3 - Heatmaps showing Pearson correlation coefficient between features.

In Figure 3, in the whole dataset, most of the features exhibit a moderate level of correlation with each other, with correlation coefficients ranging around 0.5. We expect that Naive Bayes model won't be very good because there is a moderate level of correlation within the data. Furthermore we notice the presence of strong correlations between certain features, such as

1-10 and 3-6, while others show no significant correlation, like 2-5.This suggests that we could potentially benefit from reducing the dimensionality of our data, to reduce the number of parameters required for modeling and so simplify the analysis.

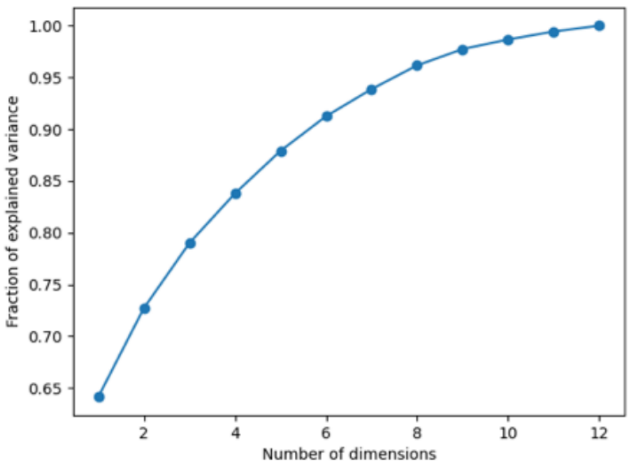

Fig. 4 - Explained Variance

Figure 4 illustrates that by eliminating 4 dimensions, we can still retain 95% of the variance, allowing us to discard 2 or 3 dimensions while preserving a substantial amount of information.

2. Classifying features

To conduct a thorough evaluation of different models and determine the most suitable one, we will employ various preprocessing techniques and implement a K-Fold cross-validation approach with K=5. This methodology is selected to ensure a substantial amount of data for both training and validation purposes. It's crucial to highlight that our dataset exhibits a high degree of imbalance, where the number of instances in different classes varies significantly. Our primary focus is on a specific application denoted as ($\tilde{\pi}$, Cfn, Cfp) = (0.5, 1, 1). However, we also intend to explore other application scenarios, specifically ($\tilde{\pi}$, Cfn, Cfp) = (0.1, 1, 1) and ($\tilde{\pi}$, Cfn, Cfp) = (0.9, 1, 1). To assess the performance of our models, we will utilize a metric known as normalized minimum detection costs. This metric takes into account the costs associated with different types of errors and offers a comprehensive evaluation of the model's efficacy in detecting the target class.
Through the application of these rigorous evaluation techniques and the consideration of various preprocessing methods, our goal is to identify the model that performs optimally. This model should effectively address the challenges posed by the imbalanced dataset and deliver accurate results across diverse application scenarios.

2.1 Gaussian classifiers

We start considering Gaussian classifiers (MVG classifier, MVG classifier with Naive Bayes assumption, MVG classifier with tied covariance) while exploring the effectiveness of different preprocessing techniques, such as Z-Score and PCA, as well as combinations of these methods.
We start considering Gaussian classifiers (MVG classifier, MVG classifier with Naive Bayes assumption, MVG classifier with tied covariance). All of them assume gaussian distributed data, given the class:

$$X|C = c \sim N(\mu c, \Sigma c)$$

Multivariate models perform better if we have enough data to reliably estimate the covariance matrices.
Tied covariance models can capture correlations, but may perform poorly when classes have very different distributions but this isn't our case.
Below, we present the results for various scenarios:

Gaussian classifiers with no PCA						
	RAW			Z-Score		
	π=0.5	π=0.1	π=0.9	π=0.5	π=0.1	π=0.9
MVG	0.122	0.306	0.339	0.122	0.306	0.339
NB	0.453	0.752	0.778	0.453	0.752	0.778
TMVG	0.114	0.300	0.321	0.114	0.300	0.321
TNB	0.451	0.768	0.782	0.451	0.768	0.782

Table 1. Gaussian Table for Raw and Z-Score results

Gaussian classifiers with PCA (12)						
	RAW			Z-Score		
	π=0.5	π=0.1	π=0.9	π=0.5	π=0.1	π=0.9
MVG	0.122	0.306	0.339	0.122	0.306	0.339
NB	0.127	0.319	0.341	0.124	0.319	0.335
TMVG	0.114	0.301	0.329	0.114	0.301	0.329
TNB	0.123	0.303	0.351	0.119	0.289	0.328

Table 2. Gaussian Table for Raw and Z-Score with Pca (12)

The tables presented indeed reflect our expectations. The Z-Score transformation attempts to normalize the data by centering it and scaling it by the standard deviation, resulting in outcomes similar to what we observed with the raw features. The Naïve Bayes assumption concerning the features isn't as restrictive as we initially assumed. This is because, as indicated by the heatmaps, there exists a moderate correlation among the features. To mitigate this issue, we attempted to project our samples into a space where we retain the highest variance. These are the directions we found using PCA, which corresponds to diagonalizing the covariance matrix. The results using PCA (12) show improvements over models employing the Naïve Bayes assumption. For these types of models, it's crucial that the features are uncorrelated, and PCA provides orthogonal directions in which to plot our data. Since orthogonality implies independence, our projected data becomes more independent.

Another noteworthy observation is that certain models maintain their performance because PCA conserves the primary discriminant information in the leading directions. As a result, MVG and TMVG do not lose critical information, leading to consistent performances.

In our task the covariance matrices of two classes are similar, so MVG and TMVG perform similarly. In this case the generic decision rule of Gaussian Model can be simplified to $s(x) = x^T b + c$. This is the reason behind the closely matched performance of the MVG and Tied model

Gaussian classifiers with PCA (11)						
	RAW			Z-Score		
	π=0.5	π=0.1	π=0.9	π=0.5	π=0.1	π=0.9
MVG	0.123	0.308	0.343	0.126	0.318	0.346
NB	0.136	0.311	0.373	0.127	0.310	0.333
TMVG	0.124	0.301	0.349	0.121	0.293.	0.343
TNB	0.126	0.302	0.377	0.125	0.282	0.359

Table 3. Gaussian Table for Raw and Z-Score with Pca (11)

Gaussian classifiers with PCA (10)						
	RAW			Z-Score		
	π=0.5	π=0.1	π=0.9	π=0.5	π=0.1	π=0.9
MVG	0.169	0.432	0.478	0.193	0.414	0.531
NB	0.173	0.448	0.481	0.185	0.431	0.552
TMVG	0.169	0.398	0.465	0.185	0.422	0.537
TNB	0.175	0.375	0.462	0.183	0.398	0.539

Table 4. Gaussian Table for Raw and Z-Score with Pca (10)

Furthermore, when we attempt to reduce the dimensionality to 11, we retain a substantial portion of the explained variance, resulting in performances that remain consistent with the 12-dimensional space. However, if we attempt to reduce the dimensionality even further, we begin to lose crucial information, leading to deteriorating results. In light of these outcomes, we have made the decision to continue using both Z-Scored and Raw data, but we will no longer explore additional dimension reduction techniques such as PCA (10) and PCA (11) as they do not provide any discernible benefits.

