Politecnico di Torino

Master Degree in Artificial Intelligence and Data Analytics

Machine Learning and Pattern Recognition

Fingerprint Spoofing detection

February 15, 2024

Roberto Gaetano, s318989
Distefano Giuseppe, s309787

1 Introduction

1.1 Problem Overview

The main objective of the project involves creating a classifier capable of distinguishing between an authentic fingerprint or a spoofed fingerprint. The dataset comprises fingerprint images represented as embeddings, with each embedding described by 10 continuous variables. These embeddings are associated with either authentic fingerprints (labeled as 1) or spoofed fingerprints (labeled as 0), with the latter belonging to 6 distinct sub-classes representing various spoofing methods.

The dataset is split into two files: Train.txt (containing 800 authentic and 1525 spoofed fingerprints) and Test.txt (comprising 2400 authentic and 5304 spoofed fingerprints). Both files contain rows representing individual samples, where the first ten entries denote features, and the final entry indicates the class label (0 for spoofed and 1 for authentic fingerprints). The dataset exhibits class imbalance, with more samples in the spoofed fingerprint class.

The goal is to perform classification using diverse techniques, analyze outcomes, and justify the chosen methodologies. Notably, the dataset's imbalance warrants attention, as the spoofed fingerprint class is significantly larger than the authentic class.

Moreover, the target application requires equal prior probabilities for authentic and spoofed classes. However, misclassifying a spoofed fingerprint as authentic incurs a higher cost due to potential security implications. The application's working point is defined as ($\pi_T = 0.5$, $C_{fn} = 1$, $C_{fp} = 10$), where π_T represents the application prior, C_{fn} denotes the cost of false negatives, and C_{fp} the cost of false positives. The analysis will also explore the model's performance across different working points.

1.2 Feature analysis

Before going in depth into a detailed discussion of classification and validation strategies, it is useful to thoroughly explore and examine our dataset.

Figure 1 depicts histograms for each of the 10 features. Notably, certain features, like feature 1 and 3, exhibit a distribution reminiscent of a Gaussian bell curve.

Upon closer inspection of other plots, it becomes evident that some features distinctly differentiate between the two classes. Notably, feature 8 emerges as one of the most discriminant features in the dataset. Additionally, there are observations indicating that certain features could be better represented by multiple Gaussian bell curves, as seen in features 6, 7, and 9.

The histogram plots offer insights into the distribution of features and their relevance for classification. Features such as 2, 3, 4, and 8 demonstrate minimal overlap in their histograms, suggesting their potential usefulness for classification. These features provide valuable information for determining the class to which a fingerprint belongs. On the contrary, other features display significant histogram overlap, so these will not discriminate well our classes.

This preliminary exploration of our data sets the foundation for a comprehensive understanding of feature distributions and their potential impact on classification and validation strategies. For example, it is evident from the histograms that the features exhibit shapes resembling Gaussian distributions. Therefore, Gaussian models may prove to be effective for our intended purpose.

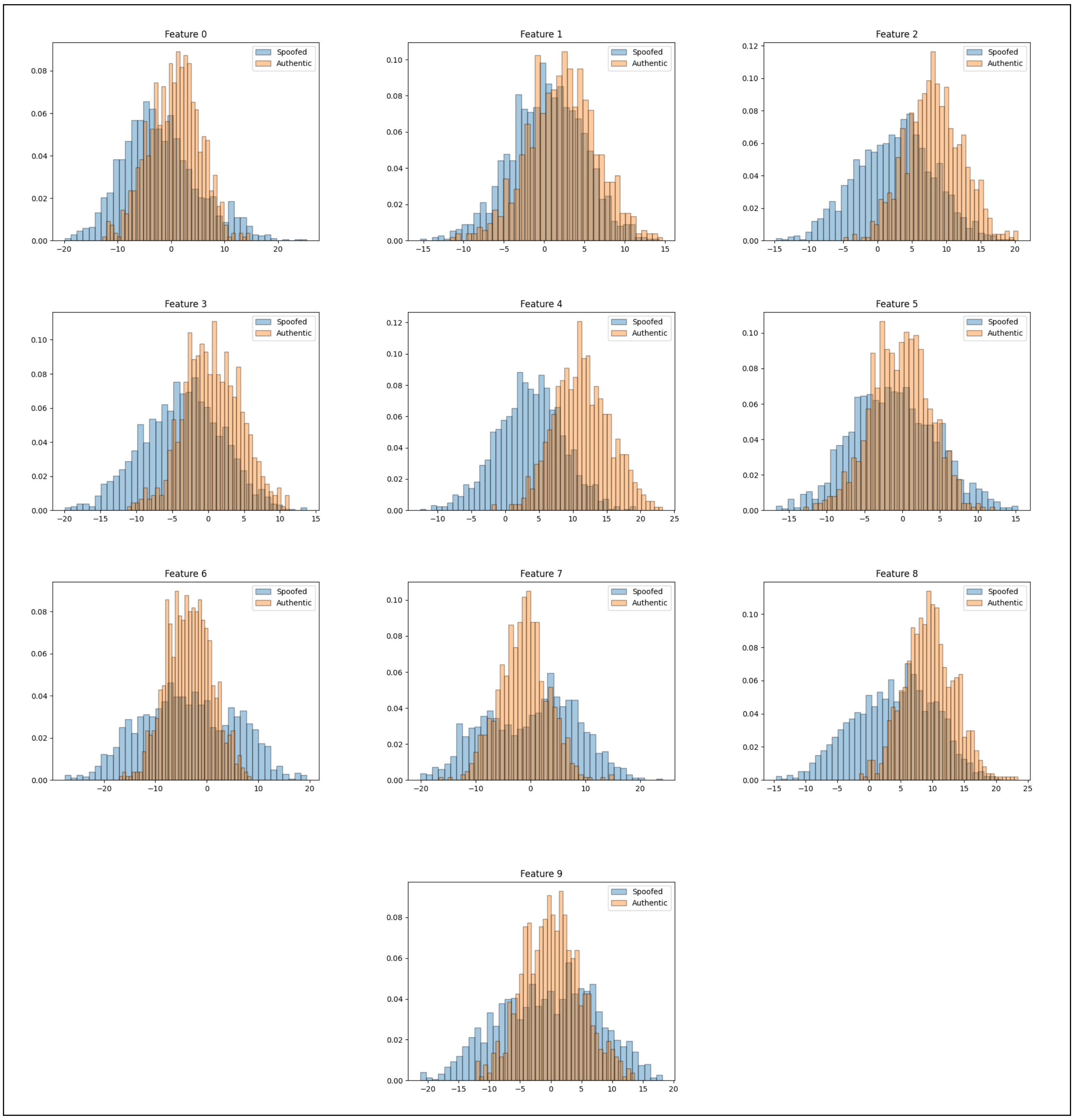
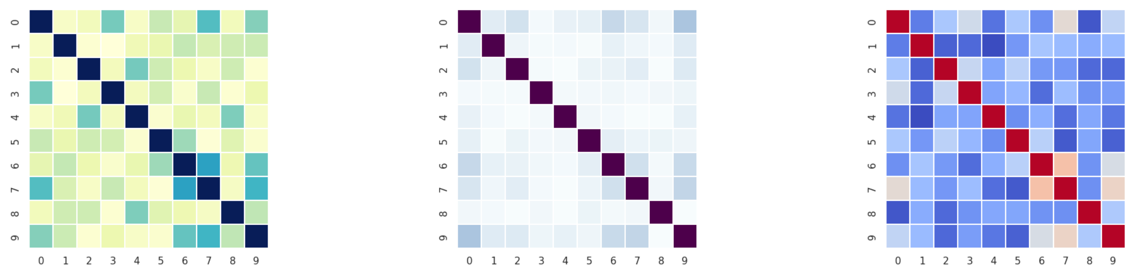


Figure 1: *Histograms of features*

Examining the correlation analysis we can draw insights by observing the Pearson Correlation represented as a heatmap, like in Figure 2. The first heatmap provides an overview of the correlation for each feature across the entire dataset, the others two denote correlation with the target and non-target classes. Despite slight variations across heatmaps, a notable finding is that features are generally not strongly correlated, except for one or two features. This suggests that eliminating some dimension may not result in a substantial loss of information.

With the Pearson correlation coefficient defined as:

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



Heatmap 1: All samples *Heatmap 2: Spoofed fingerprint samples* *Heatmap 3: Authentic fingerprint samples*

Figure 2: *Heatmaps*

Exploring the application of Linear Discriminant Analysis (LDA) to enhance class separation, we aim to determine the linear separability of our binary classification problem. LDA enables the estimation of at most C-1 directions, where C represents the number of classes. Examining the resulting plot in Figure 3, it appears that a linear classifier could perform effectively as the classes are reasonably well-separated, with a small overlapping area. Thus, LDA suggests that a linear classifier may adequately discriminate between classes.

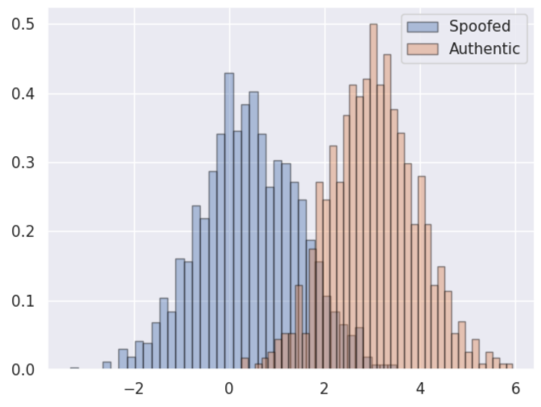


Figure 3: *LDA*

Utilizing PCA and focusing on the two most relevant dimensions by means of scatter plot in Figure 4, it becomes evident that the classes may not exhibit linear separability, suggesting that quadratic decision surfaces might be more appropriate for classification. In addition, we can see that the non-target (Spoofed fingerprint) class is composed of several clusters, perhaps representing different spoofing approaches.

