Fingerprint spoofing detection

MLPR Project (01URTOV) - September 22, 2023

Freni Davide Giovanni (s305571),
Gagliardo Domenico (s310454)

Politecnico
di Torino

Contents

1 Introduction

The main purpose of the project is the development of a classifier that is able to recognize whether a fingerprint belongs to its real owner or has been spoofed by a malicious user to impersonate another individual.
The dataset employed contains fingerprint images, which are represented by means of embeddings (i.e. low-dimensional representations of images obtained by translating a high-dimensional vector into a low-dimensional one), in order to keep the model tractable. Each embedding is described by 10 continuous variables (with no physical interpretation) and belong to either the authentic fingerprint (label 1) or the spoofed fingerprint (label 0) class. Moreover, the spoofed fingerprint samples belong to one of 6 possible sub-classes, corresponding to different spoofing approaches, but anyway the actual spoofing method (sub-class label) is not available.
The original dataset is composed by two files, named Train.txt and Test.txt, which represent respectively the training set and the test (evaluation) set. The former contains 800 authentic fingerprints and 1525 spoofed fingerprints, for a total of 2325 samples, whereas the latter contains 2400 authentic fingerprints and 5304 spoofed fingerprints, which sum up to 7704 samples. Therefore, we already understand that classes are slightly unbalanced, since spoofed fingerprints represent 66% of the training data and 69% of the evaluation data.
Samples are stored in separate rows in both training and evaluation files, and each value is comma-separated. For both files, each row is made up of ten features first and of one class label in the end (last entry of the row). The class labels used are 0 (spoofed fingerprint) and 1 (authentic fingerprint).
Our task will be to perform classification by employing different techniques, to analyze results and to motivate our choices.
One more thing to point out is that the datasets are imbalanced, with the spoofed fingerprint class having significantly more samples.
Furthermore, the target application considers an application where prior probabilities of authentic and spoofed classes are the same. However, labeling a spoofed fingerprint as authentic has a larger cost due to security issues that the errors could create. The target application working point is therefore defined by the triplet ($\pi_{\mathrm{T}} = 0.5, C_{fn} = 1, C_{fp} = 10$), where π_{T} is the application prior, C_{fn} is the cost of false negative errors and C_{fp} instead is the cost of false positive errors. However, we will also analyze how the model behaves for different working points.

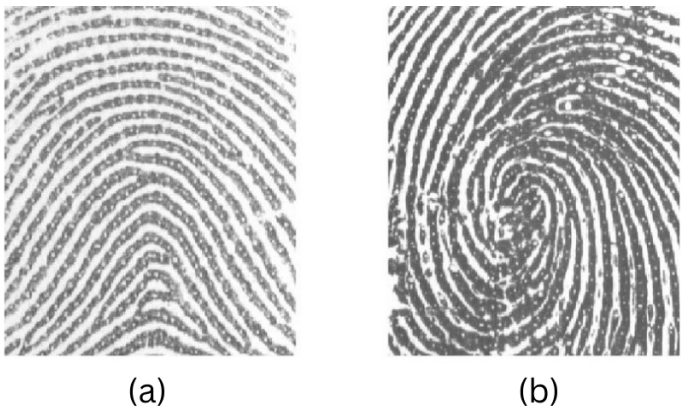


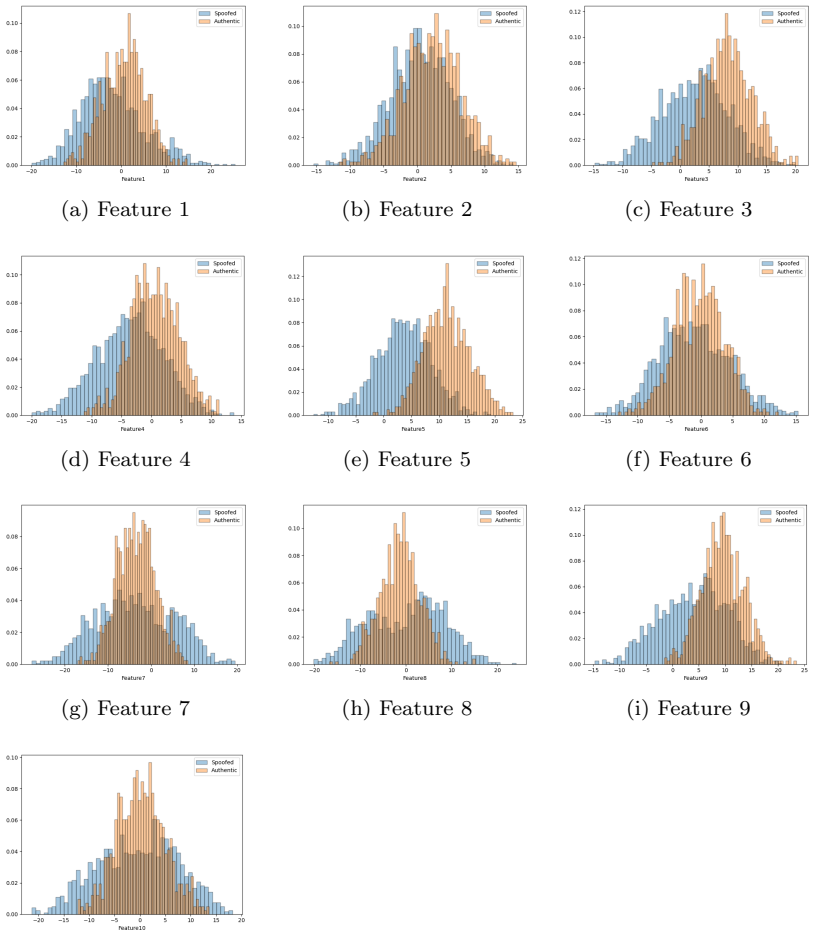
Figure 1: (a) Image of an authentic fingerprint. (b) Image of a spoofed fingeprint.

2 Dataset features analysis

Now, let's move towards dataset features analysis, reminding that each sample is described by 10 continuous variables and by a single class variable.

2.1 Histogram Plots

Therefore, this analysis allows us to understand how the features are distributed and what features are useful for classification. For instance, the histogram plot of features 5 shows that histograms do not overlap that much. So, this features could be useful for classification purposes. In the same way, features 3, 4 and 9 behave pretty well too. Based on these features, we can draw a conclusion about the class to which the current fingerprint belongs. For what concerns all the other features, we can see that the histograms overlap a lot (the classes are overlapping a lot), so these features will not discriminate well our classes.



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

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

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

(j) Feature 10

Figure 2: Histograms of the given dataset features (training set). Orange histograms refer to authentic fingerprints, Blue histograms to spoofed fingerprints.

From the different plots we can see that the histograms representing the features have gaussian-like shapes. So, gaussian models may behave well for our purpose.

2.2 Scatter Plots

It is also possible to reason on scatter plots, which visualize 2 different features on the abscissa and on the ordinate. Since all these scatter plots basically behave in the same way, we show up only the most representative ones where we can see respectively an after all good class separation, a mediocre class separation and a bad class separation.
Focusing on the first scatter plot, we can observe that on the top-right part almost all the fingerprint belong to authentic class. Thus, we understand that if we have a fingerprint with feature 5 in this specific range and feature 9 in this specific range, probably it will belong to the authentic class.

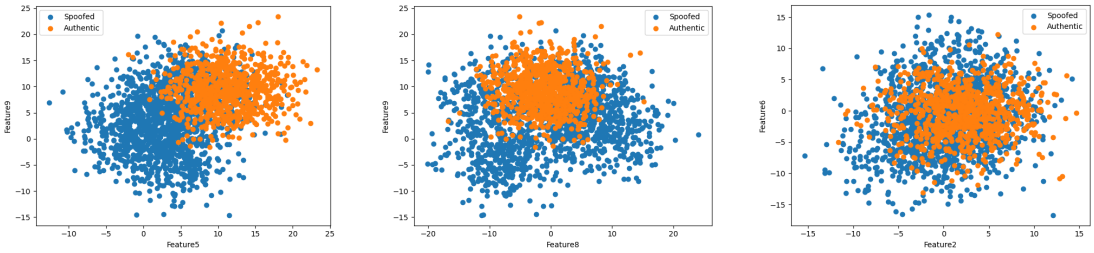


Figure 3: Scatter plots of the most representative pairs of features of the given dataset (training set) for each class. Orange points refer to authentic fingerprints, blue points to spoofed fingerprints.

By applying PCA and retaining only the 2 most relevant dimensions, we can see that probably classes are not linearly separable and quadratic surfaces may be more suited for the classification. In addition, we can see that the spoofed class shows some subsets that probably correspond to clusters representing the different spoofing approaches (the different subclasses). Therefore, authentic fingerprint class might be modeled well by a simple Gaussian distribution, but Gaussian densities may not be sufficient for spoofed fingerprint class.

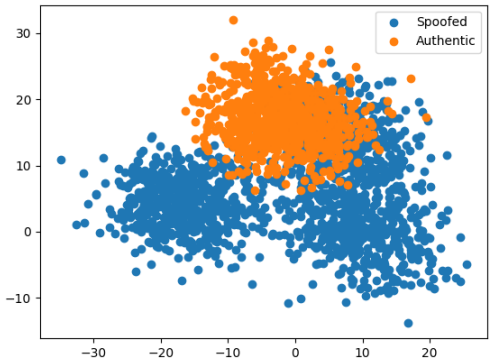


Figure 4: Scatter Plot of the whole Dataset after applying PCA by retaining 2 dimensions.

2.3 Linear Discriminant Analysis

We can also focus on employing LDA (Linear Discriminant Analysis) to better separate the classes, in order to understand if our classes are linearly separable or not. LDA allows estimating at most C-1 directions, where C is the number of classes (in our case we can have only 1 direction since we have a binary classification problem). By observing the plot we can say that a linear classifier (such as Tied classifier) could perform well, given that the classes are more or less well-separated (even if there is a small overlapping area). So, with LDA we understand that a linear classifier may discriminate the classes pretty well, but observing the distribution of the features that we had in the previous scatter plot (Figure 4), probably non-Gaussian and/or non-linear models (such as MVG) will behave better for the task.