2.2 Discriminative Models

Now, let's move our attention to discriminative models, specifically the Logistic Regression Model (LR) and the Quadratic Logistic Regression Model (QLR).
These models aim to estimate the posterior distribution directly using the sigmoid function. Since classes are unbalanced, we change the objective function that we need to minimize so that costs of the different classes are re-balanced.To compute the score, we employ the following formulation:
$s(x) = w^Tx + b$
The model parameters are (w, b) and here's what each parameter represents:
"w^T" stands for the orthogonal vector with respect to the hyperplane we've defined.
"b" represents the bias term, indicating how far we need to shift horizontally from the hyperplane.
Our objective is to estimate the parameters "w" and "b" to formulate our decision rule.
It's important to note that LR doesn't assume any specific distribution of the data; it simply seeks to find a hyperplane that maximizes the posterior probability.

2.2.1 Logistic Regression

Another critical aspect is the inclusion of a regularization term in the objective function, designed to prevent issues when the classes are linearly separable. We estimate the optimal value for λ during training. The chosen interval to find the best value ranges from 10^{-5} to 10^{5} in a logarithmic fashion, with the initial setting of $\pi_T = 0.5$, which aligns with our primary task.

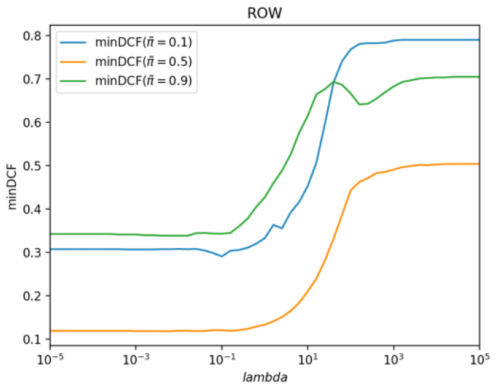

Fig. 5 - Plot LR with π_T = 0.5 and RAW features

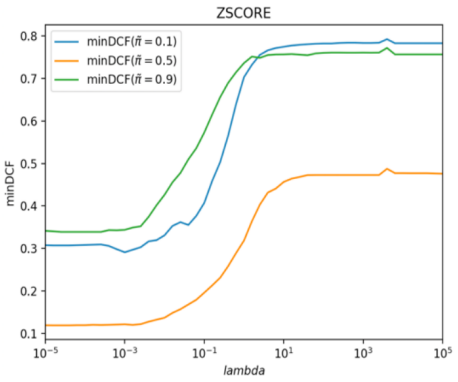

Fig. 6 - Plot LR with π_T = 0.5 and Z-Score features

As evident from these plots, the minimum value for the hyperparameter λ is achieved when λ=0. Therefore, we will adopt this value as the optimal choice for λ. We will not present the plot and results for PCA (12) as it closely resemble those for the Raw data.
To evaluate the model's performance, we will refer to the following table:

Linear Regression						
	RAW			Z-Score		
	π=0.5	π=0.1	π=0.9	π=0.5	π=0.1	π=0.9
LR (π_T = 0.5)	0.121	0.304	0.348	0.121	0.304	0.348
LR (π_T = 0.9)	0.116	0.332	0.318	0.116	0.332	0.318
LR (π_T = 0.1)	0.122	0.291	0.359	0.122	0.291	0.359

Table 5. LR Table for Raw and Z-Score with λ = 0

When the value of λ (lambda) is excessively large, we observe that the model has difficulty classifying the samples. This occurs because an increased λ makes the model too simple, where the model becomes overly simplistic. So, it might not be able to understand all the little details in the data, leading to poor classification performance.
However, for small values of λ, the model tends to achieve a good separation on the training data and can generalize effectively to unseen data. In such cases, the regularization term has minimal or no impact on enhancing the model's performance. Hence, it is more advantageous to present the results of the non-regularized version of the model.
To strike an appropriate balance between overfitting and underfitting, a suitable value for λ, typically in the order of 10^{-6}, can be selected. This value minimizes overfitting while still providing comparable performance to smaller λ values. By choosing this value, we can mitigate the risk of overfitting the model to the training data and make it better at working with new data it hasn't seen before.
Since Z-Score is a linear transformation, the performance obtained on the raw data and the Z-Scored data remains the same. Varying the values of T doesn't significantly improve the

model. There's a slight improvement when using T = 0.9, but this is primarily due to the dataset's imbalance towards class 1, the female class.

2.2.2 Quadratic Logistic Regression

Now, let's move our focus to the Quadratic Logistic Regression (QLR) version of the model. Similar to LR, we will consider the same interval for λ. Just as before, we will only present plots for Raw data and Z-Scored data, as the results with PCA (12) are similar.
Unlike LR, the QLR model exhibits poorer performance, as expected. This aligns with our earlier observations from scatter plots, histograms, and LDA analysis, which indicated that quadratic models may perform less effectively than linear ones. Data's distribution is not easily distinguishable using quadratic rules, but it can be well separated using linear decision rules.

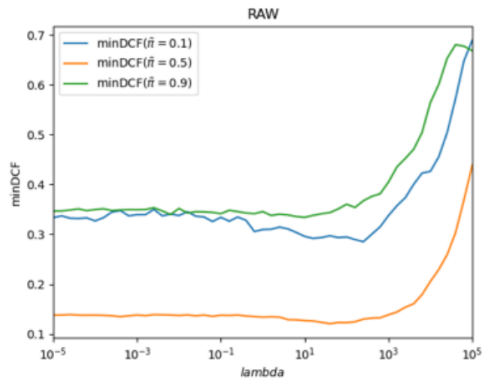

Figure 7. Plot QLR with π_T = 0.5 and RAW features

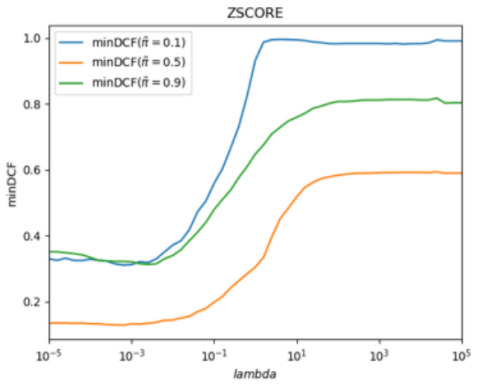

Figure 8. Plot QLR with π_T = 0.5 and Z-Score features

In contrast to the linear model, the results for the Quadratic Logistic Regression (QLR) model differ significantly between Raw features and Z-Scored features. This disparity arises because we performed a feature expansion operation that involves computing the dot product within another embedding space. Consequently, the model becomes sensitive to data transformations. For the Raw Features, we have chosen a value of λ equal to 10^2, while for the Z-Scored features, we opted for λ = 0, applying the same reasoning as with Logistic Regression.
As anticipated, the QLR model shows poorer performance compared to LR:

Quadratic Logistic Regression						
	RAW($\lambda = 10^2$)			Z-Score($\lambda = 0$)		
	π=0.5	π=0.1	π=0.9	π=0.5	π=0.1	π=0.9
QLR (π_T = 0.5)	0.124	0.299	0.351	0.133	0.325	0.347
QLR (π_T = 0.9)	0.131	0.371	0.344	0.147	0.354	0.372
QLR (π_T = 0.1)	0.125	0.281	0.406	0.141	0.351	0.366

Table 6. QLR Table for Raw and Z-Score with λ = 10^2 and λ = 0

2.3 Support Vector Machine (SVM)

2.3.1 Linear SVM

We now turn our attention to SVMs. We will consider the linear SVM, the polynomial quadratic kernel and the Radial Basis Function kernel formulations.
When comparing Logistic Regression (LR) to Support Vector Machine (SVM), SVM takes a different approach in finding the best way to separate classes. Unlike LR, SVM looks for a hyperplane that maximizes the margin, the maximum margin hyperplane.
We start considering a linear model that does not balance the two classes. To solve the SVM problem, we can consider the dual formulation:

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

We also have a parameter called "C" that we need to calibrate. We'll consider values between 10^-5 and 10^5 in a log scale. In general, we expect that a linear decision rule will work better than a quadratic one for class separation. C represents a trade-off: when it's very large, we try to minimize training errors, but this can lead to overfitting. On the other hand, when C is very small, we prioritize having a wide margin even if it means not fitting the training data perfectly.

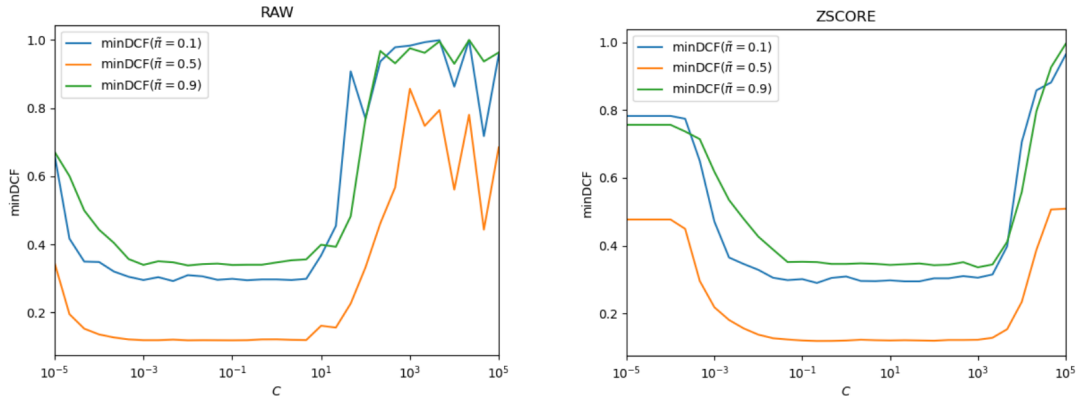

Figure 9. Plot Linear SVM with π_T = 0.5 (RAW feature on the left, Z-Score features on the right)

We have decided to use C = 10 because this hyperparameter yields the lowest minDCF value and it works well for different applications.

Linear SVM						
	RAW($C = 10$)			Z-Score($C = 10$)		
	π=0.5	π=0.1	π=0.9	π=0.5	π=0.1	π=0.9
SVM (π_T = 0.5)	0.135	0.282	0.363	0.120	0.288	0.356
SVM (π_T = 0.9)	0.121	0.333	0.367	0.115	0.349	0.324
SVM (π_T = 0.1)	0.124	0.306	0.379	0.127	0.307	0.374

Table 7. SVM Table for Raw and Z-Score with C = 10

According to the table, the results are mostly good and similar to the LR model, with just a small decrease in performance when π_T = 0.5. This matches what we expected about using a simple rule for making decisions. What's surprising is that the model works even better when π_T = 0.1. Also, there's a difference in how well the model works when we use RAW data compared to Z-Scored data. Z-Scored data gives us better results. This happens because Z-Scored data is adjusted to have a certain average and spread, which helps. Overall, it's clear from the results that this model is one of the best for classifying our sample data. So, we'll pay special attention to the SVM model with π_T = 0.9.

2.3.2 Non-linear SVM

We now analyze the non-linear formulations.
In SVM, the non-linearity is obtained through an implicit expansion of the features in a higher dimensional space. The dual SVM formulation depends on the training samples through the dot product and it's possible to compute scores through scalar products between training and evaluation samples. For this reason, it's not required to explicitly compute the feature expansion, it's enough to be able to compute the scalar product between the expanded features, the so called kernel function k:

$$k(x_i, x_j) = \phi(x_i)^T \phi(x_j)$$

We will employ two different kernels:

1. Polynomial kernel of degree d = 2: $k(x_i, x_j) = (x_i^T x_j + c)^d$

2. Radial Basis Function (RBF) kernel: $k(x_i, x_j) = e^{-\gamma ||x_i - x_j||^2}$

2.3.3 Polynomial SVM

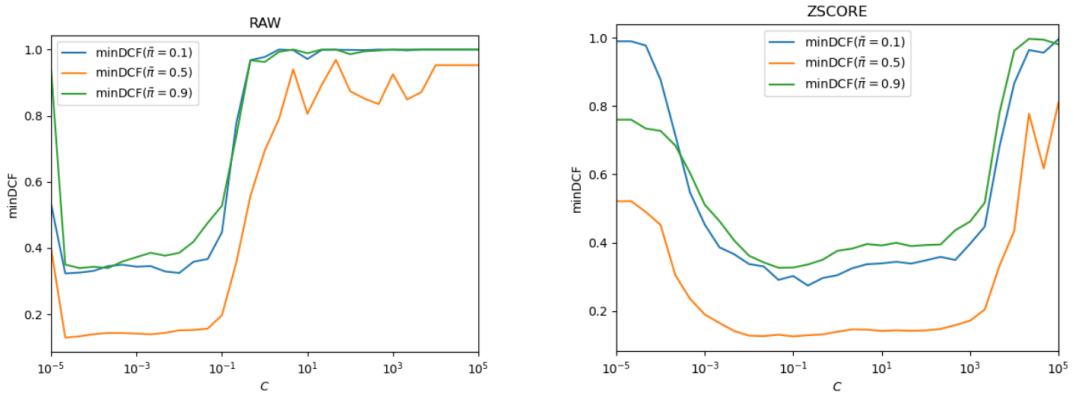

Figure 10. Plot polynomial SVM with π_T = 0.5 (RAW feature on the left, Z-Score features on the right)

Also here, the choice of C is important and, based on the plots above, we have decided to choose C = 10^3 for Raw Features while C = 10^{-1} for Z-Score Feature.