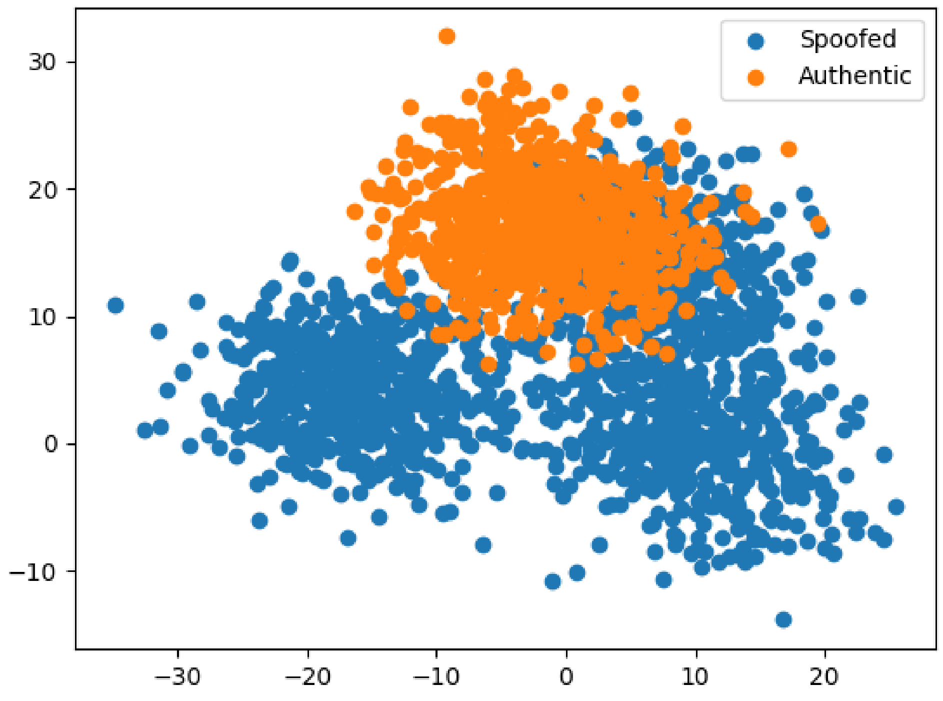


Figure 4: *Scatter plot of all features after PCA with m = 2 dimensions*

1.3 Training

The analysis focuses on robust evaluation methodologies, diverse application scenarios, model selection criteria, and the crucial consideration of imbalanced priors. The exploration of preprocessing techniques enhances our understanding of their relevance in the fingerprint recognition task. The comprehensive approach ensures a thorough evaluation of classifiers under various conditions, contributing to informed decision-making for the final model.

Expanding on our analysis, we delve deeper into methodologies and relative considerations involved in training and evaluating classifiers for the fingerprint recognition task.

We adopted a K-Fold approach with K=5, shuffling the data before splitting to create homogeneous folds. This technique allows us to efficiently handle the limited dataset for both training and validation, providing a robust evaluation framework. Then, for performance evaluation, we used the normalized minimum detection cost function (*minDCF*). This metric represents the cost associated with optimal decision-making, considering the effective prior that accounts for both the prior probability of the authentic fingerprint class and the cost of errors. The effective prior adjusts for the fact that the cost of False Positives (C_{fp}) is greater than the cost of False Negatives (C_{fn}).

Application scenarios are basically three:

- Uniform prior application

$$(\pi_T = 0.5, C_{fn} = 1, C_{fp} = 10)$$

- Unbalanced applications with priors biased towards one class:

$$(\pi_T = 0.1, C_{fn} = 1, C_{fp} = 10)$$
$$(\pi_T = 0.9, C_{fn} = 1, C_{fp} = 10)$$

These scenarios help assess the impact of imbalanced priors on classifier performance.

In addition, different preprocessing techniques, such as Z-normalization and PCA, have been employed to assess their impact on classifier performance.

2 Gaussian Classifiers

We'll start considering Multivariate Gaussian Classifiers (MVG) for those with the following covariance matrices: Full Covariances, Tied Covariances, Diagonal Covariances.

2.1 Objective function

The objective function for the Multivariate Gaussian (MVG) Classifier involves maximizing the log-likelihood of the observed data given the class labels. For each covariance matrix configuration (Full, Tied, Diagonal), the objective function will be adapted accordingly. In detail, all the models assume that data can be described with a Gaussian distribution, given the class:

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

The MVG assumes that each class has its own mean μ_c and covariance matrix Σ_c. Regarding tied models, we assume that the covariance matrices of different classes are the same. The Naive Bayes assumption presumes that all the features are independently distributed and uncorrelated, and this, for the MVG, leads to diagonal covariance matrices.

2.2 Expectations

As we have seen in the scatter plots, the classes do not seem linearly separable, so the Tied Covariance model should not perform well if compared to the other models. The same holds for Naive Bayes assumption, since the number of samples is limited compared to the dimensions. In contrast, our expectations are more optimistic for the MVG Full Covariance model: given its capability to create quadratic decision hyperplanes, this model aligns well with the characteristics of our data. Hence, we expect the MVG Full Covariance model to perform better, considering the non-linear nature of the underlying patterns in our dataset.

2.3 Results

In the following tables are reported the normalized minimum detection costs for the different scenarios.

MVG Classifier			
PCA	minDCF, π = 0.1	minDCF, π = 0.5	minDCF, π = 0.9
No	0,616	0,331	0,111
PCA: 9	0,629	0,330	0,109
PCA: 8	0,613	0,333	0,109
PCA: 7	0,618	0,341	0,115
PCA: 6	0,612	0,336	0,109
PCA: 5	0,652	0,359	0,115

MVG Classifier with tied covariance			
PCA	minDCF, π = 0.1	minDCF, π = 0.5	minDCF, π = 0.9
No	0,706	0,486	0,184
PCA: 9	0,696	0,492	0,180
PCA: 8	0,686	0,485	0,181
PCA: 7	0,700	0,484	0,183
PCA: 6	0,730	0,483	0,181
PCA: 5	0,706	0,490	0,192

Naive Bayes Classifier			
PCA	minDCF, π = 0.1	minDCF, π = 0.5	minDCF, π = 0.9
No	0,806	0,472	0,144
PCA: 9	0,740	0,369	0,113
PCA: 8	0,755	0,360	0,112
PCA: 7	0,742	0,361	0,113
PCA: 6	0,739	0,360	0,116
PCA: 5	0,774	0,371	0,115

Naive Bayes Classifier with tied covariance			
PCA	minDCF, π = 0.1	minDCF, π = 0.5	minDCF, π = 0.9
No	0,790	0,551	0,198
PCA: 9	0,772	0,543	0,202
PCA: 8	0,774	0,544	0,201
PCA: 7	0,769	0,541	0,199
PCA: 6	0,780	0,549	0,201
PCA: 5	0,787	0,530	0,207

The results seem to confirm our initial expectations.

PCA proves highly beneficial, especially for the Naive Gaussian classifier, leading to a significant reduction of the minDCF. However, its advantages for the Naive Tied model are more limited. While PCA offers notable reductions in terms of minDCF, its application to the MVG and Tied classifiers doesn't achieve substantial improvements. Furthermore, we observe a notable decline in performance when reducing dimensions to 5 or fewer, indicating significant information loss.

Results presented above represent minDCF values across various combinations; varying the prior probabilities, we can see that costs decrease by increasing prior probability. We have lower costs when the application has a prior of 90% for the target, since this decreases the number of errors with a high cost (false positive). We expect that this behavior will be the same also for all the other classifiers. Given this and that the fact that the hyperparameters depends on the target application, we will now focus exclusively on comparing the different working points for the most promising model configurations.

3 Logistic regression

3.1 Objective function

We now put our focus on the regularized linear Logistic Regression model, using the following prior weighted formulation of the objective function, and try to optimize the classifier. The implemented objective function is:

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

The regularization coefficient, denoted by λ, is a crucial hyperparameter that needs to be carefully selected to optimize the performance of the classifier. It is incorporated into the regularization term of the model, which aims to encourage simpler solutions and mitigate the risk of overfitting. If λ is too small, the model may produce a solution that effectively separates the classes in the training set, but performs poorly on unseen data, indicating overfitting. Conversely, if λ is too large, the model may fail to adequately separate the classes, leading to suboptimal performance. In the process of selecting the best value for λ, we consider the effective prior of our target application, denoted as π_T. This effective prior considers not only the distribution of classes, but also the costs associated with different types of errors. By incorporating this information, we make a more informed decision about the appropriate value for λ that balances model complexity and generalization performance.

To obtain non-linear separation rules, we can use an expanded space of the features ϕ(x) rather than X in the objective function mentioned above.

$$\phi(x) = \begin{bmatrix} vec(xx^T) \\ x \end{bmatrix}$$

3.2 Expectations

As previously stated, the linear model is expected to perform poorly, indeed we have observed unfavorable outcomes with the linear versions of the Gaussian classifier. On the other hand, the quadratic version appears to be better suited for our purposes.

3.3 Results

3.3.1 Linear logistic regression

In Figure 5 we compare model performance, in terms of minimum DCF, when varying the hyperparameter λ. We also represent how linear logistic regression is affected by pre-processing strategies like Principal Component Analysis and Z-normalization.

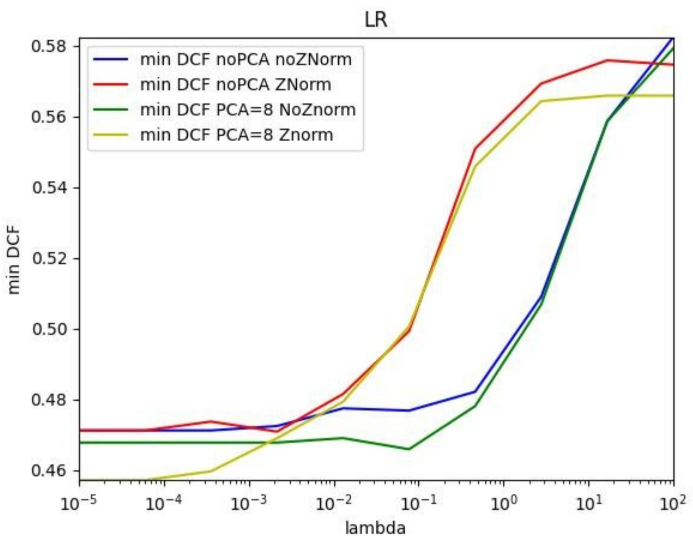
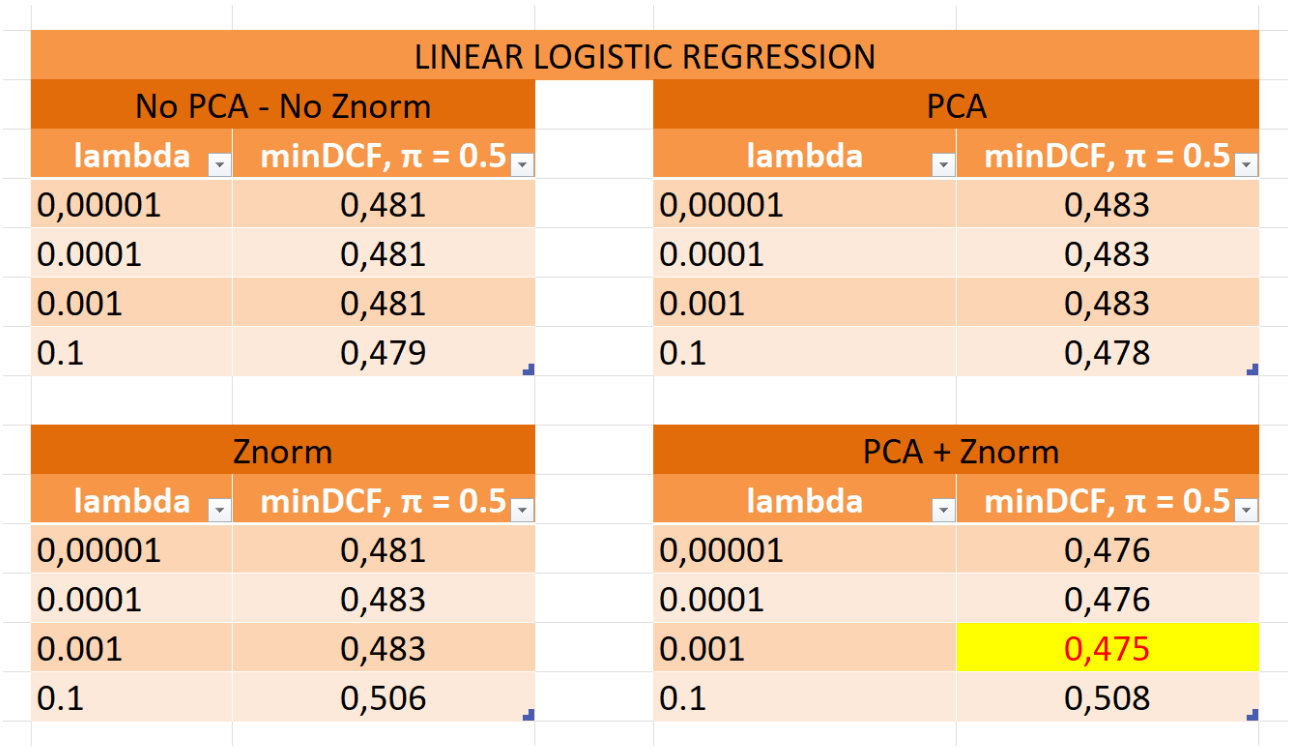