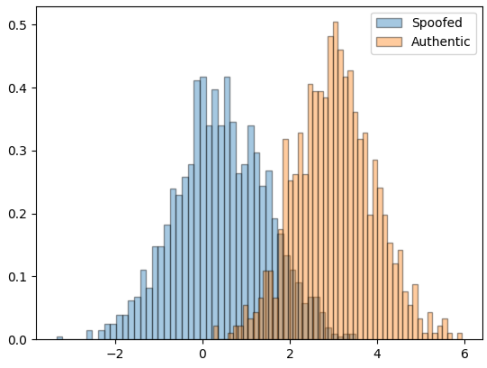


Figure 5: LDA transformation technique applied on the Dataset.

2.4 Pearson Correlation

We can also consider correlation analysis for our dataset by representing Pearson correlation for our features as a heatmap, where black heatmap represents the whole dataset, blue heatmap represents the authentic class and red heatmap represents spoofed class.
The absolute value of the Pearson correlation coefficient is:

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

The whole dataset heatmap plot does not show a significative correlation of the dataset features among them, but focusing on the spoofed fingerprint class we can see that there is a slightly higher correlation. Furthermore, we can see that mapping 10 dimensions to 8 dimensions allows us to retain most of the variance of the data (92%). For this reason it may be worth trying to map data from 10 dimensions to 8 dimensions and see if it could be beneficial, since by removing only two dimensions we will maintain 92% of the variance of the data and still preserving most of the "useful" information. In addition, applying PCA reduces also the number of parameters that we need to estimate for classification.

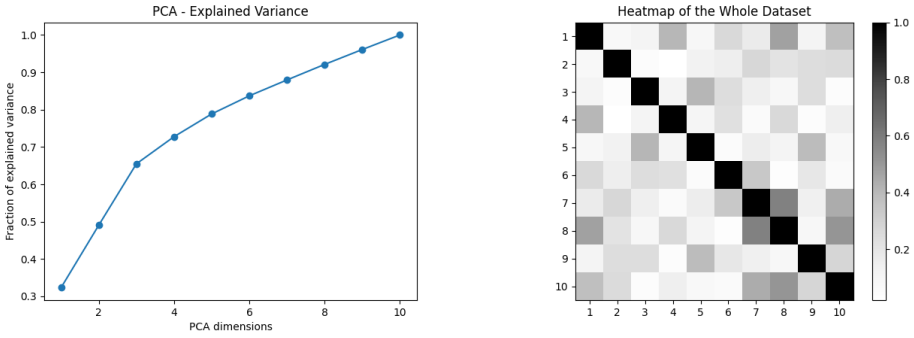


Figure 6: Explained variance and heatmaps of the whole dataset.

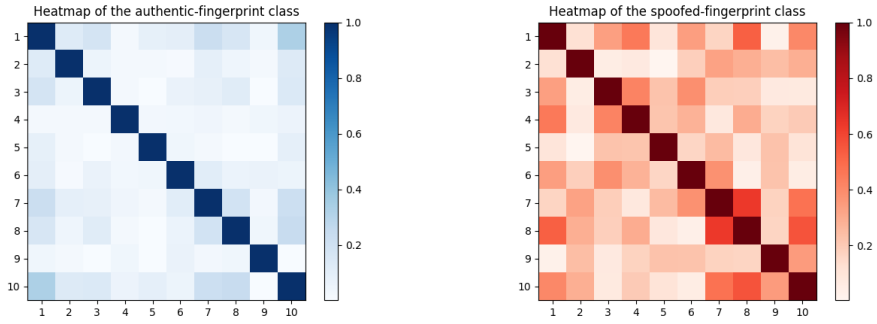


Figure 7: Heatmaps of the authentic class and of the spoofed Class.

3 Model training and model selection

3.1 Approaches

To do our analysis, we can adopt two methodologies:

1. Split the training data in two parts, where the first one is used as training data whereas the second one as validation set (appropriate for large datasets).

2. Use the K-Fold cross validation method (typically done when the data is scarce).

We followed the second way because we do not have so many data at our disposal and K-Fold cross validation allows us to have more data available for both training and validation. In particular, we implemented a K-Fold approach with K=5 (note that the data have been shuffled before splitting, in order to obtain homogeneous folds and minimize model error), which is a good trade-off. We will consider three different applications.
These are a uniform prior application ($\pi_{\mathrm{T}} = 0.5, C_{fn} = 1, C_{fp} = 10$) and two unbalanced applications where the prior is biased towards one of the two classes:

- ($\pi_{\mathrm{T}} = 0.1, C_{fn} = 1, C_{fp} = 10$)
- ($\pi_{\mathrm{T}} = 0.9, C_{fn} = 1, C_{fp} = 10$)

To measure the performance of our models, we will use as metric the normalized minimum detection cost function (minDCF). This metric represents the cost we would pay if we knew before-hand the optimal threshold for the evaluation set (in our case the validation set). We also consider the effective prior in many models, which does not take into account only the prior probability of the authentic fingerprint class, but also the cost of the errors. According to the fact that C_{fp} is greater than C_{fn}, our classifiers behave as if the prior is around 0.1, which is our effective prior that we can compute as:

$$\tilde{\pi} = \frac{\pi_T C_{fn}}{\pi_T C_{fn} + (1 - \pi_T) C_{fp}}$$

In general, we expect that working points with equal costs and unbalanced prior like 0.8 or 0.9 will behave definitely better because of the different costs.
Once we have selected the model that behaves better and the optimal values for the model hyperparameters, we will build our final model by retraining using the whole training set. Considering that the dataset is small enough, also a Leave One Out approach (which is a particular case of K-Fold, with K=N, where N is the number of samples) could have been adopted, but it requires training so many models, which is time consuming and computationally expensive. Note that different preprocessing techniques (such as Z-normalization and PCA) have been employed.

3.2 Gaussian Models

3.2.1 Models explanation

First of all let's consider the Gaussian Models, which are generative models, since they make assumptions on the distribution of the data. In particular, all these models assume that we can use a Gaussian distribution to describe our data, given the class:

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

One further assumption that we can make is the Naive Bayes assumption, which supposes that the different components are independently distributed and is particularly useful when our samples have lots of dimensions, because allows us to reduce the number of parameters that we need to estimate. For what that regards the Naive Bayes Gaussian classifier, it corresponds to a MVG classifier where all the different classes have diagonal covariance matrices (this is valid when we make Gaussian assumption and does not hold for a generic density). But having diagonal covariance matrices, means that different features are uncorrelated. However, since we are talking about Gaussian densities, uncorrelation implies independence, which means that different features are independent.
This means that we must be careful, considering that in practice we will almost never have features that are all independent of each other.
The Tied version makes instead the assumption that the covariance matrices of different classes are the same (are tied):

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

So, each class has its own mean μ_c, but the covariance matrix is the same for all classes and is Σ. With the Tied model we obtain exactly the same classification rules that we obtain using LDA as a classifier if we select the right threshold (it is not a coincidence that the covariance matrix that we compute in the Tied model corresponds to the within class covariance matrix that we compute in LDA).
We can also combine the Naive Bayes assumption with the tied covariance one, in order to obtain the Naïve Tied Gaussian model. In this case, we will have a single diagonal covariance matrix.

3.2.2 Expectations

In general, as for each assumption that we make, the model will perform well if the data behaviour is similar to the assumptions that we make, otherwise will not perform particularly well. Looking at the covariance matrices of the two classes, we observe that the authentic fingerprint class has a covariance matrix that is almost diagonal whereas the spoofed fingerprint class shows some correlation between features. Therefore, to capture this correlation, a MVG model could be more appropriate in our situation. But for a MVG model, we must pay attention to the fact that the authentic fingerprint class has few samples. It is also worth trying models that have a lower amount of parameters, such as Naive Bayes and Tied models, since they may benefit from the lower risk of overfitting. Anyway, the Tied assumptions seem quite inaccurate (as we said before, probably linear classifiers are not that good).

In fact, if we consider that the generic decision rule for a Gaussian Model is:

$$s(\boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x} + \boldsymbol{x}^T \boldsymbol{b} + c$$

where, defined the precision matrix $\Lambda = \Sigma^{-1}$, we have:

$$\boldsymbol{A} = -\frac{1}{2}\left(\boldsymbol{\Lambda_1} - \boldsymbol{\Lambda_0}\right)$$

$$\mathbf{b} = \left(\boldsymbol{\Lambda_1}\boldsymbol{\mu_1} - \boldsymbol{\Lambda_0}\boldsymbol{\mu_0}\right)$$

$$\mathrm{c} = -\frac{1}{2}\left(\boldsymbol{\mu_1^T}\boldsymbol{\Lambda_1}\boldsymbol{\mu_1} - \boldsymbol{\mu_0^T}\boldsymbol{\Lambda_0}\boldsymbol{\mu_0}\right) + \frac{1}{2}\left(\log|\boldsymbol{\Lambda_1}| - \log|\boldsymbol{\Lambda_0}|\right)$$

and we have equals covariance matrices, we will obtain the following decision rule according to the definition of matrix A:

$$s(\boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{b} + c$$

But in our scenario the covariance matrices for the 2 classes are not really similar and this represents the reason why our model does not show a good behaviour. Focusing on the Naive Bayes assumption, since some features are more or less correlated, we expect that the elements which are not on the main diagonals of the covariance matrices are slightly different from zero. So, it is not guaranteed that Tied and Naive Bayes models will behave in a good way. Consequently, we do not have great expectations also for the Naïve Tied Gaussian model.