Polynomial SVM						
	RAW (C = 10^3)			Z-Score (C = 10^{-1})		
	π=0.5	π=0.1	π=0.9	π=0.5	π=0.1	π=0.9
QSVM (π_T = 0.5)	0.140	0.347	0.372	0.126	0.301	0.323
QSVM (π_T = 0.9)	0.155	0.358	0.357	0.136	0.384	0.302
QSVM (π_T = 0.1)	0.157	0.381	0.439	0.151	0.341	0.449

Table 8. Quadratic SVM Table for Raw and Z-Score

2.3.4 RBF SVM

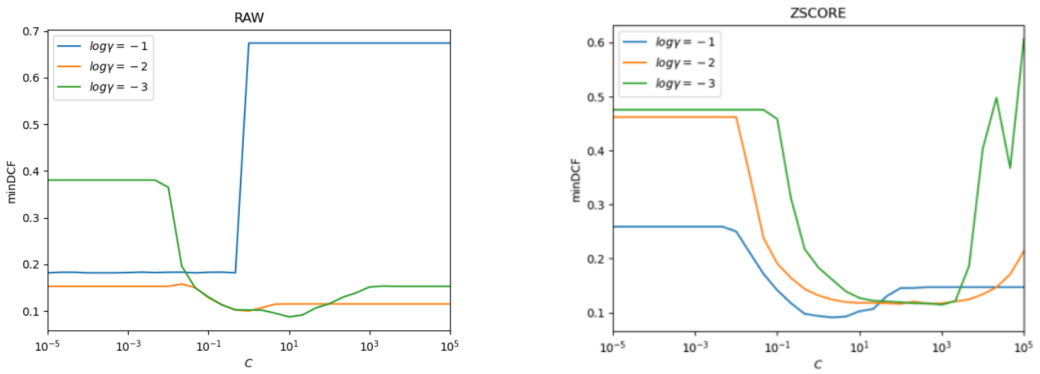

Figure 11. Plot RBF SVM with π_T = 0.5 (RAW feature on the left, Z-Score features on the right)

For the radial version based, the performances are good with γ = 0.001 and C = 10, in the case of Raw feature, and γ = 0.1 and C = 5, in the case of Z-Scored features

RBF SVM						
	RAW (γ = 0.001, C = 10)			Z-Score (γ = 0.1, C = 5)		
	π=0.5	π=0.1	π=0.9	π=0.5	π=0.1	π=0.9
RBSVM (π_T = 0.5)	0.136	0.289	0.361	0.121	0.302	0.337
RBSVM (π_T = 0.9)	0.122	0.331	0.358	0.115	0.329	0.335
RBSVM (π_T = 0.1)	0.125	0.303	0.375	0.126	0.308	0.362

Table 9. Radial Basis Function (RBF) kernel SVM Table for Raw and Z-Score

Quadratic SVM obtains worse performance than Quadratic LR, which are not good as the linear version of the two mentioned models. However, a Radial Basis kernel SVM can act as a good model respect to the Quadratic one and Quadratic LR (just the Z-Score feature for our main application).

2.4 GMM

The next type of classifier we're going to discuss is the Gaussian Mixture Model (GMM). Typically, GMMs are used when we want to estimate the underlying data distribution, assuming that our data can be thought of as a combination of one or more Gaussian distributions.
Based on our earlier results with Gaussian models and our analysis of the dataset, it seems that GMM and Tied GMM models have the potential to perform very well.
We expect to get the best results with Gaussian Mixture Models that have either 2 or 4 components. This expectation arises from our examination of the dataset, which appeared to have a distribution resembling a Gaussian with 3 components or clusters. Since GMMs are typically trained with a number of clusters that's a power of 2, we won't consider GMMs with 3 components.

Our analysis will encompass four variations: the full covariance model, the diagonal model, the model with covariance tying and the Naive Bayes tied covariance. In the case of the tied covariance model, covariance tying occurs at the level of each class, resulting in distinct covariance matrices for different classes.

When evaluating various models, it was observed that the performance deteriorates as the number of components exceeds 16. Additionally, single-fold analysis yielded worse results, prompting us to solely consider K-fold evaluations for a more reliable assessment. It is essential to conduct K-fold cross-validation to obtain a more accurate and representative evaluation of the models' performance, especially when dealing with a limited dataset.

GMM						
	RAW			Z-Score		
	π=0.5	π=0.1	π=0.9	π=0.5	π=0.1	π=0.9
2 components	0.080	0.244	0.211	0.080	0.244	0.211
4 components	0.076	0.228	0.206	0.076	0.228	0.206
8 components	0.100	0.258	0.233	0.097	0.263	0.250
16 components	0.152	0.429	0.301	0.144	0.387	0.319

Table 10. GMM Table for Raw and Z-Score up to 16 components

Naive Bayes GMM						
	RAW			Z-Score		
	π=0.5	π=0.1	π=0.9	π=0.5	π=0.1	π=0.9
2 components	0.208	0.514	0.519	0.208	0.514	0.519
4 components	0.193	0.441	0.487	0.177	0.562	0.464
8 components	0.202	0.488	0.491	0.205	0.474	0.509
16 components	0.202	0.504	0.534	0.210	0.499	0.531

Table 11. Naive Bayes GMM Table for Raw and Z-Score up to 16 components

Tied GMM						
	RAW			Z-Score		
	π=0.5	π=0.1	π=0.9	π=0.5	π=0.1	π=0.9
2 components	0.121	0.308	0.347	0.121	0.308	0.347
4 components	0.078	0.240	0.212	0.078	0.265	0.228
8 components	0.078	0.255	0.203	0.073	0.247	0.196
16 components	0.080	0.254	0.212	0.080	0.257	0.222

Table 12. Tied GMM Table for Raw and Z-Score up to 16 components

Tied Naive Bayes GMM						
	RAW			Z-Score		
	π=0.5	π=0.1	π=0.9	π=0.5	π=0.1	π=0.9
2 components	0.213	0.498	0.521	0.213	0.498	0.521
4 components	0.194	0.456	0.487	0.190	0.456	0.474
8 components	0.204	0.515	0.500	0.191	0.485	0.504
16 components	0.184	0.486	0.470	0.179	0.525	0.472

Table 13. Tied Naive Bayes GMM Table for Raw and Z-Score up to 16 components

As we expected, Tied GMM and, in particular, GMM have the best results. The two diagonal models don't perform well because we have a moderate correlation as we have seen in the heatmaps. In the final analysis phase, where the Test set will be employed, it is expected that models utilizing GMM with a high number of components will experience a decline in performance, so those GMM classifiers will be discarded.

3. Score Calibration

Up until now, we have only focused on the minimum DCF metric. The best classifiers at this point are, from each model:

	minDCF		
Model	π=0.5	π=0.1	π=0.9
Tied MVG (Z-Score)	0.114	0.301	0.329
LR (π_T =0.9, l=0; Z-Score)	0.116	0.332	0.318
RBF SVM (π_T =0.9, C=5, g=0.1; Z-Score)	0.115	0.329	0.335
GMM (4 components; Z-Score)	0.076	0.228	0.206
SVM (π_T =0.9, C=10; Z-Score)	0.115	0.349	0.324

Table 14. Best results of Classifiers on Training Set

Considering the significant difference between the GMM model and the other one in all three applications under consideration, we have designated the GMM model as the primary candidate for the evaluation phase. However, it is essential to ascertain whether the scores of this model require calibration or not.
The minimum DCF measures the expense we would face if we made the best decisions for the validation set using the scores of the recognizer. However, the actual cost we incur depends on how well we choose the threshold for class assignment. Therefore, let's now consider the actual DCFs. We'll use Logistic Regression for this purpose because it gives us a log-likelihood ratio, allowing us to calculate the calibrated score by subtracting the theoretical threshold.