Figure 5: *Costs when varying hyper-parameter λ in linear LR*

In numbers, our results are:



LINEAR LOGISTIC REGRESSION			
No PCA - No Znorm		**PCA**	
lambda	**minDCF, π = 0.5**	**lambda**	**minDCF, π = 0.5**
0,00001	0,481	0,00001	0,483
0.0001	0,481	0.0001	0,483
0.001	0,481	0.001	0,483
0.1	0,479	0.1	0,478
Znorm		**PCA + Znorm**	
lambda	**minDCF, π = 0.5**	**lambda**	**minDCF, π = 0.5**
0,00001	0,481	0,00001	0,476
0.0001	0,483	0.0001	0,476
0.001	0,483	0.001	0,475
0.1	0,506	0.1	0,508

From out experiments we can find out that the combination of both PCA and Z-normalization pre-processing strategies have a considerable impact on performance for small values of λ.

3.3.2 Quadratic logistic regression

We can study how the performance of this category of models is affected by the application of pre-processing strategies and the choice of hyper-parameter λ the same way we did for linear logistic regression.

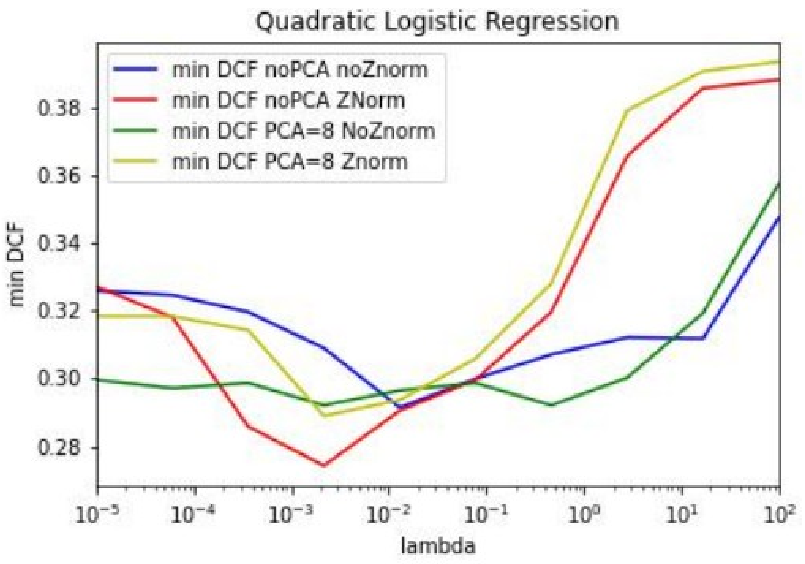


Figure 6: *Costs when varying λ in QLR*

Figure 6 graphically represents the following results:

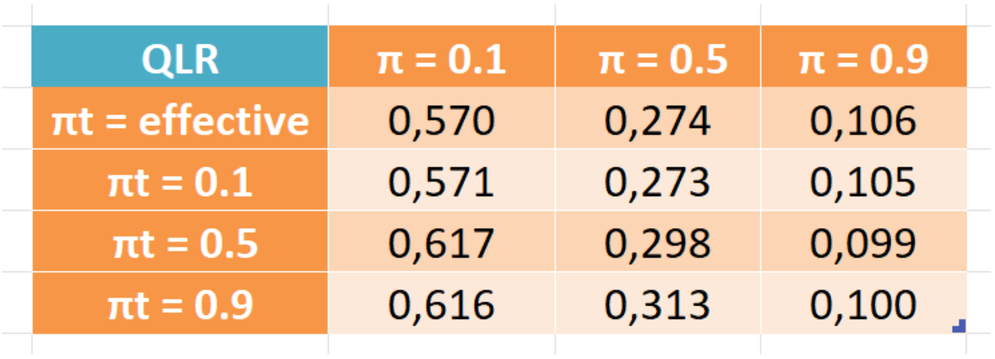
QUADRATIC LOGISTIC REGRESSION (lambda = 0,001)		
No PCA - Znorm		
π	minDCF	errorRate
0,1	0,572	0,061
0,5	0.298	0.049
0,9	0,100	0,079

As expected, quadratic methods perform way better than linear models.

3.4 Conclusions

In conclusion, we analyzed the performance of the Logistic Regression Classifier using two selected models across four different priors and evaluated them over three working points using the best configurations.

LLR	π = 0.1	π = 0.5	π = 0.9
πt = effective	0,737	0,458	0,189
πt = 0.1	0,735	0,458	0,188
πt = 0.5	0,726	0,476	0,190
πt = 0.9	0,728	0,507	0,195



QLR	π = 0.1	π = 0.5	π = 0.9
πt = effective	0,570	0,274	0,106
πt = 0.1	0,571	0,273	0,105
πt = 0.5	0,617	0,298	0,099
πt = 0.9	0,616	0,313	0,100

From graphs and tables of our experiments we can state that the best model we can choose among the studied ones is the Quadratic Logistic Regression with λ=0.001, adopting Z-normalization as pre-processing, operating around working point (π=0.9, Cfn=1, Cfp=10).

4 Support Vector Machines

4.1 Objective function

Support Vector Machines are linear classifiers that look for maximum margin separation hyper-planes. Rather than selecting the hyperplane that maximizes class probabilities like Logistic Regression, this model chooses the one that maximizes the margin, which is the distance of the closest point from the selected hyperplane. There are two different formulations for this model:

The (primal) SVM objective consists in minimizing:

$$J(\boldsymbol{w}, b) = \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n}\max(0, 1 - z_i(\boldsymbol{w}^T\boldsymbol{x}_i + b))$$

The SVM dual solution is the maximizer of:

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

$$\text{s. t. } 0 \leq \boldsymbol{\alpha}_i \leq C, \forall i \in \{1\dots n\}, \quad \sum_{i=1}^{n}\boldsymbol{\alpha}_i z_i = 0$$

With H which is a matrix whose elements are:

$$\boldsymbol{H}_{ij} = z_i z_j \boldsymbol{x}_i^T \boldsymbol{x}_j$$

4.2 Expectations

Considering the distribution of our data and the results obtained using the linear logistic function, we anticipate that the quadratic versions of this model will yield better performance than the linear version of SVMs.

4.3 Results

4.3.1 Linear SVM

Linear Support Vector Machines use the primal formulation of the problem. Figure 7 shows the behavior of the model when varying hyper-parameter *C*, whose choice is crucial to find a tradeoff between large margin and the number of errors in classification. We also consider the impact of the same pre-processing strategies we considered in Logistic Regression, PCA and Z-normalization.

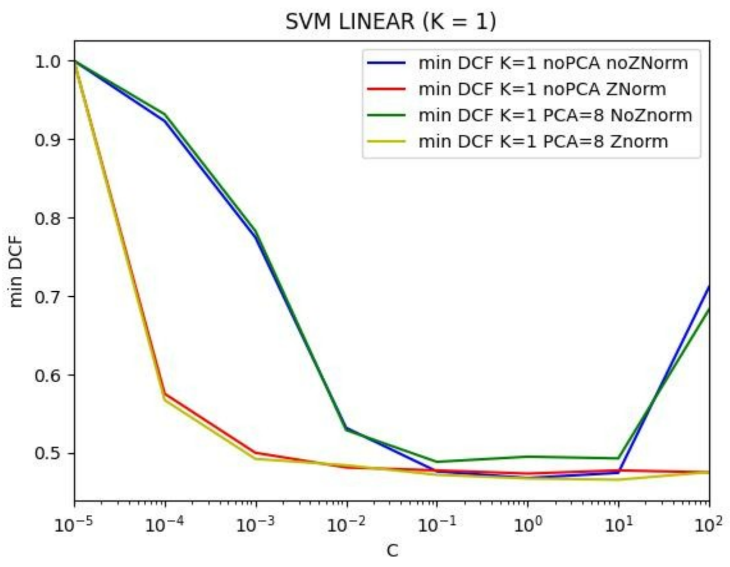
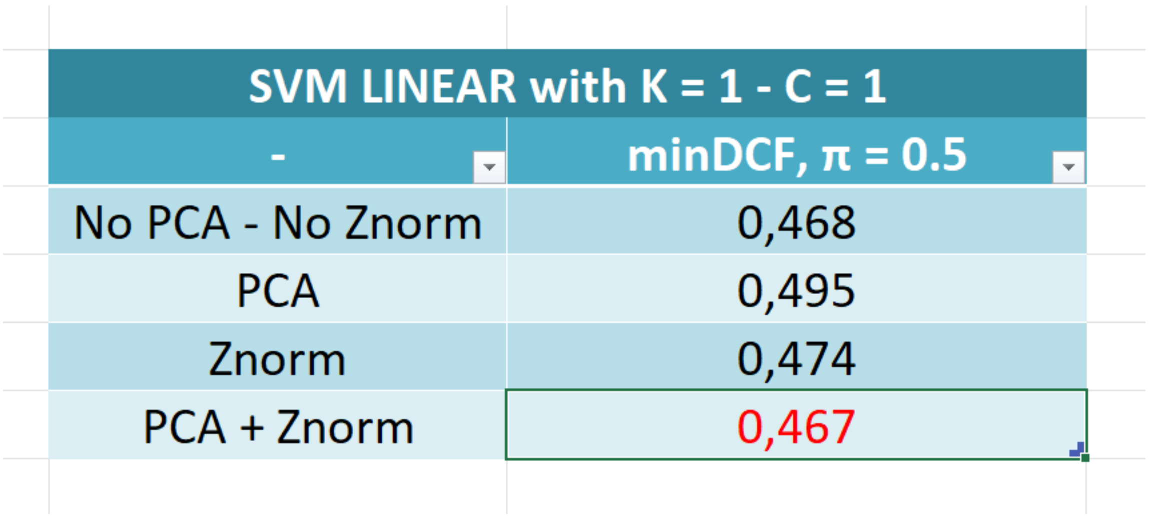


Figure 7: *Costs when varying C in linear SVM*



SVM LINEAR with K = 1 - C = 1	
-	minDCF, π = 0.5
No PCA - No Znorm	0,468
PCA	0,495
Znorm	0,474
PCA + Znorm	0,467

Best results of Figure 7 are summarized in the table above. Again, we see that the combination of both PCA and Z-normalization provide the best performance of the model in terms of costs. Being a linear model, as we could expect, minimum DCF is still to high to consider linear SVM as a valid model for our classifier.