3.2.3 Results and considerations

Now, let's visualize the results that we obtained in the different situations:

	MVG Classifier		
PCA	$\boldsymbol{\pi = 0.5}$	$\boldsymbol{\pi = 0.1}$	$\boldsymbol{\pi = 0.9}$
-	0.331	0.615	0.111
9	0.330	0.629	0.109
8	0.333	0.612	0.109
7	0.341	0.618	0.114
6	0.335	0.612	0.109
5	0.359	0.651	0.115

	Naive Gaussian Classifier		
PCA	$\boldsymbol{\pi = 0.5}$	$\boldsymbol{\pi = 0.1}$	$\boldsymbol{\pi = 0.9}$
-	0.472	0.806	0.144
10*	0.370	0.722	0.113
9	0.369	0.740	0.112
8	0.360	0.712	0.112
7	0.361	0.717	0.115
6	0.360	0.721	0.116
5	0.371	0.774	0.114

	Tied MVG Classifier		
PCA	$\boldsymbol{\pi = 0.5}$	$\boldsymbol{\pi = 0.1}$	$\boldsymbol{\pi = 0.9}$
-	0.486	0.706	0.184
9	0.492	0.696	0.180
8	0.485	0.686	0.181
7	0.484	0.670	0.183
6	0.483	0.730	0.181
5	0.490	0.706	0.192

	Tied Naive Gaussian Classifier		
PCA	$\boldsymbol{\pi = 0.5}$	$\boldsymbol{\pi = 0.1}$	$\boldsymbol{\pi = 0.9}$
-	0.551	0.790	0.198
10*	0.533	0.754	0.198
9	0.543	0.771	0.202
8	0.544	0.774	0.201
7	0.541	0.769	0.199
6	0.549	0.780	0.201
5	0.530	0.786	0.207

Table 1: Gaussian Models

As we can see, the results are not so far from what we expected. For what that regards PCA, it seems to be really useful for the Naïve Gaussian classifier and partially beneficial for the Naïve Tied model, considering that allows us definitely to obtain a lower minDCF. Applying PCA with MVG and the Tied classifiers overall does not bring significant improvement. In general, the performance starts decreasing when we reduce to 5 or less dimensions, because we start losing information. Since it is required a large amount of time to train all the models, from now on we will focus on using just 8 PCA dimensions. We also apply the diagonalization transformation without actually reducing the dimensionality (PCA with m=10, see Table 1), since our PCA implementation also diagonalizes the dataset covariance matrix, and this already gives us a nice improvement in performance. As we said before, the tied model is a linear model that can separate the classes, but it shows significant worse performance than MVG. Focusing our attention on Naïve Tied Gaussian classifier, we notice that we have a further performance deterioration, but we could have expected that since Tied model do not behave really well. Note that we have showed up only the results using raw features because using Z-normalization we encountered a worsening of performances (even though the results are not so distant from those we obtained with raw features). In our tables we can visualize the values of minDCF for different priors and, as we expected, the cost is lower when we have as prior for the authentic fingerprint class $\boldsymbol{\pi_T} = \mathbf{0.9}$, because this decreases the number of errors with an high cost (false positive). In the following we will still take into account different working points, but for all the other models we expect to obtain better results if $\boldsymbol{\pi_T} = \mathbf{0.9}$. So, we will select the best values of the hyperparameters (in combination with eventual preprocessing techniques) considering the target application and then we will show up the different working points only for the most favourable combination of the models.
So, for the moment, the best model is the MVG classifier with Full Covariance matrices with 9 PCA dimensions, which gives us 0.330 as minimum cost for the target application (under the K-fold protocol).

3.3 Logistic Regression

3.3.1 Models explanation

Now, let's consider Logistic Regression, which is a discriminative approach for classification, even though its name refers to regression. Differently from generative models, logistic regression makes assumptions on the separation rule, and not on the distribution of the data. We will consider linear and quadratic logistic regression using a prior-weighted version of the model, which allows us to take into account the fact that classes are unbalanced and simulate different priors π_T.
The objective function is:

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

where λ is the regularization coefficient and is an hyperparameter that should be selected as to optimize the performance of the classifier. Furthermore, we have to say that this hyperparameter λ is contained in the regularization term, which favors simpler solution and reduces the risk of overfitting. In fact, if λ is too small, we will get a solution that has good separation on the training set, but may have a bad behaviour for unseen data whereas, if λ is too large, we will obtain a solution that is not able to well separate the classes. To select the best value for λ, we used as value for π_T the effective prior of our target application, which takes into consideration also the costs of errors.
To compute the score, if we assume that decision rules are linear surfaces, we exploit the following formula:

$$s(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b$$

where:

- $\boldsymbol{w}$ is a vector that is orthogonal to the decision rule;
- $\boldsymbol{b}$ is the bias term, which tells us the required horizontal displacement from the hyperplane.

So, to define our decision rule, we need to find the values for the model parameters "w" and "b". However, the objective function leads to linear decision surfaces. In order to obtain non-linear separation rules, we can train a logistic regression model using, instead of x, feature vectors $\phi(x)$, defined as:

$$\phi(\boldsymbol{x}) = \begin{bmatrix} \text{vec}\left(\boldsymbol{x}\boldsymbol{x}^T\right) \\ \boldsymbol{x} \end{bmatrix}$$

In this way, we will obtain linear separation surfaces in the space defined by the mapping ϕ that actually correspond to quadratic forms in the original feature space.
In this case the posterior log-likelihood ratio will be expressed as:

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

3.3.2 Expectations

As we said previously, the linear model will presumably perform poorly, but we will try it anyway, even if we have already seen a bad behaviour for the Tied MVG classifier. The quadratic version, instead, probably is more appropriate for our purpose.

3.3.3 Linear Logistic Regression

We can make a comparison using different values of λ:

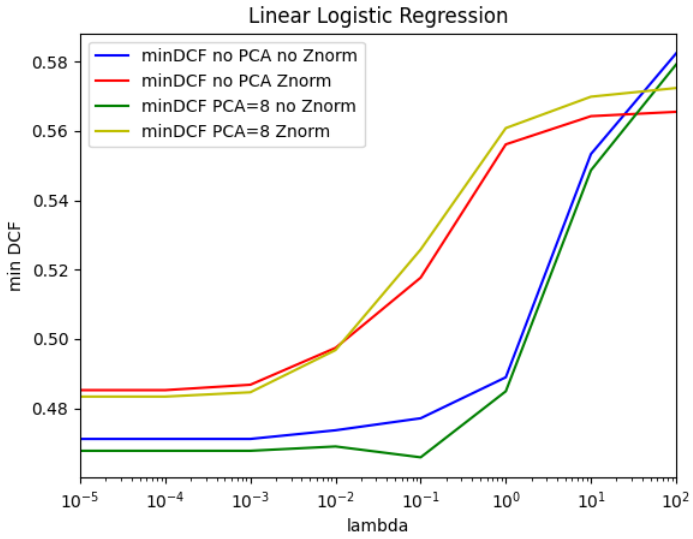


Figure 8: Linear LR

In general, we observe that ZNorm makes things worse. So, the best configurations that we consider are those with no ZNorm and, in particular, we obtain the best results using PCA. As we can see, the best combination is given by $\lambda = 0.1$ with 8 PCA dimensions and without applying ZNorm. Overall, the MVG model with full covariances and the Naive Gaussian classifier perform better than linear logistic regression.

3.3.4 Quadratic Logistic Regression

We repeat the same analysis for quadratic logistic regression, considering the same range of values for λ that we analyzed in the linear case:

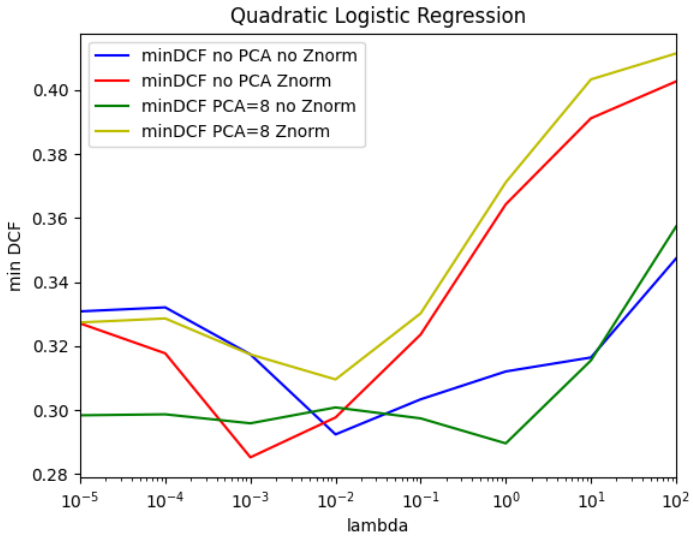


Figure 9: Quadratic LR

In contrast with linear LR, quadratic LR shows up a better behaviour according to what we said previously, in fact results have significantly improved. In this case, we obtain the best result for $\lambda = 10^{-3}$, applying ZNorm and without applying PCA. So, the quadratic logistic regression model outperforms the MVG classifier, which until now was the model with the best behaviour.

3.3.5 Results and considerations

As expected, for both these two models we obtain better results when we have as prior for the authentic fingerprint class $\boldsymbol{\pi_T = 0.9}$, because this decreases the number of errors with an high cost. For both linear and quadratic LR we notice that, when the value of λ is too large, the model struggles to correctly classify the samples. At this point we have proof that on our dataset quadratic models behaves better than linear ones, since the Tied MVG and the linear LR, which are linear models, perform worse than the MVG and Naive classifiers, which are quadratic models.

Linear Logistic Regression ($\boldsymbol{\lambda = 10^{-1}}$)			
PCA = 8 and no Znorm			
	$\boldsymbol{\pi = 0.5}$	$\boldsymbol{\pi = 0.1}$	$\boldsymbol{\pi = 0.9}$
LR(π_T = effective prior $\tilde{\pi}$)	0.466	0.705	0.183
LR($\pi_T = 0.5$)	0.478	0.716	0.188
LR($\pi_T = 0.9$)	0.510	0.719	0.194
LR($\pi_T = 0.1$)	0.469	0.705	0.183

Quadratic Logistic Regression ($\boldsymbol{\lambda = 10^{-3}}$)			
no PCA and Znorm			
	$\boldsymbol{\pi = 0.5}$	$\boldsymbol{\pi = 0.1}$	$\boldsymbol{\pi = 0.9}$
QLR(π_T = effective prior $\tilde{\pi}$)	0.285	0.593	0.104
QLR($\pi_T = 0.5$)	0.304	0.613	0.103
QLR($\pi_T = 0.9$)	0.314	0.647	0.104
QLR($\pi_T = 0.1$)	0.288	0.594	0.105

So, up to now, the best model is the quadratic LR classifier with $\lambda = 10^{-3}$, applying ZNorm and without applying PCA. According to the results reported on the tables, we notice that is better to use as training prior the effective prior, because it leads to the best results.