First, we aim to improve the calibration of all models by employing score calibration techniques. The general approach involves computing a transformation function, denoted as f(s), which maps the scores.We assume that function f is linear in s:

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

This function 'f(s)' can be interpreted as the log-likelihood ratio for the two class hypotheses, denoted as 'HT' and 'HF':

$$f(s) = log\frac{f_{S|C}(s \mid HT)}{f_{S|C}(s \mid HF)} = \alpha s + \beta$$

so the class posterior probability for a prior π' can be written as:

$$f(s) = log\frac{P(C=HT \mid s)}{P(C=HF \mid s)} = \alpha s + \beta + log(\frac{\pi'}{1-\pi'})$$

We can interpret the scores as features so that this expression will be similar to the log posterior ratio of the Logistic Regression model. If we rewrite:

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

By specifying a certain prior, we effectively optimize the calibration for this specific application. Nevertheless, this approach provides good calibration for different applications as well.
Now, let's evaluate the actual DCF to determine how well the models would perform if we were using the theoretical threshold for each application, assuming that the scores are already well-calibrated. To estimate the parameters of the calibration function, we'll use a method called K-Fold Approach, since our sample size is limited.
Among the models we've explored, the GMM Full-Tied Covariance model on Z-Scored data seems to be the most promising one, so we'll adjust its scores for calibration. This analysis can also be verified through Bayes error plots, which show the DCFs for different applications.

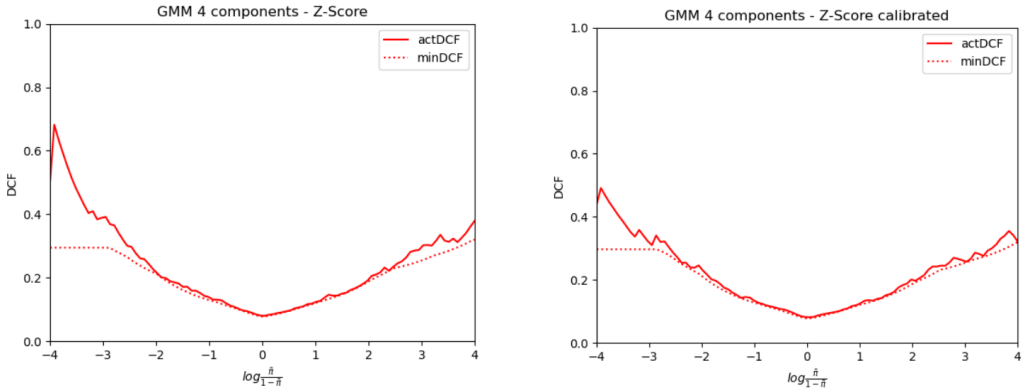

Figure 12. On the left GMM uncalibrated Model, on the right GMM calibrated model

Interestingly, some scores don't need adjustment. The transformation doesn't seem to significantly benefit the model. There's a slight improvement only in the unbalanced application with $\tilde{\pi}$ = 0.9. Given that calibrated and uncalibrated scores show similar performance, for simplicity, we'll use the uncalibrated model as the final classifier for GMM.

	actDCF		
Model	π=0.5	π=0.1	π=0.9
uncalibrated GMM (4 components; Z-Score)	0.080	0.257	0.212
calibrated GMM (4 components; Z-Score)	0.081	0.233	0.219

Table 15. actDCF for calibrated and uncalibrated GMM 4 components

Furthermore, we'll also consider the other models and explore the option of calibrating them. Specifically, Logistic Regression (LR), RBSVM, and SVM have uncalibrated scores, and calibration could potentially improve their performance. We RBSVM shows better results in our primary application after calibration.

Uncalibrated models						
	minDCF			actDCF		
Model	π=0.5	π=0.1	π=0.9	π=0.5	π=0.1	π=0.9
SVM	0.115	0.349	0.324	0.231	0.374	0.703
RBF SVM	0.115	0.329	0.335	0.161	0.646	0.907
LR	0.116	0.332	0.318	0.239	0.591	0.544

Table 16. minDCF and actDCF for uncalibrated models

Calibrated models						
	minDCF			actDCF		
Model	π=0.5	π=0.1	π=0.9	π=0.5	π=0.1	π=0.9
SVM	0.116	0.343	0.334	0.117	0.363	0.353
RBF SVM	0.098	0.324	0.247	0.107	0.329	0.256
LR	0.118	0.338	0.325	0.120	0.361	0.356

Table 17. minDCF and actDCF for calibrated models

4. Fusion

The approach we've used for calibration enables us to merge the various outputs from different models. We can also experiment with combining different classifiers to merge their distinct decisions and enhance overall performance. We assume that the fused score is a linear function of the scores of the different classifiers:

$$s_t = f(s_{t,A}, s_{t,B})$$

Where st is the fused score for sample xt, while st,A and st,B are the scores of that sample generated by the classifier A and B.

In this case, we aim to fuse the three best models we've selected: LR, SVM, and GMM. The fusion method we can employ is similar to calibration; we'll apply the K-Fold Approach and attempt to combine the decisions using prior-weighted logistic regression.