4.3.2 Polynomial SVM

Polynomial SVM is defined as:

$$k(\boldsymbol{x}_1, \boldsymbol{x}_2) = \left(\boldsymbol{x}_1^T \boldsymbol{x}_2 + 1\right)^d$$

We again study the impact of the choice of hyper-parameter C and the impact of PCA and Z-normalization over costs, representing results in graph in Figure 8, differing in the choice of parameter *c*.

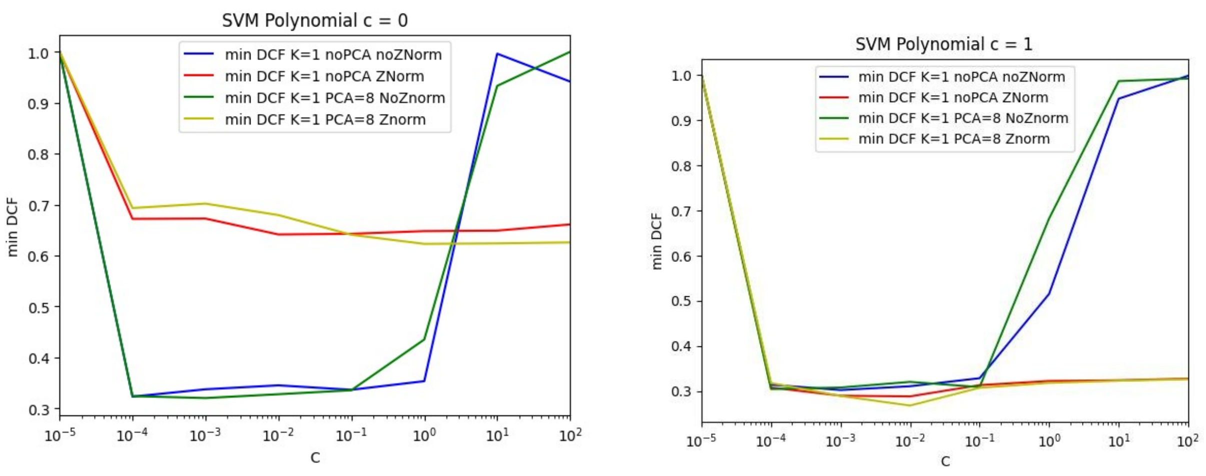
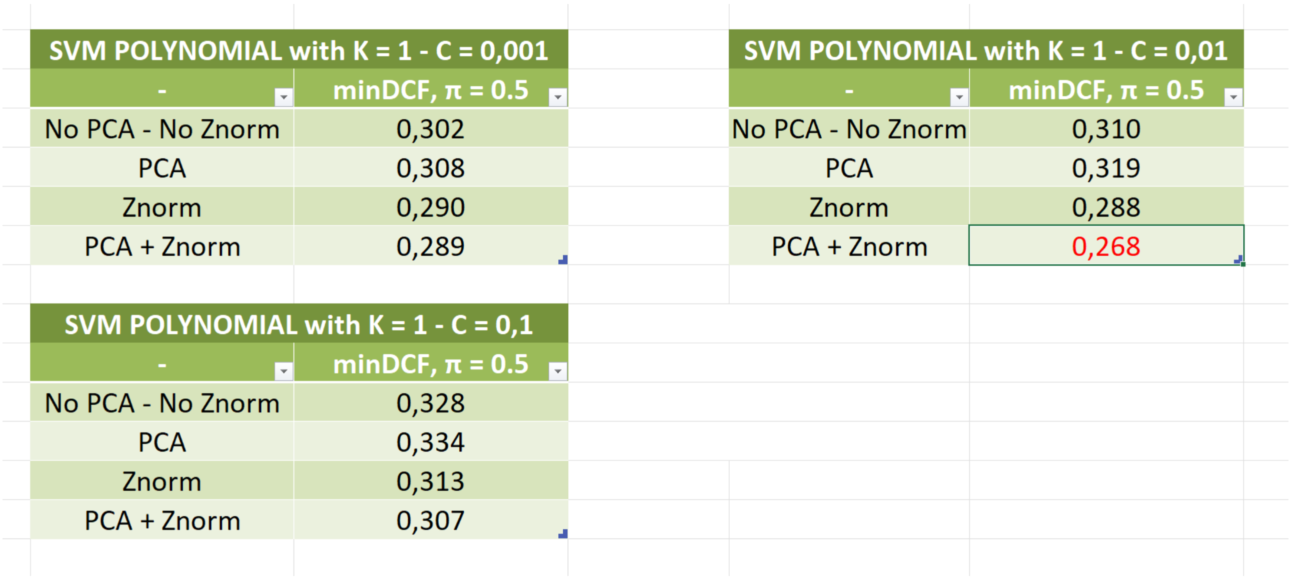


Figure 8: *Costs when varying C in polynomial SVM*

The choice of parameter c=1 is easy to make, while it is worth studying the minDCF across C=10^{-4} and C=10^{-1} in terms of numbers. We have:



SVM POLYNOMIAL with K = 1 - C = 0,001	
-	minDCF, π = 0.5
No PCA - No Znorm	0,302
PCA	0,308
Znorm	0,290
PCA + Znorm	0,289

SVM POLYNOMIAL with K = 1 - C = 0,01	
-	minDCF, π = 0.5
No PCA - No Znorm	0,310
PCA	0,319
Znorm	0,288
PCA + Znorm	0,268

SVM POLYNOMIAL with K = 1 - C = 0,1	
-	minDCF, π = 0.5
No PCA - No Znorm	0,328
PCA	0,334
Znorm	0,313
PCA + Znorm	0,307

As highlighted in tables, the model providing the best performance is the one we get by choosing C=0.01 and applying, as in the other cases, both PCA and Z-normalization.

It is worth noting that cost with Polynomial Kernel SVM is almost half the cost we got while studying Linear SVM.

4.3.3 Radial Basis Function SVM

This category of SVM follows the definition:

$$k(\boldsymbol{x}_1, \boldsymbol{x}_2) = e^{-\gamma \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|^2}$$

In Figure 10 we represent how we studied the behavior of the system when varying parameter γ and hyper-parameter C. Each graph differs in the application of pre-processing strategies.

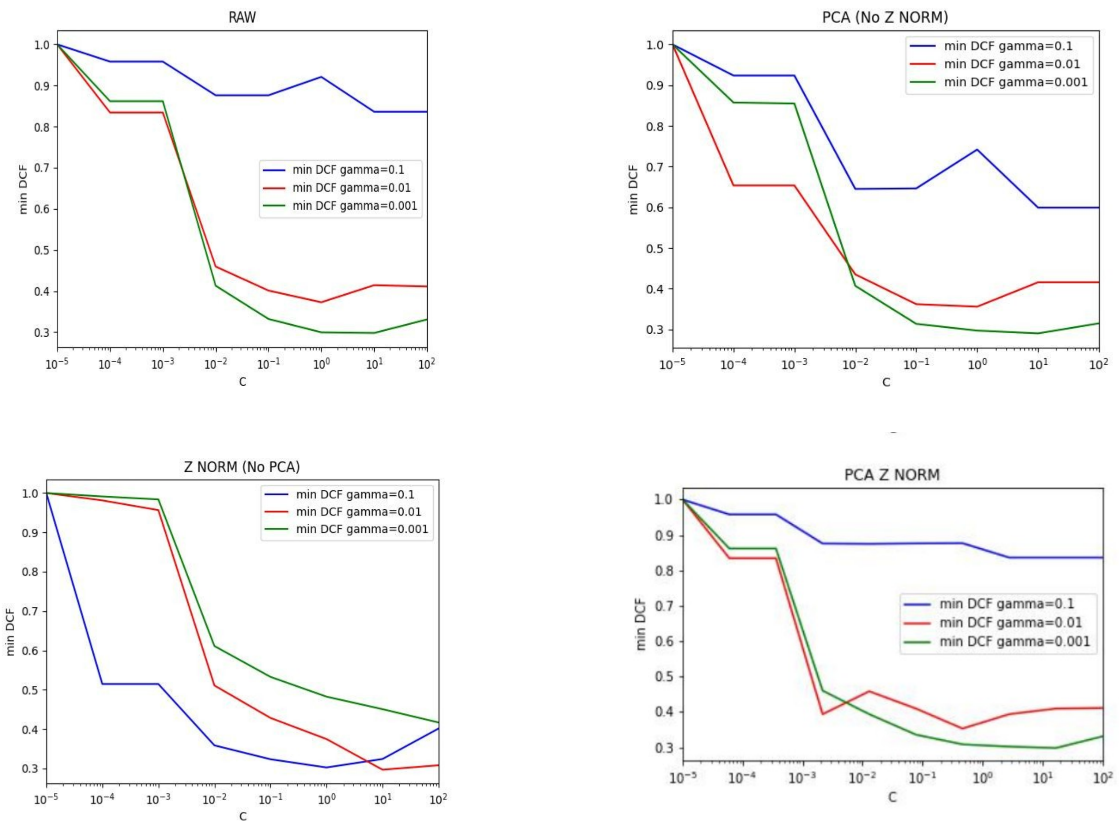
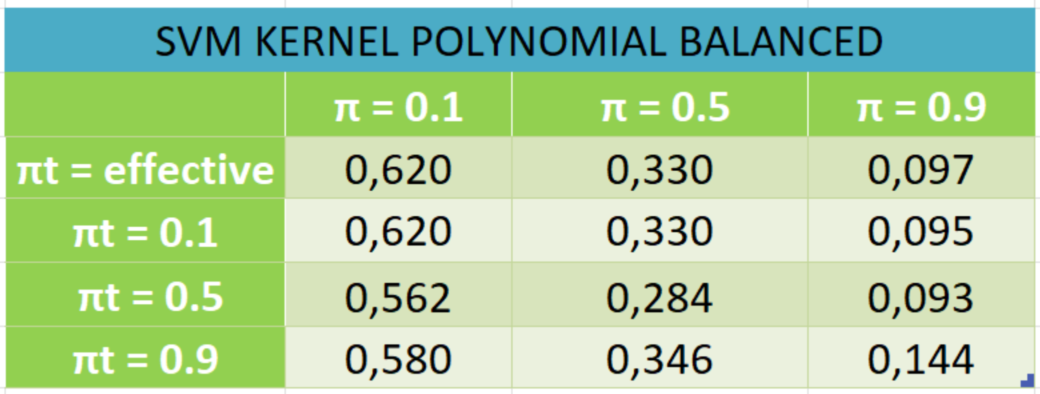