3.4 Support Vector Machines

3.4.1 Models explanation

We now turn our attention to Support Vector Machines, which is a non-probabilistic model, since the produced score has not a probabilistic interpretation. These models select the hyperplane that separates the classes with the largest margin. So, instead of choosing the hyperplane that maximize class probabilities like Logistic Regression, this model picks the one that maximizes the margin (distance of the closest point from the selected hyperplane). For this model we consider two different formulations:

- The primal formulation is employed for linearly separable classes and used in the linear SVM model. The related objective function to minimize is:

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

  where:

$$\widehat{\boldsymbol{x}}_i = \begin{bmatrix} \boldsymbol{x}_i \\ K \end{bmatrix} \qquad \widehat{\boldsymbol{w}} = \begin{bmatrix} \boldsymbol{w} \\ b \end{bmatrix}$$

  The two hyperparameters on which we focus on are **C**, that is crucial to control the trade-off between maximizing the margin and minimizing the training error, and **K**, that is used to mitigate the regularization effect of the bias term (as K becomes larger, the regularizing effect of b becomes weaker), due to the regularization of the norm of $\widehat{\boldsymbol{w}}$:

$$\|\widehat{\boldsymbol{w}}\|^2 = \|\boldsymbol{w}\|^2 + b^2$$

- The dual formulation is employed for non-linearly separable classes and used for polynomial SVM and Radial Basis Function SVM models. This formulation consists in the maximization of the objective function:

$$L_D(\boldsymbol{\alpha}) = \alpha^T\mathbf{1} - \frac{1}{2}\boldsymbol{\alpha}^T\boldsymbol{H}\boldsymbol{\alpha}$$

  subject to the following constraints:

$$0 \leq \alpha_i \leq C_i, i = 1, \ldots, n$$

$$\sum_{i=1}^{n}\alpha_i z_i = 0$$

  This formulation is able to embed a non–linear transformation by computing efficiently the dot–products in the expanded space through a kernel function without explicitly computing the new space:

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

  It order to add a regularized bias in the dual formulation, we can add a constant value $\xi = K^2$ to the kernel function:

$$\widehat{k}\left(x_1, x_2\right) = k\left(x_1, x_2\right) + \xi$$

  Where $k\left(x_1, x_2\right)$ will be, depending on the case, the kernel function of polynomial SVM or Radial Basis Function SVM.

It is also possible to consider a balanced version of the models by multiplying the hyperparameter **C** by two factors (that depend on the class). We have:

$$\mathrm{C_i} = \mathrm{C}\frac{\pi_T}{\pi_T^{emp}} \qquad \mathrm{C_i} = \mathrm{C}\frac{\pi_F}{\pi_F^{emp}}$$

where i represent the i-th sample belonging to authentic or spoofed class and π_T^{emp} and π_F^{emp} represent respectively the empirical priors of the authentic class and spoofed class calculated on the training set.

3.4.2 Expectations

Since our classes have proved to be effectively separable with non-linear decision rule, we expect that the linear SVM will perform poorly, while instead Polynomial SVM and RBF (Radial Basis Function) SVM will perform better. For this reason, we will firstly consider the unbalanced version of the models in order to find the best ones and, then, apply class balancing only where it's worth doing it.

3.4.3 Linear SVM

Since, as we said above, this model uses primal formulation, we try to tune the hyperparameter **C** and **K**.

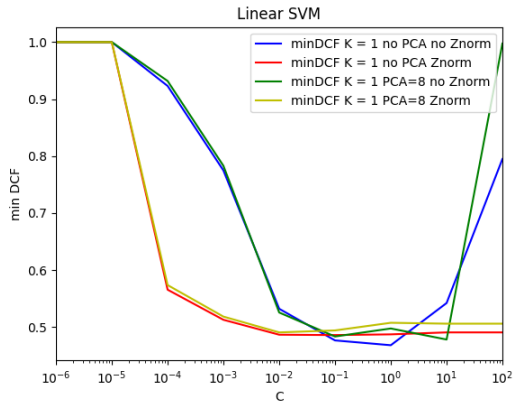


Figure 10: Linear SVM with K = 1

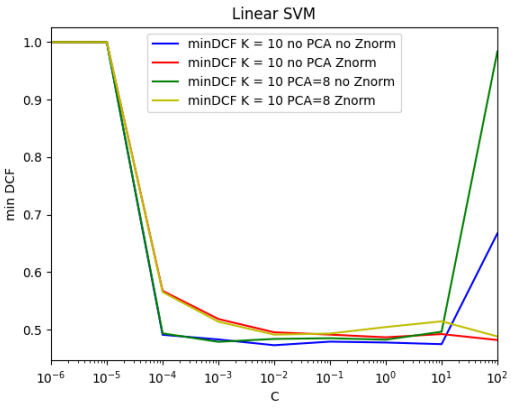


Figure 11: Linear SVM with K = 10

For what that regards the preprocessing techniques, for this model the best configuration seems to be the one without any preprocessing, so both PCA and ZNorm are not applied. The choice of the hyperparameter K and C was not trivial. We report the best combinations of the two hyperparameters in order to make a decision.

Linear SVM		
no PCA no ZNorm		
$\boldsymbol{K=1 \quad C=1}$	$\boldsymbol{K=10 \quad C=10^{-2}}$	$\boldsymbol{K=10 \quad C=10}$
0.468	0.473	0.475

It seems that mitigating the regularization effect of the bias term too much is counterproductive, so we will keep $K = 1$ both for primal and dual formulation. Fixed $K = 1$, the best value for C is $C = 1$.

3.4.4 Polynomial SVM

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

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

where **c** and **d** are two hyperparameters that represent the coefficient of the polynomial and its degree. We will exploit the polynomial SVM with a degree 2 (so it will be quadratic SVM). Then we try to tune **c** and **C**.

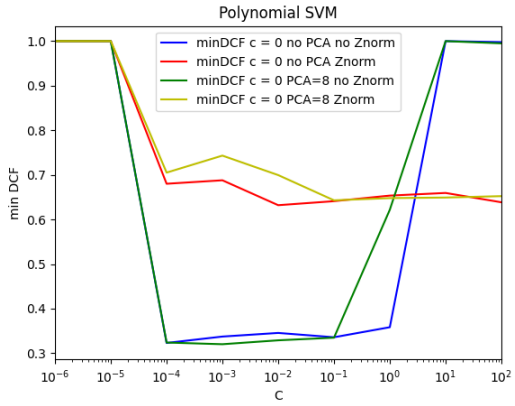


Figure 12: Polynomial SVM with c = 0

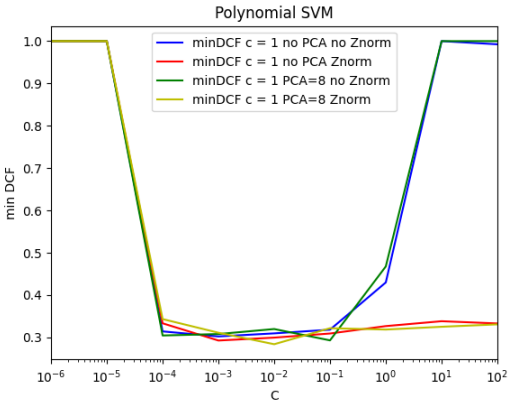


Figure 13: Polynomial SVM with c = 1

As we can see from the plots, $c = 1$ is clearly the best choice for the model. For what concerns preprocessing techniques, since it is not so immediate, we report the best ones in combination with the values of the hyperparameters, in order to find the optimal strategy.

Polynomial SVM		
no PCA ZNorm	PCA = 8 ZNorm	PCA = 8 no ZNorm
$\boldsymbol{C = 10^{-3}}$	$\boldsymbol{C = 10^{-2}}$	$\boldsymbol{C = 10^{-1}}$
0.293	0.284	0.293

Hence, the best configuration seems to be the one in which both PCA and ZNorm are applied, and the best value for C is $C = 10^{-2}$.

3.4.5 Radial Basis Function SVM

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

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

where $\boldsymbol{\gamma}$ is an hyperparameter that defines the width of the kernel. With a small γ, we have a wide kernel where a support vector influences most other points, while with a large γ, we have a narrow kernel where a support vector has very small influence on points that are not close. So, we want to tune γ.

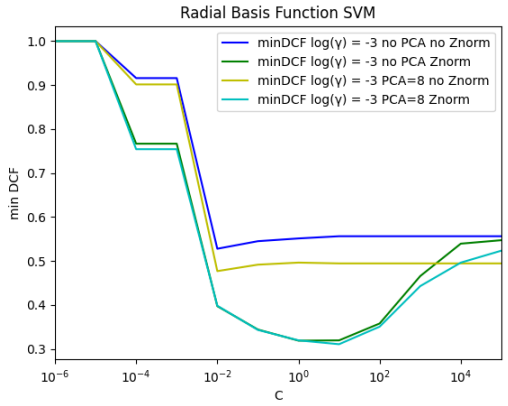


Figure 14: RBF SVM with $\log(\gamma) = -3$

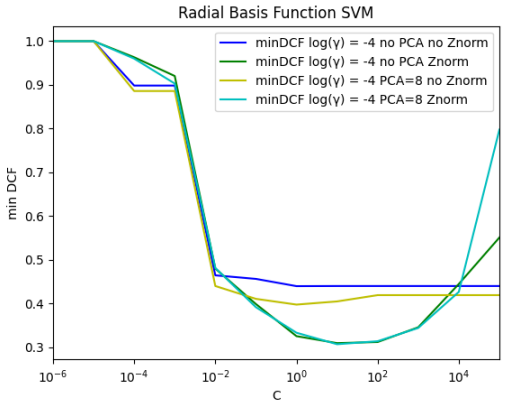


Figure 15: RBF SVM with $\log(\gamma) = -4$

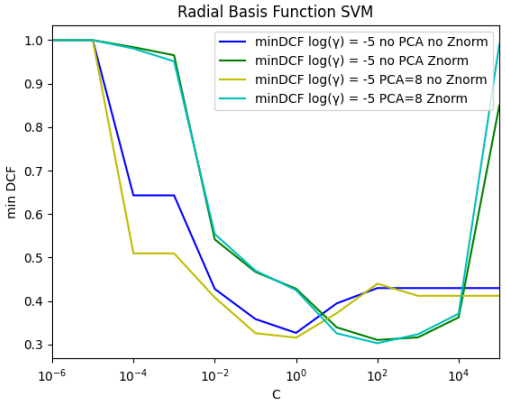


Figure 16: RBF SVM with $\log(\gamma) = -5$

Here in all cases seems that the best pre-processing techniques are no PCA + ZNorm and PCA + ZNorm. So, in order to find the optimal value for γ, we report the minimum values with these techniques.