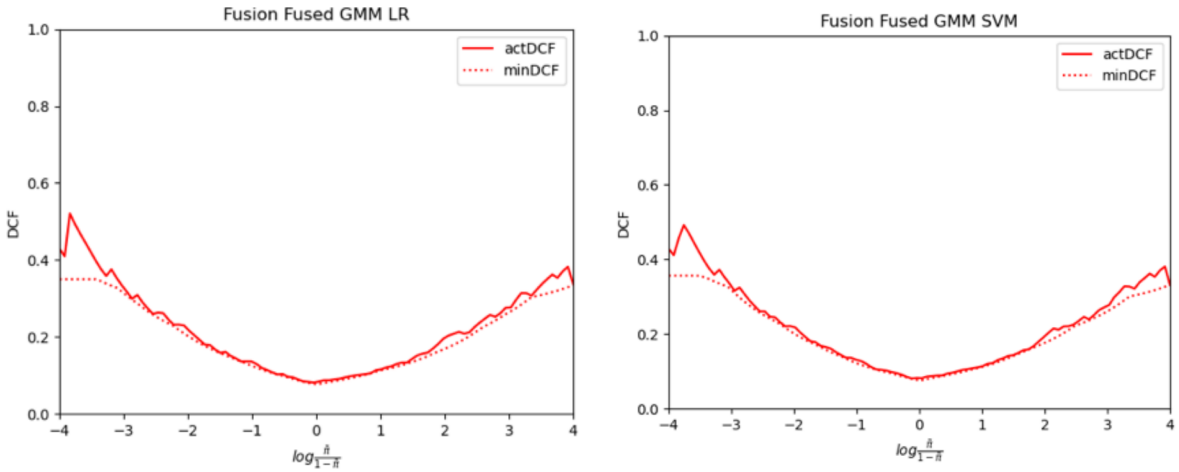

Figure 13. On the left Fusion GMM+LR, on the right Fusion GMM+SVM

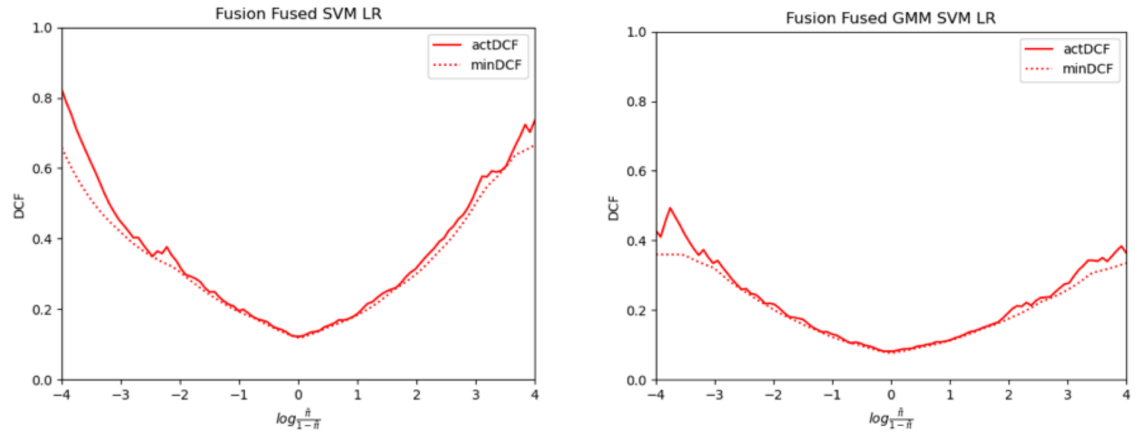

Figure 14. On the left Fusion LR+SVM, on the right Fusion GMM+SVM+LR

	minDCF		
Model	π=0.5	π=0.1	π=0.9
GMM+LR	0.076	0.222	0.189
GMM+SVM	0.076	0.227	0.184
LR+SVM	0.115	0.331	0.334
GMM+LR+SVM	0.075	0.225	0.191

Table 18. minDCF for Fusion models

As evident from the plots, all the scores are already calibrated, eliminating the need for any additional calibration steps. In table 17, we assess the results of these models in terms of minCDF and the top-performing model for our primary application with π = 0.5 is the combination of GMM + SVM. Additionally, GMM + SVM + LR and GMM + LR also show promising results. Since we're using two types of linear decision boundaries, LR and SVM (both trained with π = 0.9), they essentially find the same hyperplane to separate the data. Consequently, fusing these two models brings only minor changes.
Furthermore, we observe improvements for π = 0.1 and π = 0.9, suggesting that the GMM model assists the other models in reducing errors. In fact, SVM + LR doesn't perform well in the other two types of applications.

5. Experimental results

Let's analyze the different models' performances on the test set in terms of minDCF and actDCF values. Unlike the training phase, we only focus on the models that performed the best in the previous phase. Additionally, we exclude the Gaussian models since we have already incorporated them into the Gaussian Mixture Models as a single component Gaussian.

We start to analyze single model with the best hyper parameters founded during the training phase:

	minDCF			actDCF		
Model	π=0.5	π=0.1	π=0.9	π=0.5	π=0.1	π=0.9
TMVG(Z-Score)	0.117	0.299	0.305	0.118	0.306	0.321
LR (π_T =0.9, λ=0; Z-Score)	0.121	0.342	0.282	0.215	0.641	0.503
SVM (π_T =0.9, C=10; Z-Score)	0.130	0.358	0.278	0.205	0.361	0.689
RBF SVM(π_T =0.9, C=5, g=0.1; Z-Score)	0.097	0.332	0.201	0.145	0.571	0.894
GMM (4 components; Z-Score)	0.061	0.152	0.155	0.062	0.161	0.163

Table 19. minDCF for Fusion models

The performance results have experienced notable fluctuations, but the GMM with Full Covariance remains the best-performing model, demonstrating a significant performance lead across all applications. The TMVG reinforces this performance pattern, while RBSVM outperforms its performance during training. However, the other two models, LR and SVM, exhibit decreased performance compared to the training phase. Consequently, we will now assess a DET plot for GMM, SVM, and RBSVM:

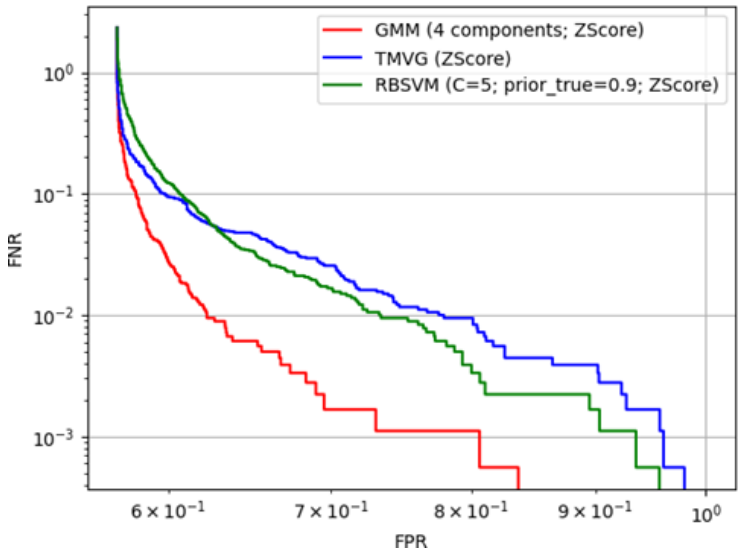

Figure 15. Det Plot of TMVG, RBF SVM and GMM

Furthermore, as evident from the table, the Gaussian models do not require any calibration. In contrast, LR, RBSVM, and SVM need to be calibrated. Our first step will be to examine the Bayes error plot to observe the differences, and then we will analyze the same plot for calibrated scores.