Figure 10: *Costs when varying C and γ in radial basis SVM*

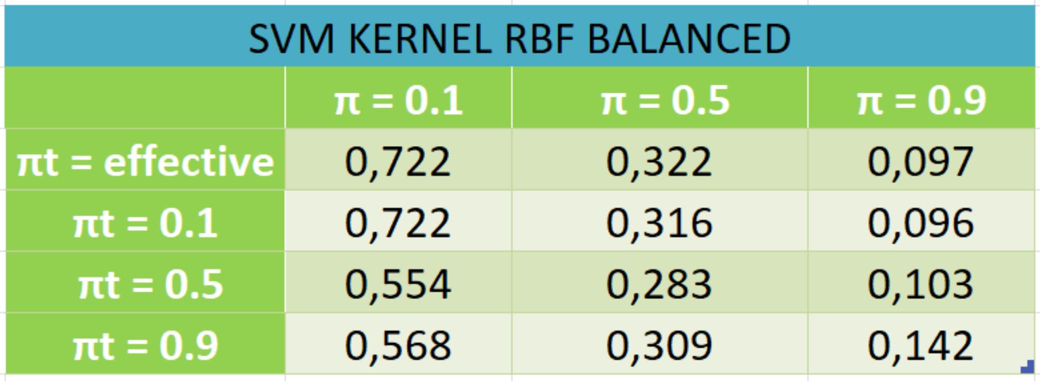
Studying these results, we identify C=10 and γ=10^{-3} as the best parameters, and we choose to not apply Principal Component Analysis and Z-normalization.

4.4 Conclusions

We proceeded with our analysis of the Support Vector Machines model by employing class rebalancing techniques for various priors and evaluating costs across different working points. Specifically, we focused on Kernel models, as the Linear one was found to be unsuitable for our dataset. We maintained the settings previously chosen for both polynomial and radial basis models during this extended analysis.



SVM KERNEL POLYNOMIAL BALANCED			
	π = 0.1	π = 0.5	π = 0.9
πt = effective	0,620	0,330	0,097
πt = 0.1	0,620	0,330	0,095
πt = 0.5	0,562	0,284	0,093
πt = 0.9	0,580	0,346	0,144



SVM KERNEL RBF BALANCED			
	π = 0.1	π = 0.5	π = 0.9
πt = effective	0,722	0,322	0,097
πt = 0.1	0,722	0,316	0,096
πt = 0.5	0,554	0,283	0,103
πt = 0.9	0,568	0,309	0,142

The choice of (π=0.9, Cfn=1, Cfp=10) as working point is confirmed to be the one associated with lower costs, since we reduce the chance of making errors associated with high costs.

Among our experiments, the model that performs better, and that we choose to use in the following steps of our project, is the Polynomial Kernel SVM, to which we apply PCA and Z-normalization as pre-processing strategies.

5 Gaussian Mixture Models

5.1 Objective function

In conclusion, we return to generative approaches and focus on Gaussian Mixture Models (GMMs). These models are usually not used for classification but are used for clustering through techniques such as K-means clustering and for model estimation.

5.2 Expectations

From our dataset analysis and the examination of Gaussian models, we deduce that certain models perform well, particularly GMMs and Diagonal GMMs. GMMs exhibit promising because of their ability to represent different distributions, leading us to anticipate superior outcomes compared to standard Gaussian models. Authentic fingerprint samples appear to align well with Gaussian distributions. Conversely, spoofed fingerprint samples comprise various subclasses. Therefore, we consider models with varying numbers of components for target and non-target classes. For authentic fingerprint samples, a few components (or clusters) may suffice, given their Gaussian-like distribution. In contrast, spoofed fingerprint samples may need a higher number of components, as observed from the scatter plot, which indicates distinct clusters likely corresponding to different spoofing techniques. Thus, optimal results are anticipated with four to eight components for the spoofed class (excluding five, six, or seven, as component numbers typically adhere to powers of two) and one or two components for the authentic class. The two diagonal models are also expected to yield favorable outcomes, as the features exhibit minimal correlation, as evidenced by the heatmaps.

5.3 Results

We assess performance by testing different combinations of the number of components for the two classes, considering both full covariance and diagonal models, with and without covariance tying. It's important to note that for tied covariance models, tying occurs at the class level, meaning different classes have different covariance matrices. The number of components of each Gaussian Mixture Model (GMM) serves as the hyperparameter to tune. Initially, our focus is on raw features.

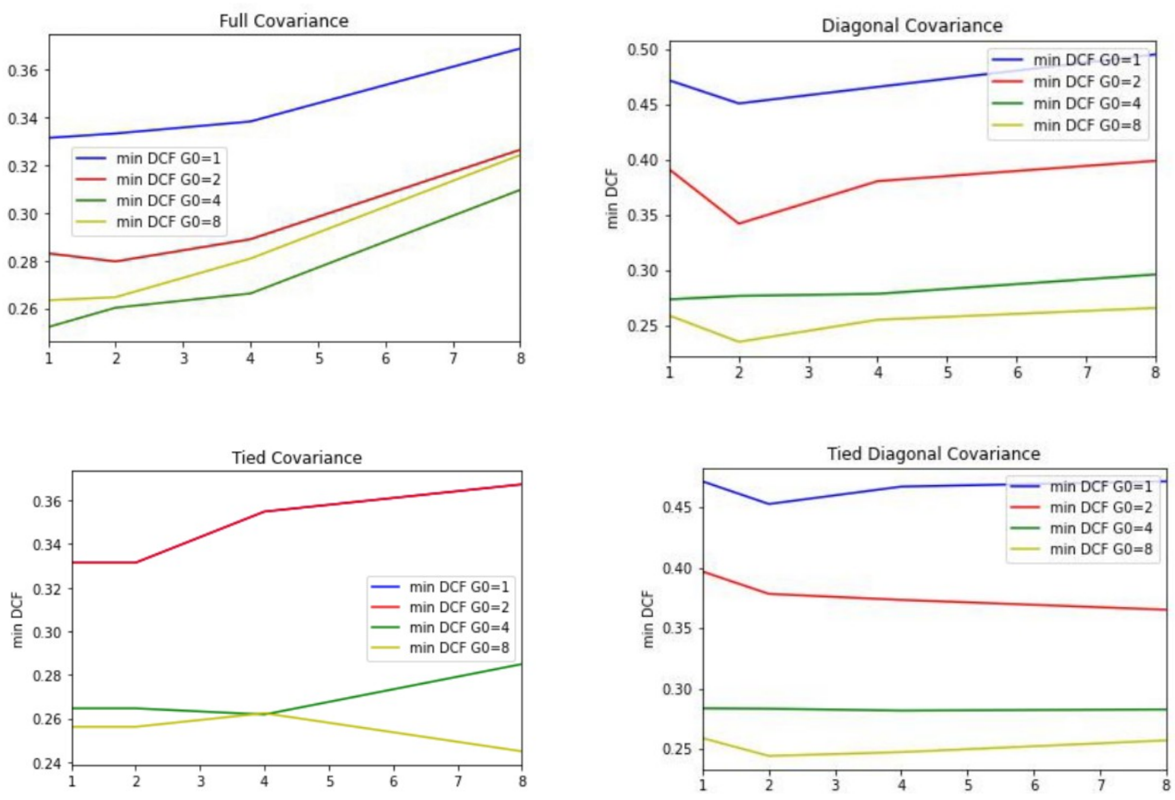
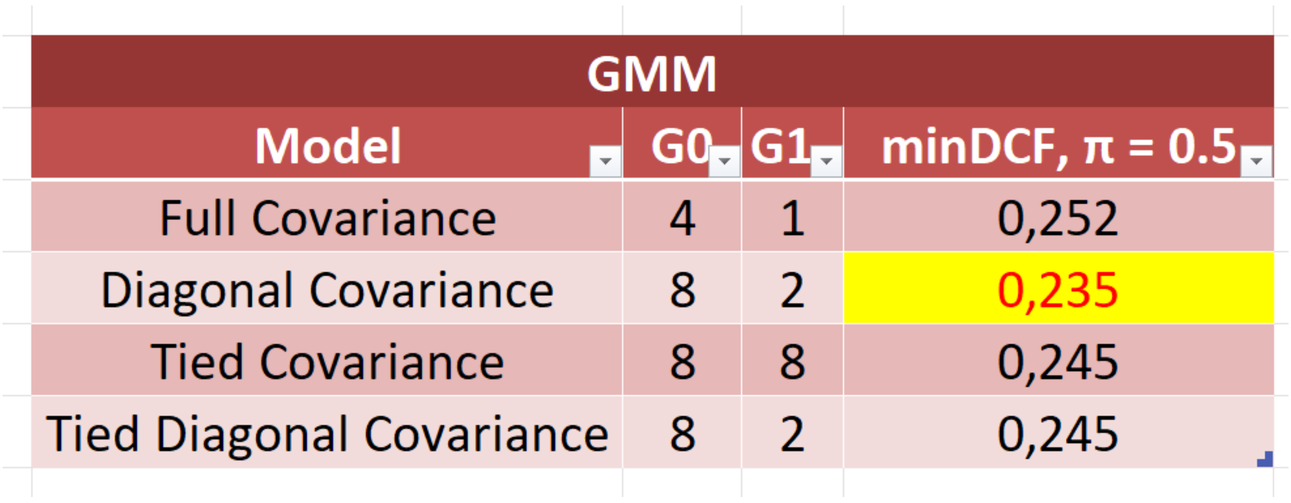


Figure 11: *Performance of GMM models*



GMM			
Model	G0	G1	minDCF, π = 0.5
Full Covariance	4	1	0,252
Diagonal Covariance	8	2	0,235
Tied Covariance	8	8	0,245
Tied Diagonal Covariance	8	2	0,245

From both graphs in Figure 11 and the table above we can see that the best model with raw feature is the Diagonal with G0=8 and G1=2.

We also tried to apply pre-processing strategies, again combining Principal Component Analysis and Z-normalization, but none of them performed better than models without pre-processing.

It turned out that the best GMM model is the diagonal one trained on raw features with 8 components for class 0 and 2 components for class 1. This configuration aligns with our expectations for both target and non-target classes. The diagonal model performs well, likely because our features exhibit low correlation. As we increase the number of considered Gaussians, the diagonal model becomes more effective. Given the data's distribution across many subclasses, particularly for the non-target class, the assumption of feature independence within each distribution becomes more accurate when using a diagonal GMM. This model also resulted in the lowest DCF we have encountered up to this point.