Radial Basis Function SVM			
		PCA = 8 ZNorm	no PCA ZNorm
$\boldsymbol{C = 10}$	$\boldsymbol{\log(\gamma) = -3}$	0.311	0.320
$\boldsymbol{C = 10}$	$\boldsymbol{\log(\gamma) = -4}$	0.307	0.310
$\boldsymbol{C = 10^2}$	$\boldsymbol{\log(\gamma) = -5}$	0.303	0.310

Therefore, the best configuration seems to be the one in which both PCA and ZNorm are applied and the best value for C is $C = 10^2$, while for γ is $\log(\gamma) = -5$, so $\gamma = \frac{1}{e^5}$.

3.4.6 Results and considerations

As we expected previously, linear SVM model performs poorly on these data and is the worst model among the SVM models. Given that, we now try to apply class balancing only on the most promising models (so we don't consider linear SVM) to see if there is any improvement.

Class-Balanced Polynomial SVM			
$\boldsymbol{K=1 \quad c=1 \quad C=10^{-2} \quad d=2}$			
PCA = 8 Znorm			
	$\boldsymbol{\pi=0.5}$	$\boldsymbol{\pi=0.1}$	$\boldsymbol{\pi=0.9}$
π_T = effective prior $\tilde{\pi}$	0.297	0.592	0.096
$\pi_T = 0.5$	0.285	0.621	0.097
$\pi_T = 0.9$	0.300	0.613	0.096
$\pi_T = 0.1$	0.296	0.589	0.096

Class-Balanced RBF SVM			
$\boldsymbol{K=1 \quad \gamma=1/e^5 \quad C=10^2}$			
PCA = 8 Znorm			
	$\boldsymbol{\pi=0.5}$	$\boldsymbol{\pi=0.1}$	$\boldsymbol{\pi=0.9}$
π_T = effective prior $\tilde{\pi}$	0.315	0.580	0.107
$\pi_T = 0.5$	0.300	0.608	0.098
$\pi_T = 0.9$	0.297	0.602	0.099
$\pi_T = 0.1$	0.322	0.586	0.107

As we previously said, our expectations are confirmed for $\pi = 0.9$. Furthermore, it seems that class balancing is not effective, so we will not consider it. So, until now the best model is Polynomial SVM with $c = 1$, $C = 10^{-2}$ and with both PCA and ZNorm applied.

3.5 Gaussian Mixture Models

3.5.1 Models explanation

Finally, we go back to generative approaches and consider the Gaussian Mixture Models, which tend to be used not just to perform classification (we train a GMM over the data of each class), but also for clustering (K-means clustering) and density estimation (we can model a generic distribution with a GMM). In general, a GMM is a density model obtained as a weighted combination of Gaussian:

$$f_{\boldsymbol{X}}(\boldsymbol{x}) = \sum_{c=1}^{K} w_c \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)$$

where the distribution parameters are:

- $\boldsymbol{M} = [\boldsymbol{\mu}_1 \dots \boldsymbol{\mu}_K]$, which are the component means;
- $\boldsymbol{S} = [\boldsymbol{\Sigma}_1 \dots \boldsymbol{\Sigma}_K]$, which are the component covariances;
- $\boldsymbol{w} = [w_1 \dots w_K]$, which are the weights.

3.5.2 Expectations

From the analysis of the dataset and from what we have seen in the section regarding the analysis of gaussian models, we infer that some of these models can behave in a good way (in particular GMM and Diagonal GMM). Especially, since GMMs can model generic distributions, we expect to obtain better results than those that we obtained with Gaussian models. Authentic fingerprint class samples seem to be well described by Gaussian densities. The spoofed fingerprint class, instead, is composed by different sub-classes. So, for the target and non-target classes we take into account models with different number of components. As we can imagine, to provide good models, for the authentic fingerprint class may be sufficient few components (such as one or two), considering that this class has a gaussian-like shape, whereas for the spoofed fingerprint class we may need a greater number of components. In fact, from the scatter plot we have already noticed that samples of spoofed fingerprint class seem to correspond to different clusters, which probably represent the six different spoofing approaches. Hence, we expect to obtain the best results if we have a number of components between four and eight (considering that usually the number of components of these type of models is a power of two, so we won't take into account five, six or seven components) for spoofed class and one or two components for the authentic class. We expect the two diagonal models to produce good result because the features that we have do not show a significant correlation (as we have seen in the heatmaps).

3.5.3 Results and considerations

We evaluate the performance testing various combinations of number of components for the two classes and considering both full covariance and diagonal models, with and without covariance tying. Please note that for tied covariance models tying takes place at class level (which means that different classes have different covariance matrices). For these models the number of components of each GMM represents the hyperparameter to tune and, first of all, we focus on raw features:

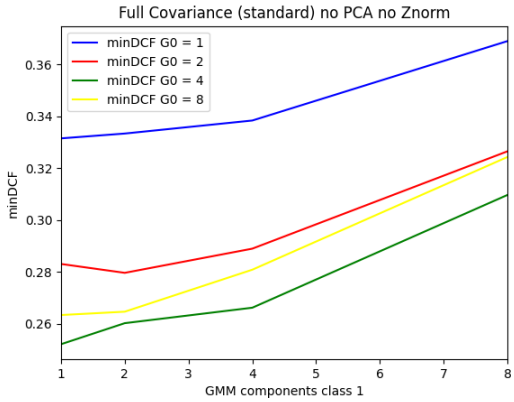


Figure 17: Full Covariance GMM

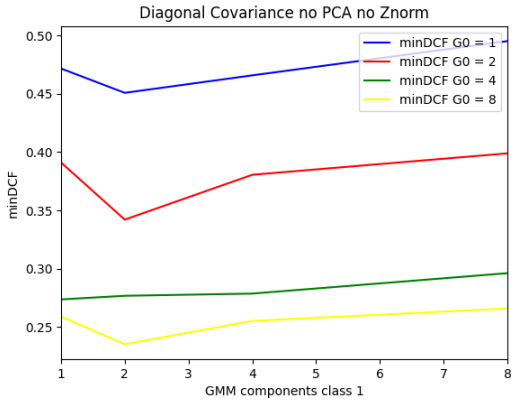


Figure 18: Diagonal Covariance GMM

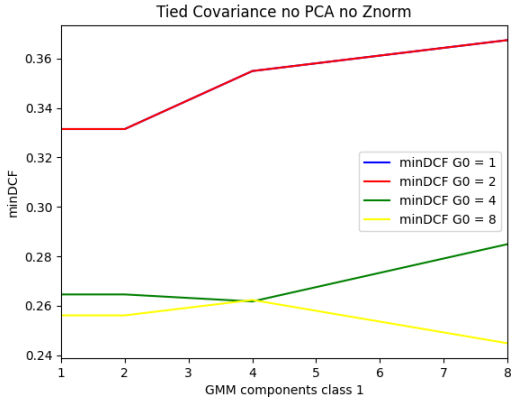


Figure 19: Tied Covariance GMM

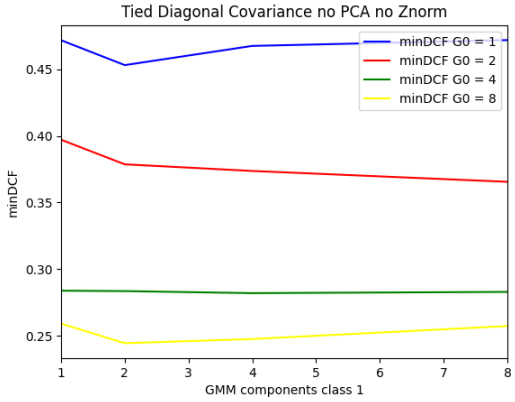


Figure 20: Tied Diagonal Covariance GMM

			minDCF		
RAW GMM Model	G0	G1	$\pi = 0.5$	$\pi = 0.1$	$\pi = 0.9$
Full Covariance	4	1	0.252	0.471	0.085
Diagonal Covariance	8	2	0.235	0.565	0.085
Tied Covariance	8	8	0.245	0.551	0.085
Tied Diagonal Covariance	8	2	0.245	0.518	0.081

As we can see, the Gaussian Mixture Models give us good results and the best one is the Diagonal with G0=8 and G1=2 (with raw features). As predicted, the number of components for each class is congruous with our expectations. Furthermore, we have to say that the Diagonal GMM probably behaves well because our features do not have a notable correlation. As expected, also in this case we obtain better results if $\boldsymbol{\pi_T = 0.9}$.
For what that regards preprocessing techniques, we tried out PCA, ZNorm and PCA + ZNorm, but none of them led us to significant improvements of the results. In particular, the diagonal model gets worse if combined with preprocessing strategies and, instead, performs at his best when trained on raw features.
To sum up, our best models for each family are:

	minDCF		
Model	$\pi = 0.5$	$\pi = 0.1$	$\pi = 0.9$
MVG (PCA=9)	0.330	0.629	0.109
QLR (ZNorm no PCA; $\lambda = 10^{-3}$; π_T = effective prior)	0.285	0.593	0.104
Pol SVM (ZNorm and PCA; $K = 1$; $d = 2$; $c = 1$; $C = 10^{-2}$)	0.284	0.606	0.095
Diagonal GMM (RAW; G0=8; G1=2)	0.235	0.565	0.085

So, up to now, the best model is is the diagonal one trained on raw features with 8 components for the class 0 and 2 for the class 1.

3.6 Score Calibration

Until now, we have considered as metric to evaluate the performance of the models the minDCF, so we have considered the cost that we would pay if we knew in advance the optimal threshold. Now, we also consider another metric, known as actualDCF (actDCF), that is obtained using the threshold corresponding to the effective prior $\tilde{\pi}$, and we focus on the difference between actDCF and minDCF, that represents the loss due to miscalibrated scores. The approach we adopted to minimize this difference and, therefore, to calibrate the scores, is to use prior weighted logistic regression model and subtract the optimal threshold to the scores. In this way we optimize the calibration specifically for our application with prior $\tilde{\pi}$. The calibrated scores are retrieved through the formulation:

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

Since, thanks to minDCF evaluation, we have already found the best models, we will consider score calibration only for the best ones, that are Quadratic Logistic Regression, Polynomial SVM and GMM Diagonal Covariance. To see if the models are calibrated we use the Bayes Error Plot, who shows actualDCF and minDCF (more distance between them, more miscalibration). For the sake of simplicity we will simply refer to them as QLR, SVM and GMM. Furthermore, we expect the polynomial SVM to be highly miscalibrated, since the model scores don't have a probabilistic interpretation.