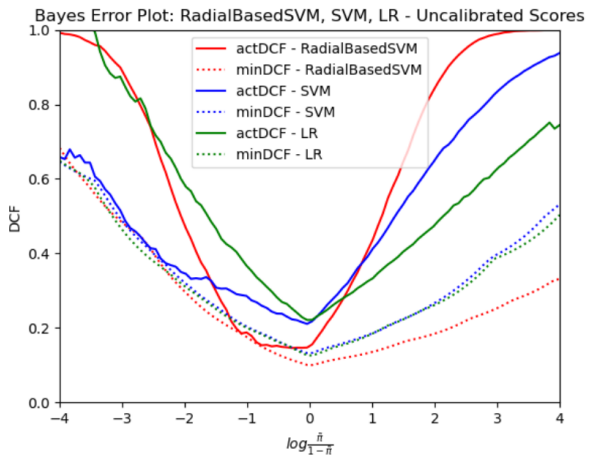

Figure 16. Uncalibrated Scores for RBF SVM, SVM and LR

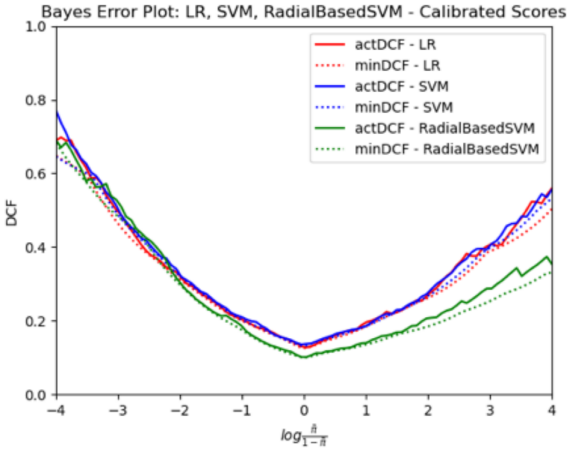

Figure 17. Calibrated Scores for RBF SVM, SVM and LR

The scores are now more effectively calibrated, evident from both the plots and tables. However, the GMM Full Covariance with 4 components remains the top-performing model.

Uncalibrated models						
	minDCF			actDCF		
Model	π=0.5	π=0.1	π=0.9	π=0.5	π=0.1	π=0.9
LR (π_T =0.9, λ=0; Z-Score)	0.124	0.337	0.282	0.217	0.643	0.505
RBF SVM(π_T =0.9, C=5, g=0.1; Z-Score)	0.098	0.329	0.197	0.149	0.563	0.899
SVM (π_T =0.9, C=10; Z-Score)	0.129	0.346	0.291	0.208	0.360	0.693

Table 20. minDCF and actDCF for uncalibrated models

Calibrated models						
	minDCF			actDCF		
Model	π=0.5	π=0.1	π=0.9	π=0.5	π=0.1	π=0.9
LR (π_T =0.9, λ=0; Z-Score)	0.124	0.337	0.282	0.126	0.345	0.302
RBF SVM(π_T =0.9, C=5, g=0.1; Z-Score)	0.098	0.329	0.197	0.101	0.352	0.216
SVM (π_T =0.9, C=10; Z-Score)	0.129	0.346	0.291	0.132	0.361	0.309

Table 21. minDCF and actDCF for calibrated models

We now shift our focus to the fused and calibrated models discussed earlier and assess their performance on the evaluation data. To recap, the GMM model employs full covariance with 4 components, the LR model utilizes logistic regression trained with π_T=0.9, and the SVM model is trained with π_T =0.9. All of these models have achieved their results through Z-Score preprocessing of the data.
It's worth noting that we opted not to fuse the TMVG model, as it exhibits Gaussian-like characteristics. Therefore, we've chosen to represent only one model from each family for evaluation.

	minDCF		
Model	π=0.5	π=0.1	π=0.9
GMM+LR	0.060	0.152	0.151
GMM+SVM	0.061	0.154	0.151
LR+SVM	0.123	0.334	0.284
GMM+LR+SVM	0.060	0.153	0.150

Table 22. Results of Fusion models on evaluation set

The outcomes are positive, suggesting that the GMM model improves the performance of the other two models and reduces the overall cost. This phenomenon explains why the performances are similar in the main application and relatively consistent in the other two. Notably, when we examine the fusion of SVM+LR, their performances are less impressive compared to the other models. Next, we'll examine the Bayes error plot to assess whether any of the models require calibration.

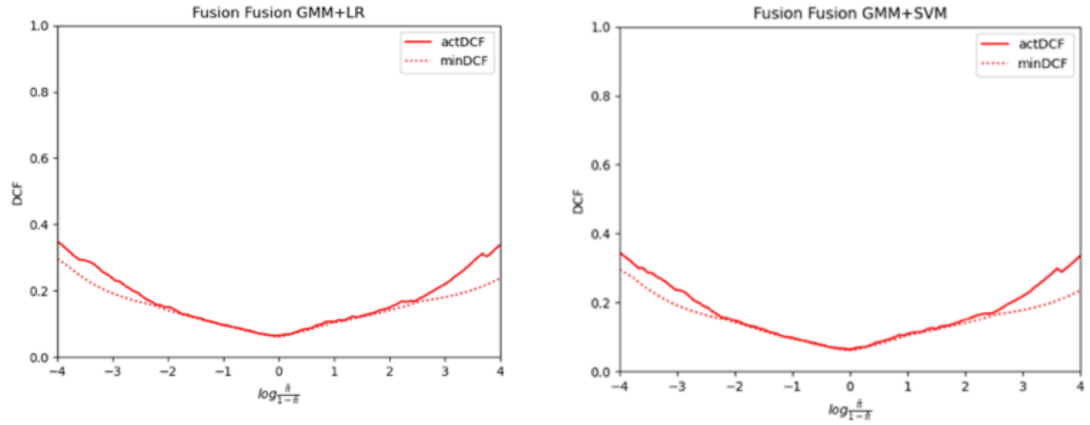

Figure 18. On the left Fusion GMM+LR, on the right Fusion GMM+SVM

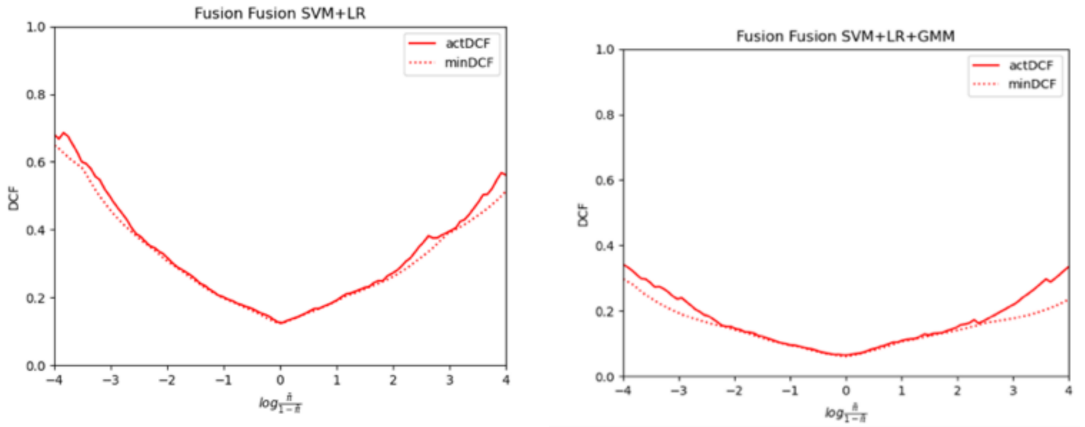

Figure 19. On the left Fusion LR+SVM, on the right Fusion GMM+SVM+LR

Looking the plots we understand that scores are already effectively calibrated, eliminating the need for any additional calibration preprocessing on the models:

Calibrated models						
	minDCF			actDCF		
Model	π=0.5	π=0.1	π=0.9	π=0.5	π=0.1	π=0.9
GMM+LR	0.060	0.152	0.151	0.063	0.158	0.165
GMM+SVM	0.061	0.154	0.151	0.064	0.156	0.158
LR+SVM	0.123	0.334	0.284	0.124	0.338	0.301
GMM+LR+SVM	0.060	0.153	0.150	0.064	0.156	0.159

Table 23. Results on Evaluation Set for Fused models

In the following section, we will showcase only the best model from each family of classifiers. Additionally, to focus on the best-performing models, we exclude quadratic classifiers like QLR and Quadratic SVM. As we observed during the training phase, quadratic models are not the most suitable options.