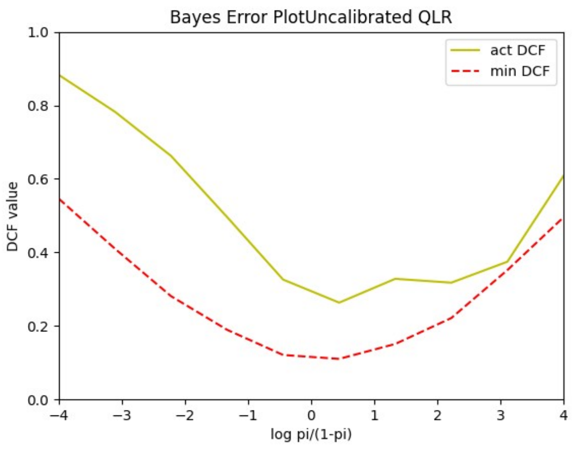
6 Calibration

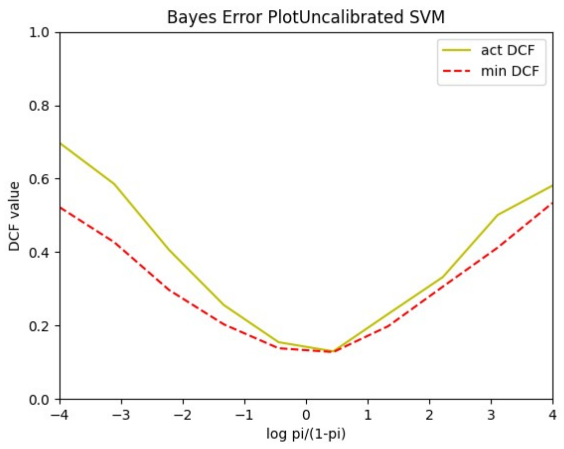
Up to this point, we have employed the minimum DCF as the primary metric for selecting the best model candidates. This metric reflects the minimal cost incurred if optimal decisions were made based on the known optimal threshold. Now, we shift our focus to the actual DCF metric, which accounts for the cost obtained using the theoretical threshold calculated with the effective prior $\tilde{\pi}$.

The disparity between the minimum and actual DCF signifies the loss due to miscalibrated scores. In our scenario, we anticipate that the SVM scores are likely to be substantially uncalibrated, given that they are non-probabilistic scores. To mitigate this miscalibration, we utilize the uncalibrated scores as inputs for prior-weighted logistic regression. Specifically, we subtract the optimal threshold from the scores. The formulation to derive the new scores is as follows:

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

In the following, we are going to adopt the three best models among the ones we studied above: Quadratic Logistic Regression (QLR), Polynomial Kernel SVM (SVM), and Diagonal Gaussian Mixture Model (GMM). In Figure 12 we plotted actualDCF and minDCF using Bayes Error Plot:





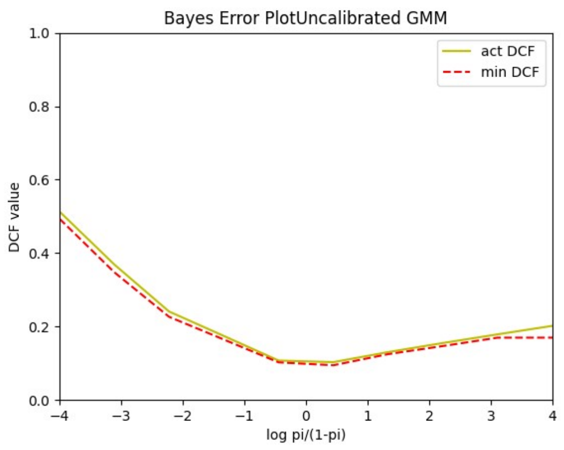
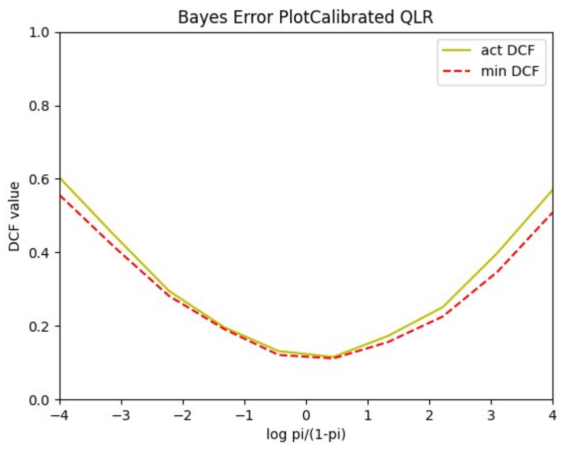
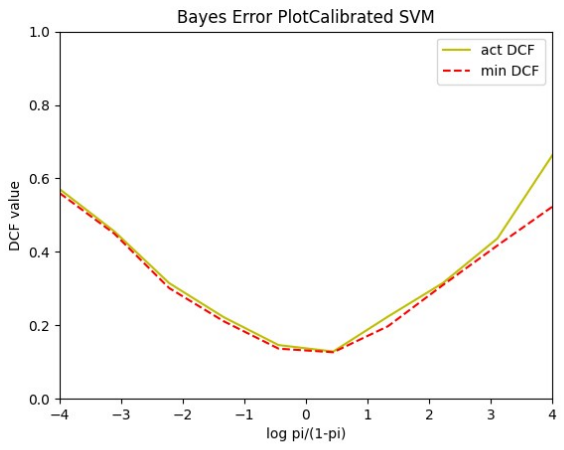


Figure 12: *Uncalibrated scores*

As expected, the Support Vector Machine performs the worst in terms of actual DCF,while the quadratic logistic regression would likely benefit from calibration. Finally, the scores of the Gaussian Mixture Model seem to be well calibrated. Fgiure 13 contains Bayes error plots to show the performance of models after calibration.





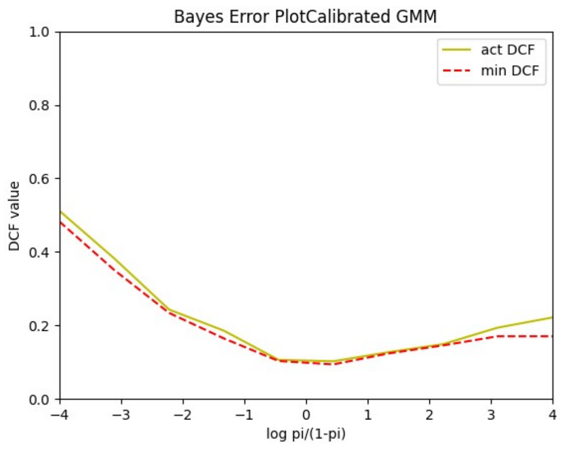
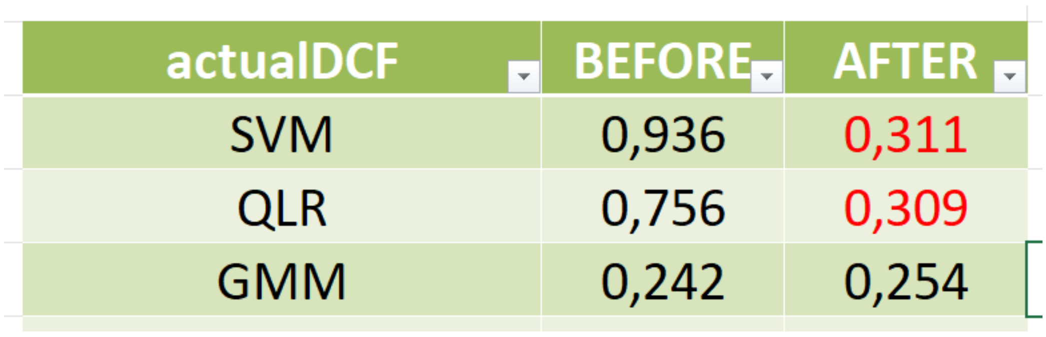


Figure 13: *Calibrated scores*

In a numerical point of view, the table below shows values of actual DCF before and after calibration for the three models we selected. We observed significant improvements for the Support Vector Machine and Quadratic Logistic Regression models, as expected. However, there was a slight decrease in performance for the Gaussian Mixture Model, although it is not considered significant.

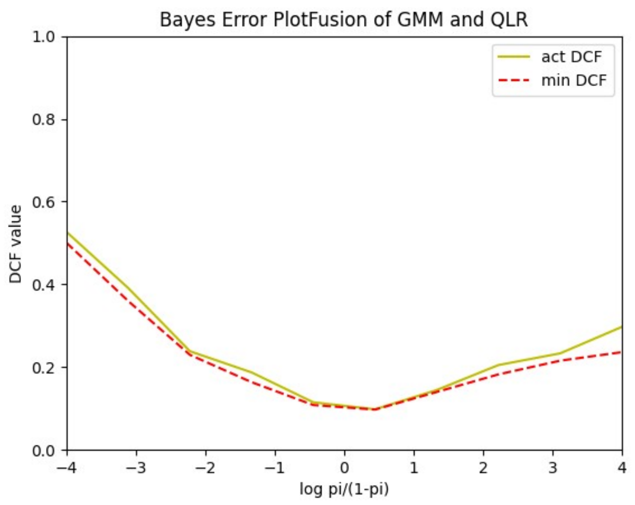


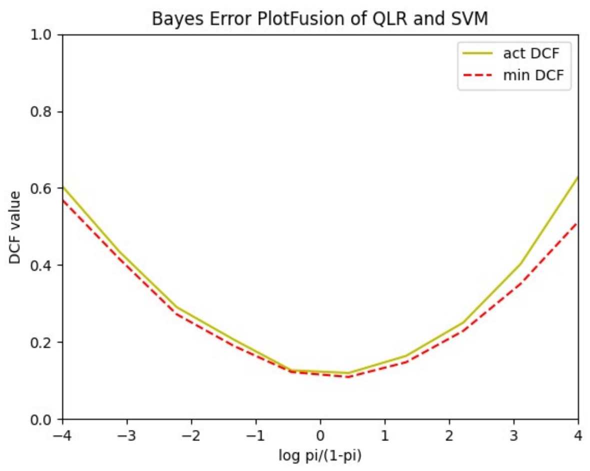
actualDCF	BEFORE	AFTER
SVM	0,936	0,311
QLR	0,756	0,309
GMM	0,242	0,254

7 Fusion

We now combine the outputs of the models we selected, trying to improve the overall performance, putting together benefits and strengths of two models and reducing individual limitations.

In practical terms, we realized fusion stacking the calibrated scores of two classifiers and apply calibration. Figure 14 shows performance for each combination of two models.