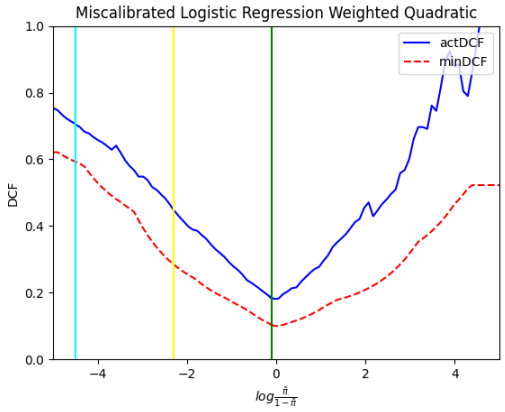


Figure 21: Bayes Error Plot QLR Miscalibrated

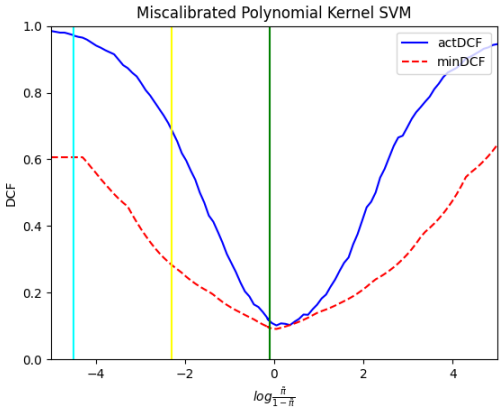


Figure 22: Bayes Error Plot SVM Miscalibrated

Yellow Line: $(\pi_T = 0.5, C_{fn} = 1, C_{fp} = 10)$ application working point
Cyan Line: $(\pi_T = 0.1, C_{fn} = 1, C_{fp} = 10)$ working point
Green Line: $(\pi_T = 0.9, C_{fn} = 1, C_{fp} = 10)$ working point.

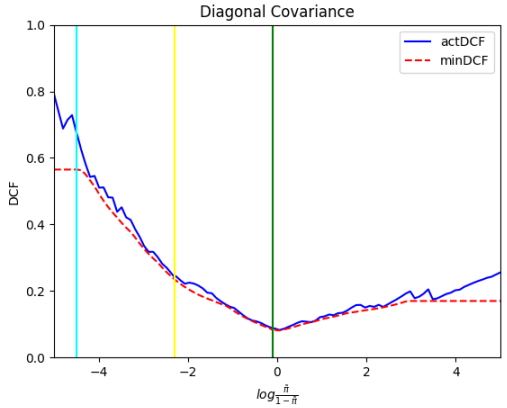


Figure 23: Bayes Error Plot GMM without score calibration

Yellow Line: $(\pi_T = 0.5, C_{fn} = 1, C_{fp} = 10)$ application working point
Cyan Line: $(\pi_T = 0.1, C_{fn} = 1, C_{fp} = 10)$ working point
Green Line: $(\pi_T = 0.9, C_{fn} = 1, C_{fp} = 10)$ working point.

As we can see from the plots, QLR is quite miscalibrated and may require recalibration of the scores. Polynomial SVM, as we predicted, is the most miscalibrated model due to the geometrical interpretation, while the GMM Diagonal Covariance model is already well-calibrated, so it's not particularly useful to apply score calibration on this model. Now we apply the score calibration.

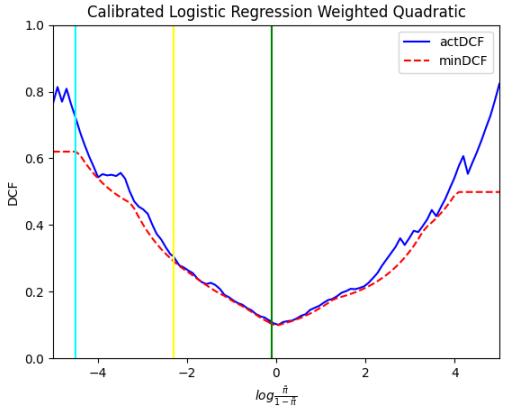


Figure 24: Bayes Error Plot QLR Calibrated

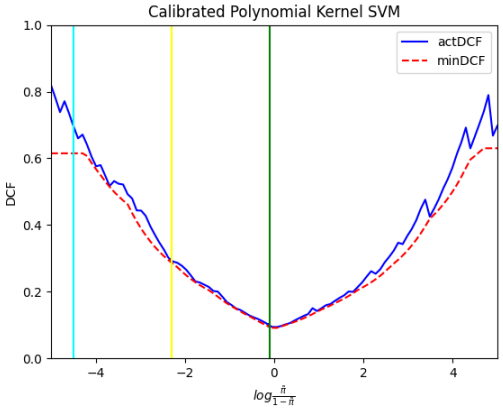


Figure 25: Bayes Error Plot SVM Calibrated

Yellow Line: $(\pi_T = 0.5, C_{fn} = 1, C_{fp} = 10)$ application working point
Cyan Line: $(\pi_T = 0.1, C_{fn} = 1, C_{fp} = 10)$ working point
Green Line: $(\pi_T = 0.9, C_{fn} = 1, C_{fp} = 10)$ working point.

From these plots we observe that there is an evident improvement, but in addition we show up the actDCF and minDCF values before and after score calibration:

QLR Model							
actDCF	$\pi = 0.5$	$\pi = 0.1$	$\pi = 0.9$	minDCF	$\pi = 0.5$	$\pi = 0.1$	$\pi = 0.9$
Miscalibrated	0.449	0.706	0.185	Miscalibrated	0.285	0.593	0.104
Calibrated	0.306	0.725	0.109	Calibrated	0.292	0.620	0.104

SVM Model							
actDCF	$\pi = 0.5$	$\pi = 0.1$	$\pi = 0.9$	minDCF	$\pi = 0.5$	$\pi = 0.1$	$\pi = 0.9$
Miscalibrated	0.690	0.972	0.116	Miscalibrated	0.284	0.606	0.095
Calibrated	0.294	0.697	0.100	Calibrated	0.287	0.615	0.095

GMM Model							
actDCF	$\pi = 0.5$	$\pi = 0.1$	$\pi = 0.9$	minDCF	$\pi = 0.5$	$\pi = 0.1$	$\pi = 0.9$
Calibrated	0.242	0.678	0.088	Calibrated	0.235	0.565	0.085

Also here, our expectations have been confirmed for $\pi = 0.9$. We expect the same outcome also for model fusion.

3.7 Model Fusion

Exploiting the same formulation used before for the score calibration, it is also possible to combine different classifiers and their decision in order to improve the overall performance. As for score calibration, this technique will be applied only to the most promising models. We stack the scores of the models together and then we apply score calibration by computing the final score as a weighted combination of the scores of the combined models, rather than taking a majority voting. In this way, we expect that the models that perform better will compensate the worst models, so we expect that the GMM will compensate for QLR/SVM.

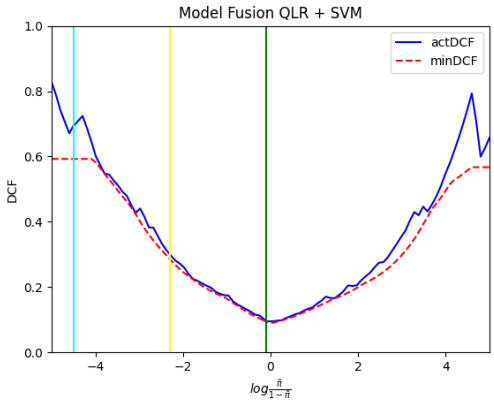


Figure 26: Bayes Error Plot of QLR and SVM fusion

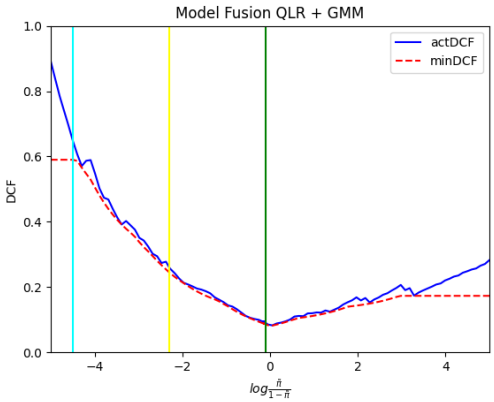


Figure 27: Bayes Error Plot of QLR and GMM fusion

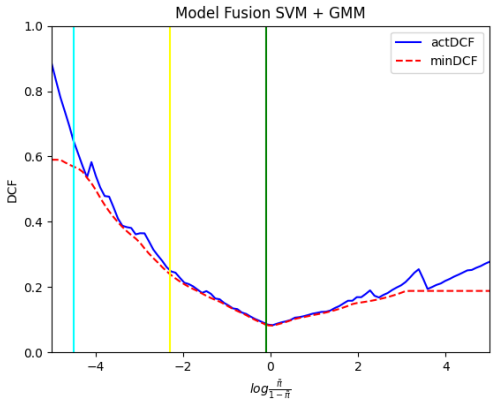


Figure 28: Bayes Error Plot of SVM and GMM fusion

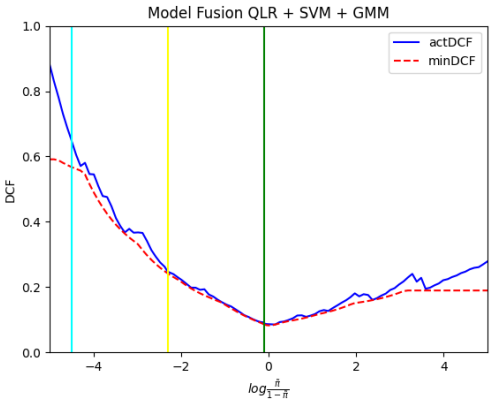


Figure 29: Bayes Error Plot of QLR, SVM and GMM fusion

Yellow Line: $(\pi_T = 0.5, C_{fn} = 1, C_{fp} = 10)$ application working point
Cyan Line: $(\pi_T = 0.1, C_{fn} = 1, C_{fp} = 10)$ working point
Green Line: $(\pi_T = 0.9, C_{fn} = 1, C_{fp} = 10)$ working point.

These plots already show us a good performance, but we report also the minDCF for each model, in order to find the best classifiers.

Model	$\pi = 0.5$	$\pi = 0.1$	$\pi = 0.9$
QLR + SVM	0.286	0.592	0.095
QLR + GMM	0.245	0.590	0.086
SVM + GMM	0.240	0.569	0.085
QLR + SVM + GMM	0.242	0.568	0.086

As we also expected before, we have lower minDCF values for $\pi = 0.9$. From these results, we can see that QLR + GMM, SVM + GMM and QLR + SVM + GMM perform really well and, even better than the other models considered so far. Hence, we consider them as additional classifiers for the evaluation phase.