6. Evaluation

LR

We begin by examining the logistic regression models and plotting λ within the same interval as during the training phase. Our top-performing model during training was LR trained with π_t = 0.9, so we will focus on plotting this type of model (specifically Z-Scored, as the results are comparable with Raw features):

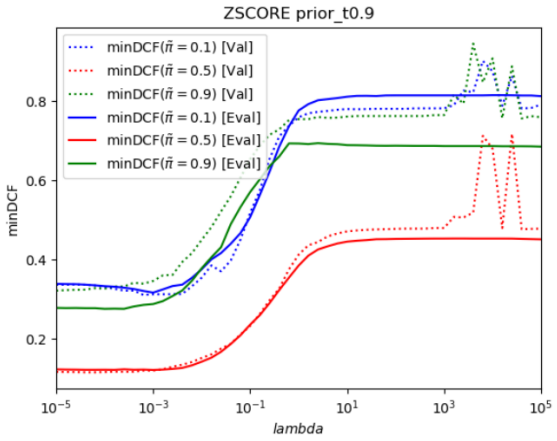

Figure 20. LR on evaluation set with π_t =0.9

The curves overlap until λ is less than 10^3, and the performances closely resemble what we observed during training, reaffirming that regularization doesn't offer any advantages. We won't present the results in a table because the curves are highly similar, essentially duplicating the previous table's outcomes. However, for QLR, we noticed slightly inferior performance compared to the linear version, so we won't proceed with an evaluation of this model type.

SVM

Next, we turn our attention to the non-probabilistic models like SVM and the Radial Based Version, as the quadratic model didn't yield favorable results. Specifically, for SVM, we focus solely on the data with Z-Score transformation, and the results align with our expectations:

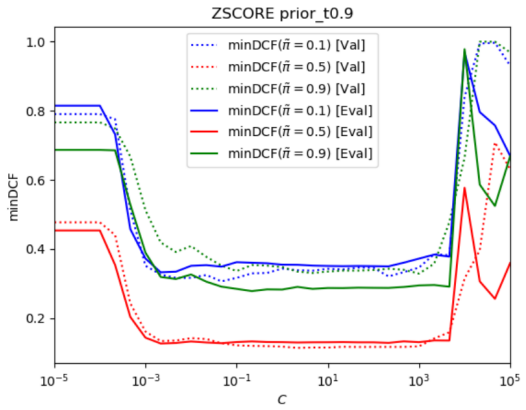

Figure 21. SVM on evaluation set with π_t =0.9

The optimal value selected for C remains unchanged, but we've observed slight improvements in performance, as mentioned earlier. Specifically, we've improved from 0.115 to 0.113.

RBV

We now turn our attention to the Radial Based Version:

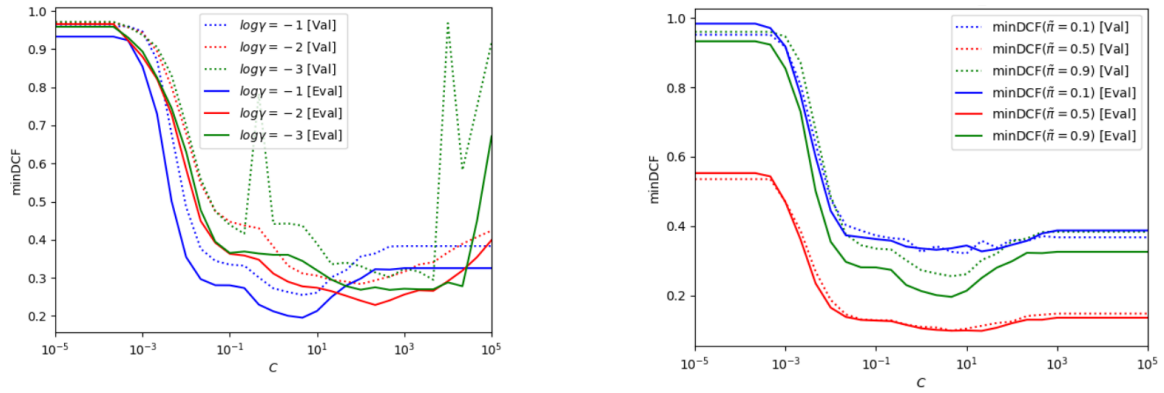

Figure 22. SVM on evaluation set with π_t =0.9

As evident from these plots, the value of 0.1 remains the optimal choice for γ, and the performances remain consistent when this γ value is set. While during training, we selected C=5, it's evident that 5 is not the optimal choice but rather a suboptimal one, albeit with slight differences.

GMM

Moving on to the Gaussian Mixture Models (GMM), we'll focus on evaluating only the best options within this model category. These include the GMM Full Covariance with Z-Score Features, GMM Tied Covariance with Z-Score Features, Diagonal GMM Full Covariance with PCA (12) + Z-Score, and Diagonal Tied GMM.

There are minimal differences, and the top-performing models in this scenario are still the GMM Tied and GMM with four components. To assess our choices during training, we will evaluate the performances of only these two classifiers.

	RAW			Z-Score		
	π=0.5	π=0.1	π=0.9	π=0.5	π=0.1	π=0.9
GMM(4 components)	0.061	0.157	0.157	0.060	0.152	0.157
Tied GMM(4 components)	0.059	0.177	0.160	0.058	0.177	0.161

Table 24. Best results GMM models

As observed, the chosen model may not be the optimal solution, but its performance remains relatively consistent. These results align with those obtained using fusion techniques, and it's worth noting that the scores are already well-calibrated.

Conclusion

The striking similarity between the validation and evaluation sets indicates that the evaluation data shares a comparable distribution with the training data. The strategies and choices implemented during the training phase have proven effective when applied to the test data.

The utilization of the Gaussian Mixture Model (GMM) with full covariance, operating on z-score normalized data with 4 components, has showcased its efficacy in producing finely-tuned scores across various application scenarios. In our primary application ($\tilde{\pi}$ = 0.5), we achieved an impressively low DCF (Detection Cost Function) cost of 0.06. Additionally, for sub-scenarios with $\tilde{\pi}$ values of 0.1 and 0.9, we attained commendable DCF costs of 0.152 and 0.157, respectively.