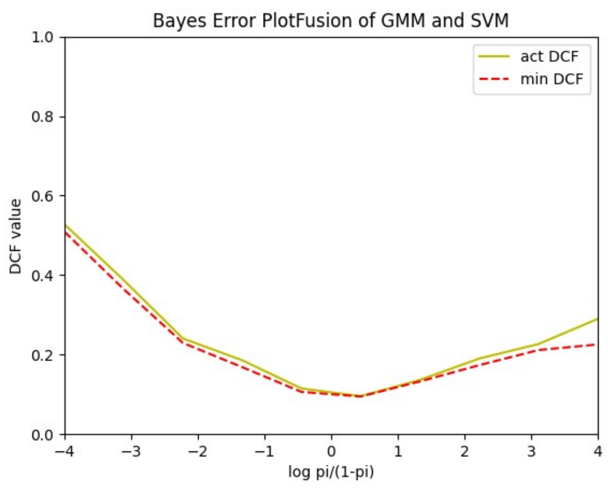
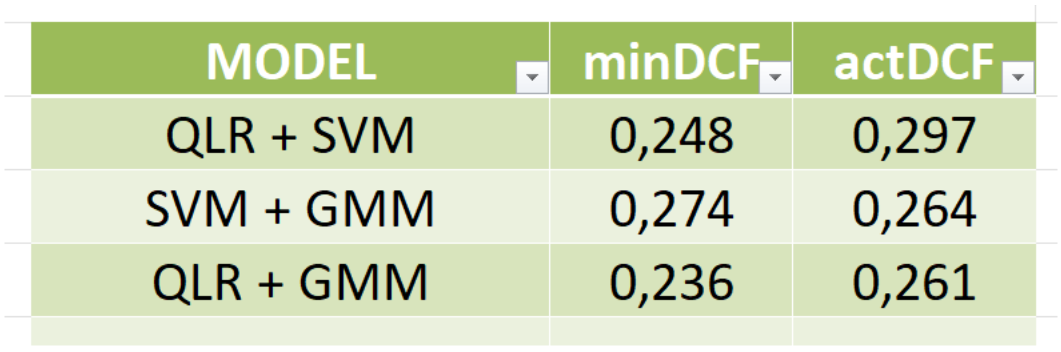


Figure 14: *Fused scores*

In terms of numbers, the table below shows values of actual DCF and minimum DCF for the new three models. The fused model perform well, particularly QLR + GMM and SVM + GMM.



MODEL	minDCF	actDCF
QLR + SVM	0,248	0,297
SVM + GMM	0,274	0,264
QLR + GMM	0,236	0,261

8 Evaluation

In this phase we use the test dataset to measure the performance of our model after all choices we have made so far.

Coherently to what we did for calibration and fusion activities, we consider only the top three models we studied during training step: Quadratic Logistic Regression (QLR), Polynomial Kernel SVM (SVM), and Diagonal Gaussian Mixture Models (GMM). We also include in evaluation the models we built during fusion (QLR + GMM and SVM + GMM).

8.1 Score Calibration

We start applying score calibration to models, particularly only to QLR and SVM, given that GMM is already well calibrated.

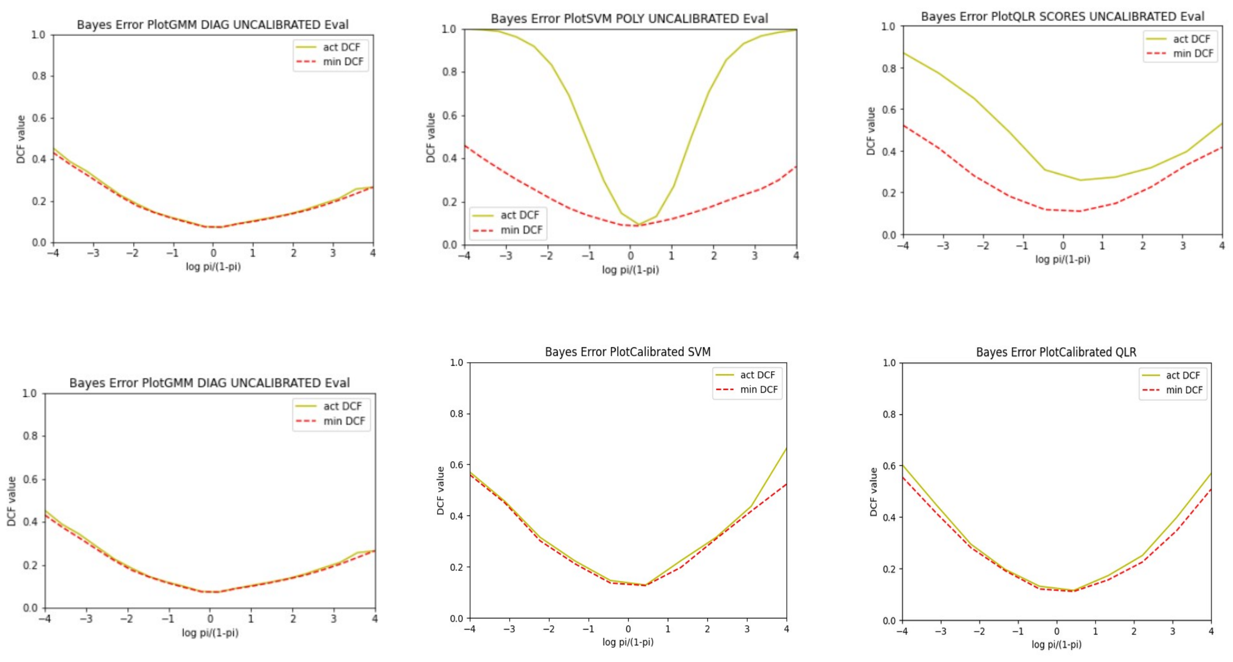
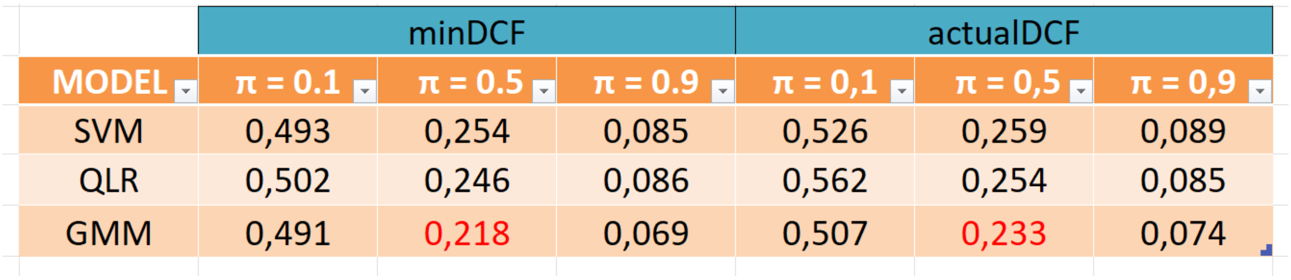


Figure 15: *Uncalibrated and calibrated evaluation scores*

In Figure 15 we show that the performance of the classifiers aligns with what was observed on the training set. Next table summarizes the both the minimum and actual DCF for all the working points we have considered:

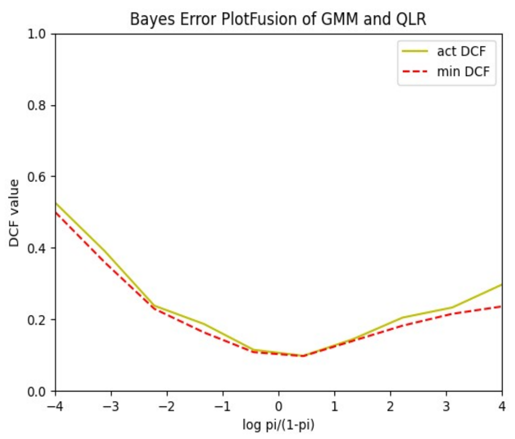


	minDCF			actualDCF		
MODEL	π = 0.1	π = 0.5	π = 0.9	π = 0,1	π = 0,5	π = 0,9
SVM	0,493	0,254	0,085	0,526	0,259	0,089
QLR	0,502	0,246	0,086	0,562	0,254	0,085
GMM	0,491	0,218	0,069	0,507	0,233	0,074

The cost values are consistent with those obtained previously, sometimes even showing improvement, the GMM model continues to maintain its leading position.

8.2 Fused Models

In Figure 16 we analyze the performance of the two fused model: Quadratic Logistic Regression and the Polynomial Kernel SVM both of them fused with the Gaussian Mixture Model.



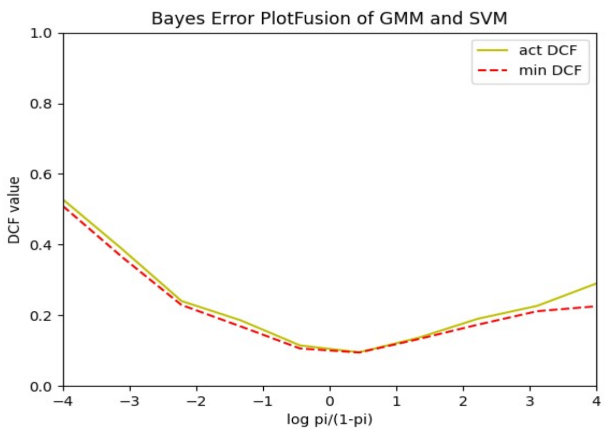


Figure 16: *Fused scores*

In the following table we report the values of both the minimum and actual DCF for all the working points we have considered.

	minDCF			actualDCF		
MODEL	π = 0.1	π = 0.5	π = 0.9	π = 0,1	π = 0,5	π = 0,9
GMM + QLR	0,487	0,214	0,071	0,491	0,237	0,074
GMM + SVM	0,466	0,221	0,070	0,499	0,231	0,074

It seems that the QLR + GMM perform very well even on the evaluation set.

8.3 SVM Polynomial Kernel Evaluation

Since the SVM Polynomial Kernel Evaluation model keeps performing well on the evaluation set, we now revisit the previously choice of the value of the hyperparameter C for the chosen configuration of the Polynomial Kernel SVM.