4 Evaluation

Now we turn our attention to the evaluation phase. Differently from the training phase, here we will consider models that have achieved better performance and reassess some models where it's worth. Below we show up a recap of the candidate models and their best configuration for our purposes:

- QLR, trained with the effective prior, with $\lambda = 10^{-3}$, with ZNorm and without PCA.
- Polynomial Kernel SVM with $c = 1, C = 10^{-2}$, with both ZNorm and PCA applied.
- Diagonal Covariance GMM, with 8 components for the spoofed class and 2 components for the authentic class.
- QLR + GMM model fusion.
- SVM + GMM model fusion.
- QLR + SVM + GMM model fusion.

Once again, for the sake of simplicity, we will refer to them as SVM, QLR and GMM. Furthermore, we will also reconsider hyperparameter values related to these models, to see if our choices are still optimal.

4.1 Score Calibration

As we have done in the training phase, here we consider the best models and check if they are calibrated: if they are not, we calibrate them. In particular, we expect a behaviour quite similar to the one that we had in the training phase (SVM is more miscalibrated due to the geometrical interpretation and also QLR is quite miscalibrated, while GMM should be fine without any further adjustment).

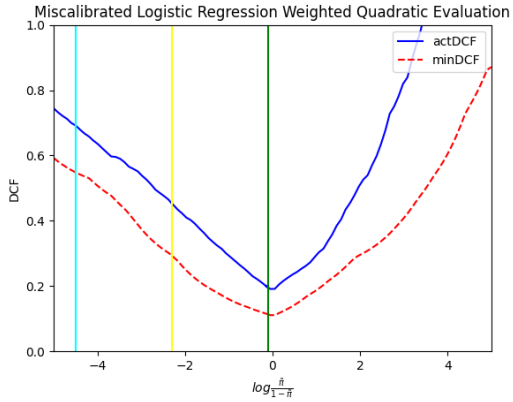


Figure 30: Bayes Error Plot QLR Miscalibrated Evaluation

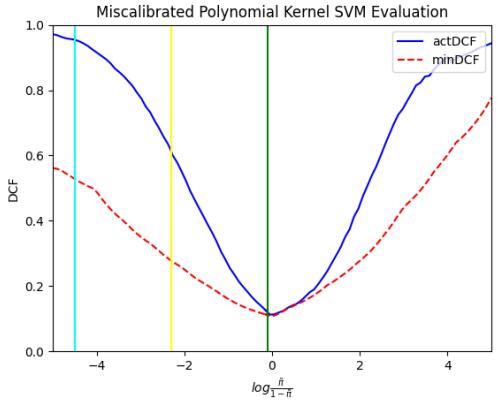


Figure 31: Bayes Error Plot SVM Miscalibrated Evaluation

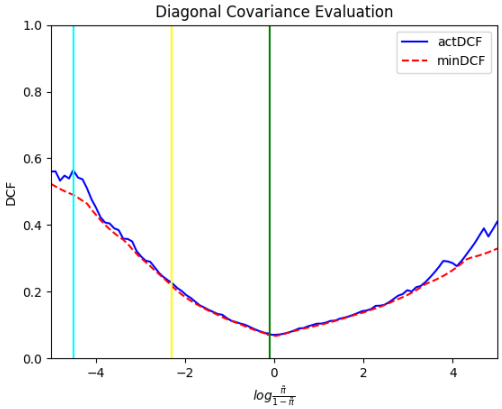


Figure 32: Bayes Error Plot Diagonal Covariance GMM Without Score Calibration Evaluation

Yellow Line: $(\pi_T = 0.5, C_{fn} = 1, C_{fp} = 10)$ application working point
Cyan Line: $(\pi_T = 0.1, C_{fn} = 1, C_{fp} = 10)$ working point
Green Line: $(\pi_T = 0.9, C_{fn} = 1, C_{fp} = 10)$ working point.

As we expected, the most miscalibrated model is the SVM, but also QLR is quite miscalibrated. GMM is already well-calibrated so it's useless to apply score calibration. Hence, we proceed to calibrate SVM and QLR.

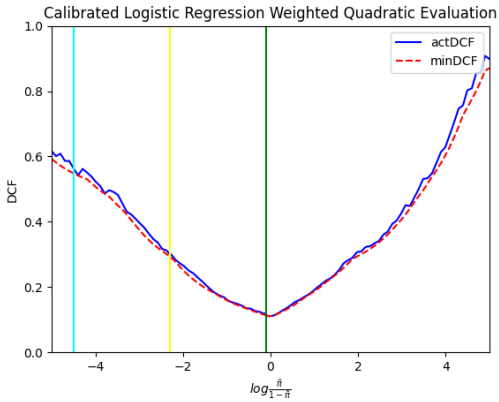


Figure 33: Bayes Error Plot QLR Calibrated Evaluation

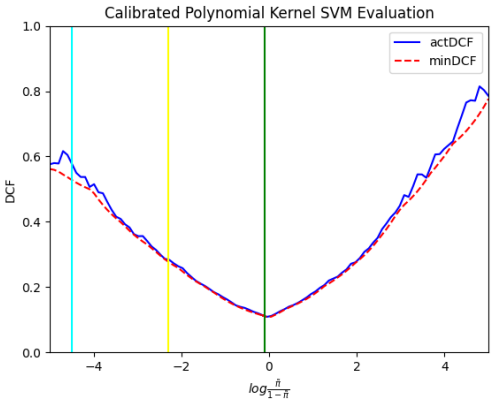


Figure 34: Bayes Error Plot SVM Calibrated Evaluation

Yellow Line: $(\pi_{\mathrm{T}} = 0.5, C_{fn} = 1, C_{fp} = 10)$ application working point
Cyan Line: $(\pi_{\mathrm{T}} = 0.1, C_{fn} = 1, C_{fp} = 10)$ working point
Green Line: $(\pi_{\mathrm{T}} = 0.9, C_{fn} = 1, C_{fp} = 10)$ working point.

We can clearly observe an improvement, but for the sake of completeness we show up the actDCF and minDCF values before and after score calibration:

QLR Model							
actDCF	$\pi = 0.5$	$\pi = 0.1$	$\pi = 0.9$	minDCF	$\pi = 0.5$	$\pi = 0.1$	$\pi = 0.9$
Miscalibrated	0.453	0.692	0.196	Miscalibrated	0.294	0.548	0.114
Calibrated	0.301	0.564	0.116	Calibrated	0.294	0.548	0.114

SVM Model							
actDCF	$\pi = 0.5$	$\pi = 0.1$	$\pi = 0.9$	minDCF	$\pi = 0.5$	$\pi = 0.1$	$\pi = 0.9$
Miscalibrated	0.614	0.955	0.120	Miscalibrated	0.278	0.528	0.111
Calibrated	0.285	0.579	0.111	Calibrated	0.278	0.528	0.111

GMM Model							
actDCF	$\pi = 0.5$	$\pi = 0.1$	$\pi = 0.9$	minDCF	$\pi = 0.5$	$\pi = 0.1$	$\pi = 0.9$
Calibrated	0.228	0.565	0.072	Calibrated	0.218	0.490	0.070

Once again, our expectations have been confirmed for $\pi = 0.9$. We expect the same outcome also for model fusion.

4.2 Model Fusion

Now, let's move towards the analysis of the best fused models and let's see how they perform on the evaluation set.

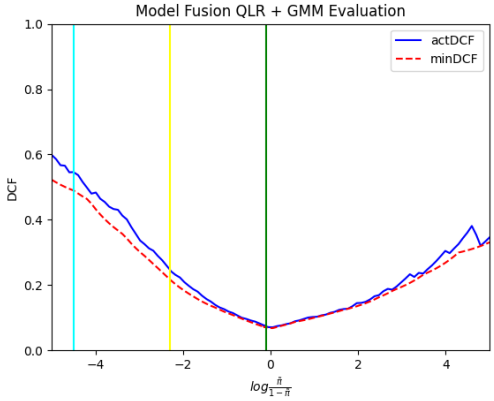


Figure 35: Fusion of QLR and GMM Evaluation

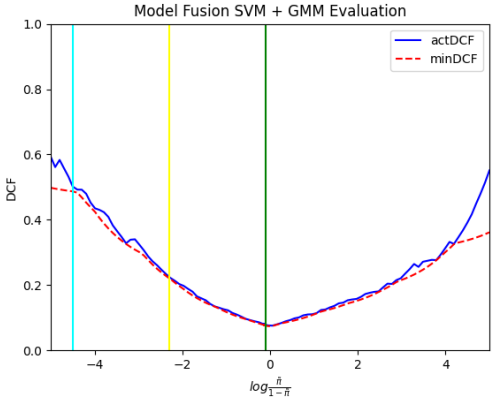


Figure 36: Fusion of SVM and GMM Evaluation

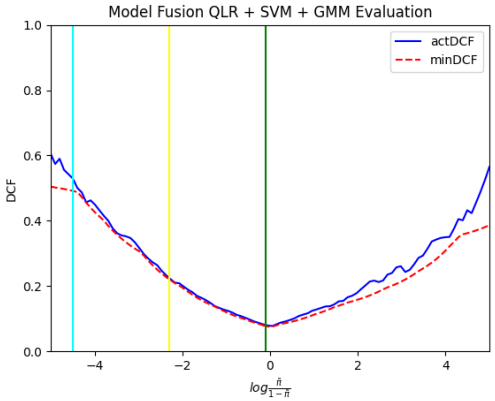


Figure 37: Fusion of QLR, SVM and GMM Evaluation

Yellow Line: $(\pi_T = 0.5, C_{fn} = 1, C_{fp} = 10)$ application working point
Cyan Line: $(\pi_T = 0.1, C_{fn} = 1, C_{fp} = 10)$ working point
Green Line: $(\pi_T = 0.9, C_{fn} = 1, C_{fp} = 10)$ working point.

From the Bayes Error plots it's clear that the fused models perform very well even on the evaluation set. We also report actDCF and minDCF of the models:

	actDCF			minDCF		
Model	$\pi = 0.5$	$\pi = 0.1$	$\pi = 0.9$	$\pi = 0.5$	$\pi = 0.1$	$\pi = 0.9$
QLR + GMM	0.248	0.546	0.074	0.217	0.490	0.070
SVM + GMM	0.225	0.502	0.078	0.221	0.487	0.076
QLR + SVM + GMM	0.225	0.530	0.080	0.221	0.491	0.078

As we also expected before, we have lower minDCF values for $\pi = 0.9$.