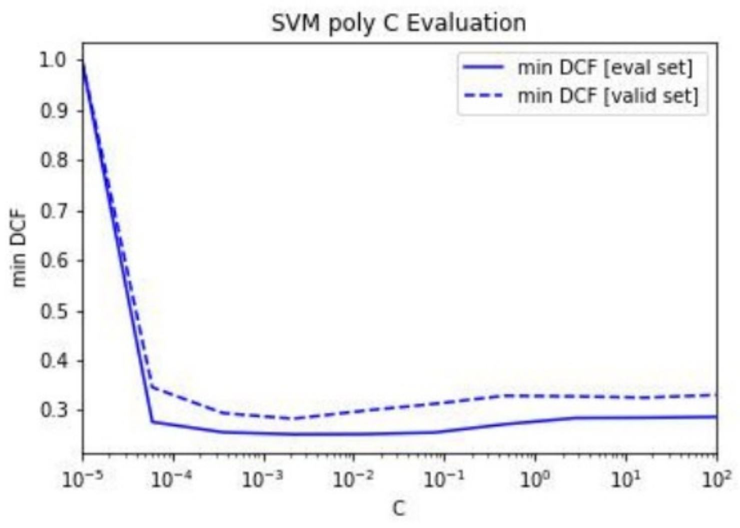
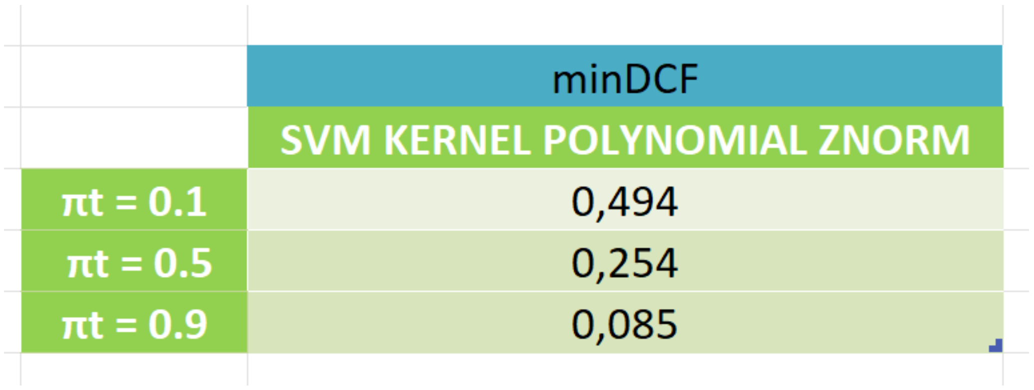


Figure 17: *Costs of SVM when varying C*

The plot in Figure 17 confirms that our choice for the training set remains optimal for the evaluation set as well. The table below shows the DCF values recalculated across the three working points.



	minDCF
	SVM KERNEL POLYNOMIAL ZNORM
πt = 0.1	0,494
πt = 0.5	0,254
πt = 0.9	0,085

Given the results, we confirm our previous choice of the hyperparameter C equal to 0.001. We observe an improvement in cost compared to the one found during the training phase.

8.4 Quadratic Logistic Regression Evaluation

Since the QLR model keeps performing well on the evaluation set, we now revisit whether the previously chosen value of the hyperparameter λ (0.001) remains optimal. However, focusing only on the model with Z Normalization.

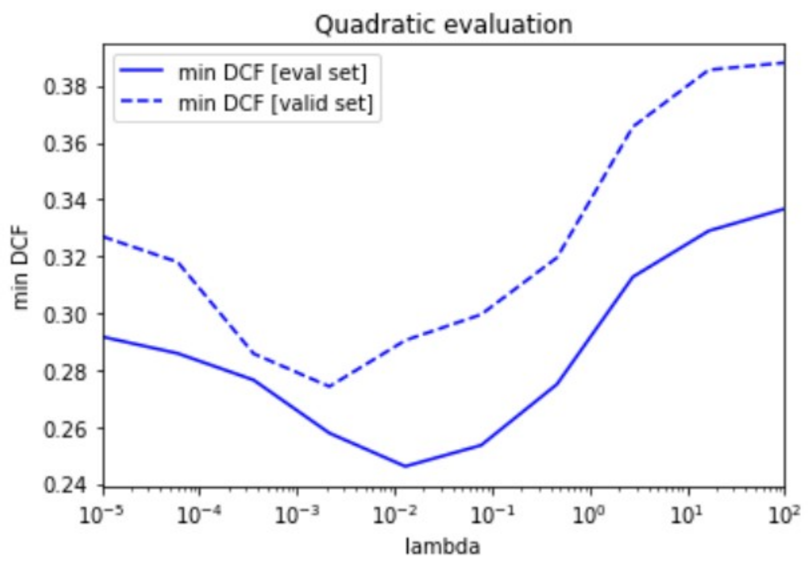


Figure 18: *Costs of QLR when varying λ*

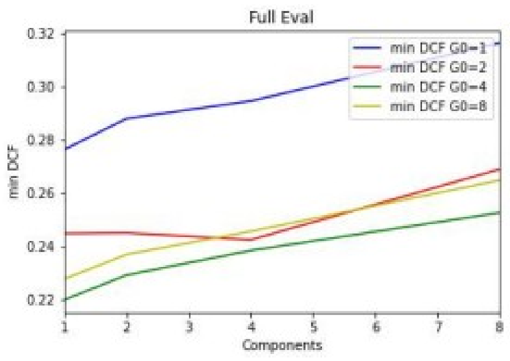
As depicted in the plot in Figure 18, although the costs are relatively low, the selection of λ appears to be suboptimal across the evaluation set. Therefore, we will compare the outcomes between λ set to 0.001 and 0.01. Below, we present the varying results for the three working points: 0.5, 0.1, and 0.9, respectively.

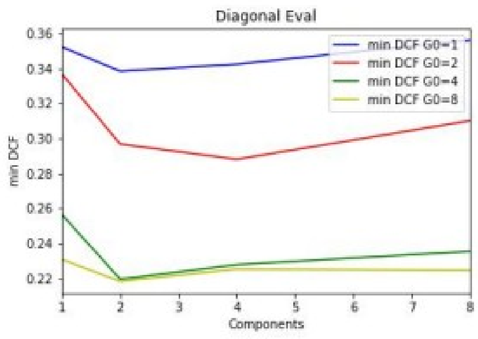
	minDCF	
	QLR (ZNORM) lambda = 0,001	QLR (ZNORM) lambda = 0,01
πt = 0.1	0,501	0,502
πt = 0.5	0,265	0,245
πt = 0.9	0,089	0,084

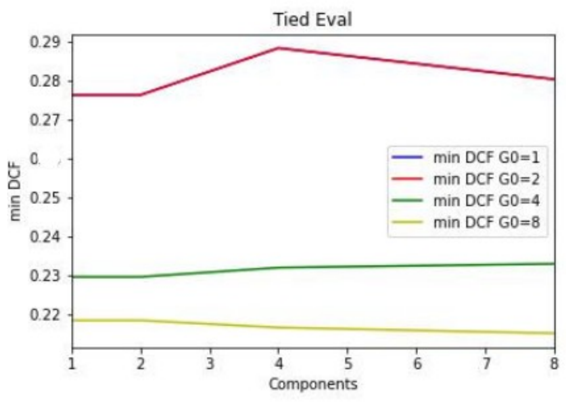
We can confirm our previous expectation, as the value of 0.01 appears to be more suitable for the evaluation, even though the difference is not substantial.

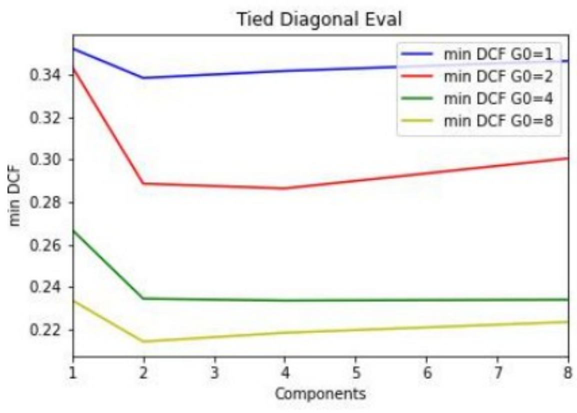
8.5 Gaussian Mixture Model Evaluation

To assess the effectiveness of the Gaussian Mixture Model (GMM) classifier, we revisited alternative variants that were initially excluded during the training phase. This reconsideration was prompted by the observation that most GMM models performed well on our dataset during training. Additionally, the scores demonstrated excellent calibration, and when fused, they improve the performance of other classifiers.









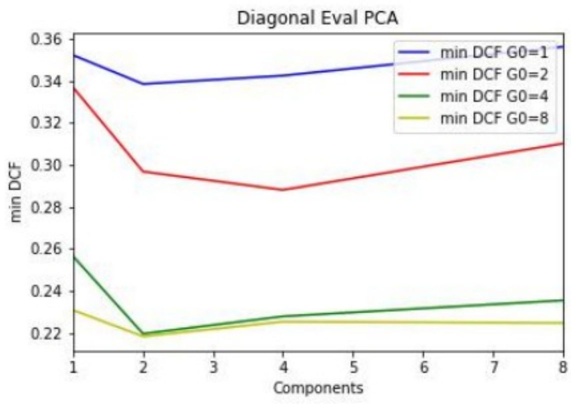
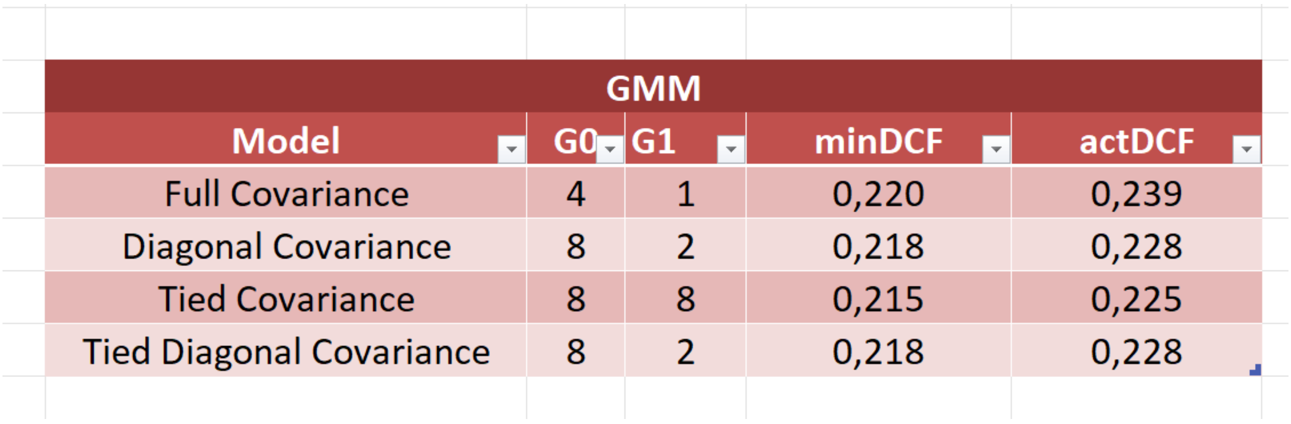


Figure 19: *Performance of studied GMMs*



GMM				
Model	G0	G1	minDCF	actDCF
Full Covariance	4	1	0,220	0,239
Diagonal Covariance	8	2	0,218	0,228
Tied Covariance	8	8	0,215	0,225
Tied Diagonal Covariance	8	2	0,218	0,228

As we expected, all GMM models exhibit excellent performance on the evaluation set. The Tied Covariance model with 8 components for each class appears to slightly outperform the Diagonal model, although the difference is minimal. While each model produces low costs and well-calibrated scores, we find that the assumptions of the diagonal model, along with its chosen number of components, are better aligned with our specific case study.

9 Conclusions

All decisions we took during training and validation phases resulted to be effective during the evaluation phase. The models providing the best results are Diagonal Gaussian Mixture Model and its fusion with Quadratic Logistic Regression. Performances of these two models are similar, therefore we might want to choose GMM to reduce overall complexity. This selection is also justified in terms of results consistency: we have good results for the target application, acceptable results in correspondence of the worst working point, and optimal results in correspondence of the best working point for our application.