4.3 Quadratic Logistic Regression

Now, let's make a deeper analysis of the Quadratic Logistic Regression model by reviewing if the choice of the λ hyperparameter ($\lambda = 10^{-3}$) is still the optimal one on the evaluation set. We do this analysis in the light of the best model configuration mentioned above.

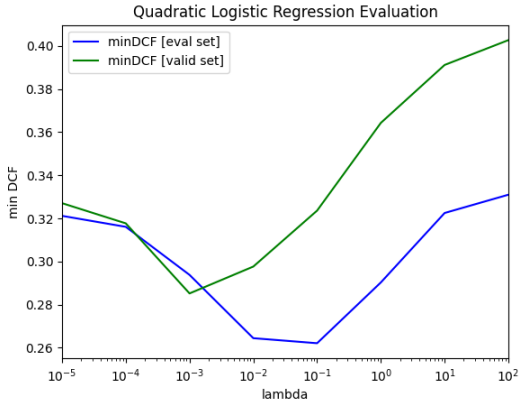


Figure 38: Quadratic LR Evaluation.

Even if in both cases the costs are pretty low, we can see from the plots that the previous choice of λ is suboptimal on the evaluation set and that $\lambda = 10^{-1}$ seems more suited for the evaluation set. We report the table to compare the performance with these λ values. Since the minDCF values are very similar, we included also $\lambda = 10^{-2}$:

	minDCF		
QLR ZNorm no PCA	$\pi = 0.5$	$\pi = 0.1$	$\pi = 0.9$
$\lambda = 10^{-3}$	0.294	0.548	0.114
$\lambda = 10^{-2}$	0.264	0.516	0.105
$\lambda = 10^{-1}$	0.262	0.526	0.103

As we can observe, our previous expectations are confirmed and $\lambda = 10^{-1}$ is more suited for the evaluation set, even though our previously found value on the training set ($\lambda = 10^{-3}$) still provides very good results.

4.4 Polynomial SVM

We make the same reasoning done with QLR also with this model, so we analyze if the choice of c and C hyperparameters is still the best one on the evaluation set. For $c = 1$ we chose the best model configuration ($C = 10^{-2}$, ZNorm PCA), whereas with $c = 0$ we chose the best configuration found on the training phase for this c value (PCA applied without zNorm) to see if on the evaluation set this configuration could lead to better results.

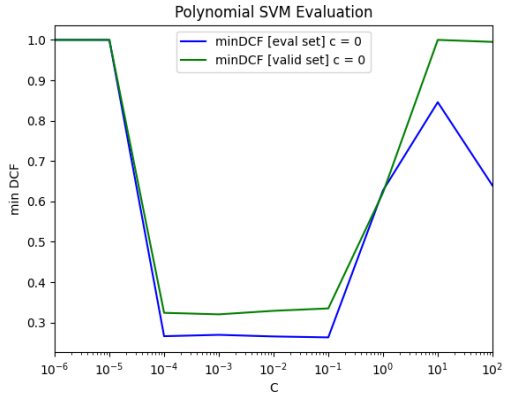


Figure 39: Polynomial SVM with c = 0 Evaluation

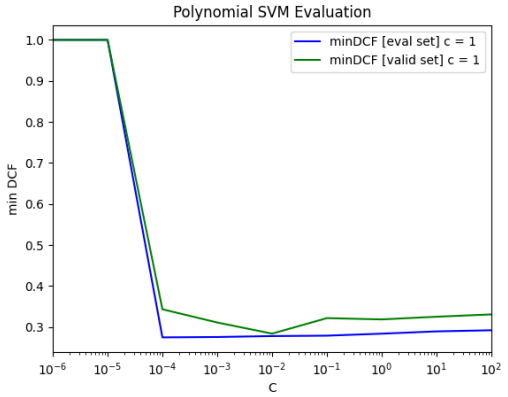


Figure 40: Polynomial SVM with c = 1 Evaluation

As we can see, it seems that on the evaluation set the best configuration is now $c = 0$ with PCA applied and without ZNorm. Here we report the minDCF values:

		minDCF	
$c = 1$ PCA ZNorm	$\pi = 0.5$	$\pi = 0.1$	$\pi = 0.9$
$C = 10^{-3}$	0.276	0.505	0.103
$C = 10^{-2}$	0.278	0.528	0.111
$C = 10^{-1}$	0.279	0.537	0.117

$c = 0$ PCA no ZNorm	$\pi = 0.5$	$\pi = 0.1$	$\pi = 0.9$
$C = 10^{-3}$	0.269	0.553	0.106
$C = 10^{-2}$	0.265	0.575	0.103
$C = 10^{-1}$	0.263	0.560	0.099

Hence, we will choose this new configuration ($c = 0, C = 10^{-1}$, with PCA applied and without ZNorm) on the evaluation set.

4.5 Gaussian Mixture Models

Now we turn our attention to GMM Models. Even if in the training phase we found as the best the Diagonal Covariance GMM with 8 components for the spoofed class and 2 components for the authentic class, we still choose to analyze all the GMM models. This due to the fact that, overall, all GMM models have shown very good performance. In addition, in the score calibration phase the scores showed an excellent behaviour without any need to be calibrated and in the model fusion phase the contribution of GMM models always improved the overall model. Here we have considered only raw results, given the fact that preprocessing techniques proved to be not effective in training phase.

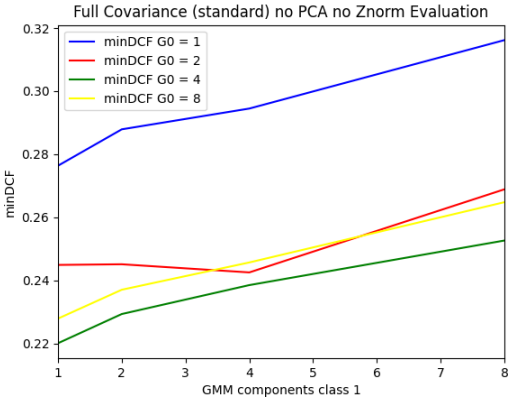


Figure 41: RAW Full Covariance Evaluation

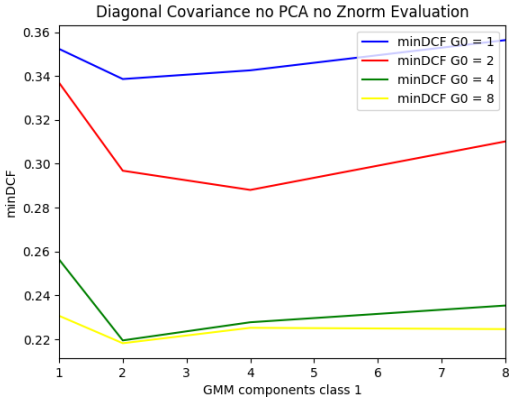


Figure 42: RAW Diagonal Covariance Evaluation

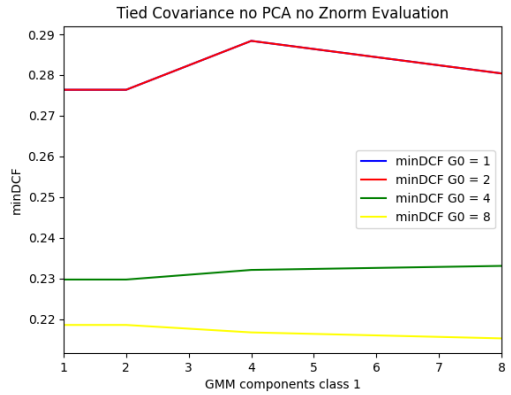


Figure 43: RAW Tied Covariance Evaluation

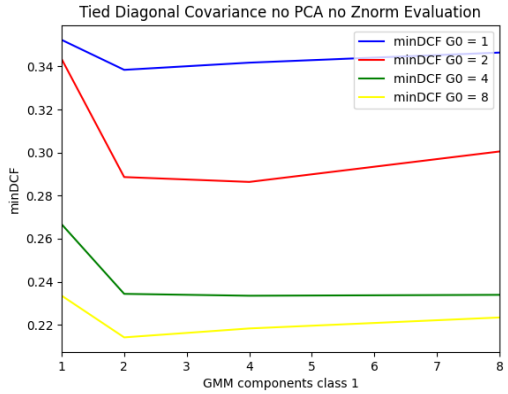


Figure 44: RAW Tied Diagonal Covariance Evaluation

From the plots we can analyze some of the best configurations for these models. We have reported both actDCF and minDCF values for these configurations:

			actDCF			minDCF		
RAW GMM Model	G0	G1	$\pi = 0.5$	$\pi = 0.1$	$\pi = 0.9$	$\pi = 0.5$	$\pi = 0.1$	$\pi = 0.9$
Full Covariance	4	1	0.240	0.483	0.078	0.220	0.473	0.076
Diagonal Covariance	8	2	0.228	0.565	0.072	0.218	0.490	0.070
Tied Covariance	8	8	0.225	0.481	0.078	0.215	0.481	0.074
Tied Diagonal Covariance	8	2	0.230	0.547	0.072	0.214	0.498	0.070

As we expected before, overall all GMM models show a good performance. In this case, the Tied Covariance model with 8 components for both authentic and spoofed class seems to perform slightly better. However, we still tend to prefer the Diagonal Covariance model, since it's a very small difference and also because, given the small correlation of the dataset features, we consider this model more suitable to our case.

5 Conclusions

To summarize, we can see that all the analysis and the consequent choices taken during the training phase have proved fruitful also for the evaluation phase. At this point, the best models to take into consideration are:

	actDCF			minDCF		
Model	$\pi = 0.5$	$\pi = 0.1$	$\pi = 0.9$	$\pi = 0.5$	$\pi = 0.1$	$\pi = 0.9$
GMM Diagonal Covariance	0.228	0.565	0.072	0.218	0.490	0.070
SVM + GMM	0.225	0.502	0.078	0.221	0.487	0.076
QLR + SVM + GMM	0.225	0.530	0.080	0.221	0.491	0.078

Even if fused models perform very well, our choice falls on the simplest one, the GMM Diagonal Covariance model, which still produces good results for the target application and for both unbalanced applications. This because the slight increase of performances obtained by training a fused model does not justify the overhead introduced.