Report Fingerprint Spoofing Detection

Michael Di Giorgio 317819 - Ethan Morleo 317821

November 2023

Contents

1 Introduction

1.1 Abstract

This paper aims to examine the structure of a dataset of embeddings extracted from fingerprint images that are divided into two classes: Authentic (1) and Spoofed (0). What we know a priori is that each sample is described by 10 features and that the classes are slightly unbalanced. In the first part of this report we will initially go to describe the dataset (divided into training and test set) analyzing its various features and their distribution, after that we will train various models discussed during the course, analyze and calibrate them appropriately, test them on a validation set until we draw some conclusions. Each paragraph described below is the result of analyses obtained by applying the notions and labs conducted during the course. They have been implemented in the attached code *"main.py"* which is also divided into subsections parallel to what will be written below.

1.2 About the problem

Embeddings extracted from fingerprint images are a representation of a subspace of the image itself with no physical meaning. Each sample is described by a 10-component vector of real numbers that are obtained by mapping the image into a lower-dimensional space. The images belong to two classes: Authentic which are labeled as 1 and Spoofed which are labeled as 0. What we know about Spoofed is that they are made through artificial intelligence techniques that can be of six types but we do not know which ones. The training set has 2325 samples and the test set 7704 with a large imbalance in favor of Spoofed.

2 Features analysis

The first analysis conducted was on the distributions of the various features. Below are the histograms of the various features for each class.

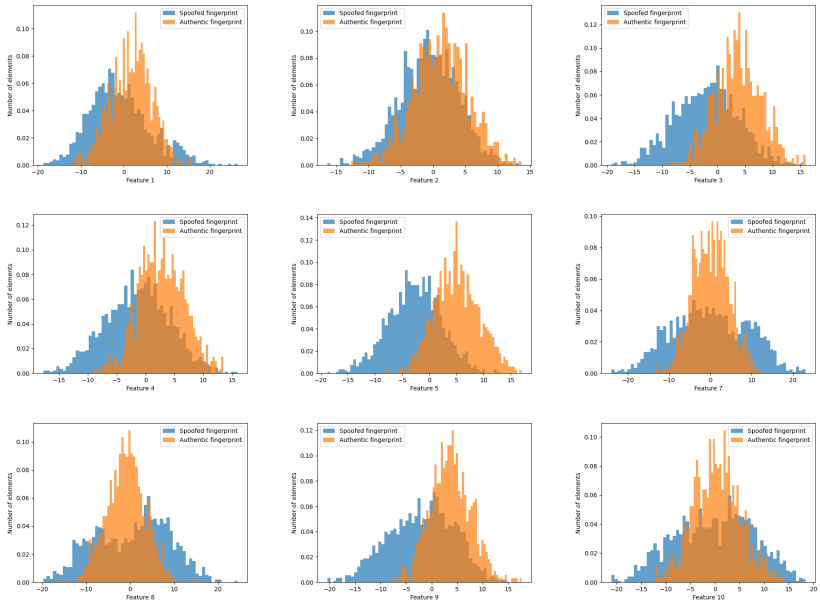

Figure 1: Histograms of the features

We can easily see that particularly the features belonging to the Authentic class can be described by a Gaussian distribution. This, on the other hand, is not always true for the Spoofed class, which seems to be described by other distributions: one might think that this is actually due to the fact that the fingerprint images belonging to this class were obtained by various artificial intelligence techniques that could then represent its subclasses. By printing cross-features plots these considerations become even more valid. Below there are the most significant ones, the rest of them is stored in the folder *"/plots/feature_display/cross_feature"*

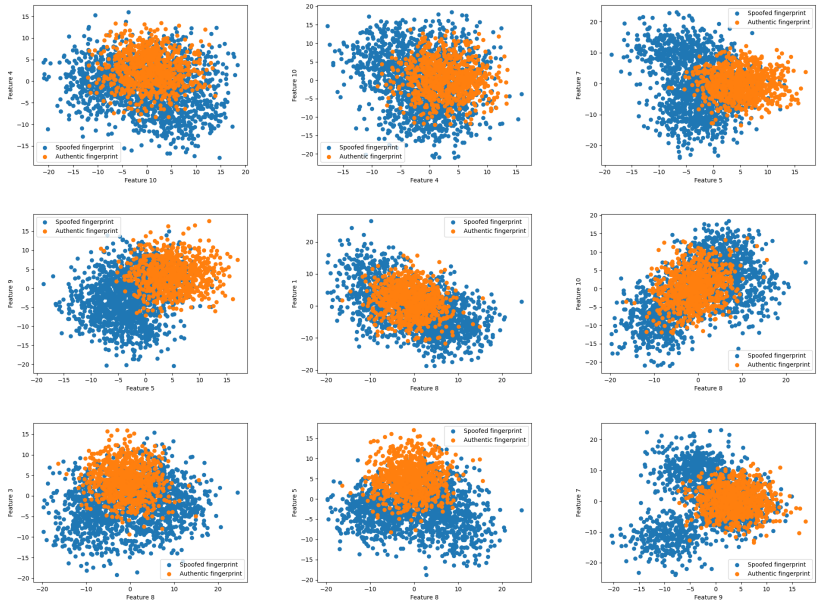

Figure 2: Cross feature plots

2.1 Linear Discriminant Analysis

The first transformation we apply is the LDA. This pre-processing helps us determine whether features tend to be linearly separable along C - 1 direzioni where C is the number of classes (in our case one dimension since the classification problem is binary). These considerations can give us information about who may perform better among Gaussian/linear and non-Gaussian/non-linear models.

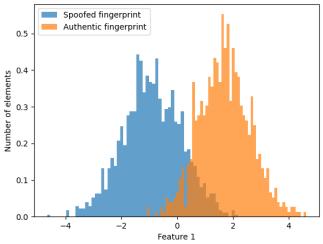

Figure 3: LDA, m = 1

Analyzing the graph, it can be concluded that a linear separation between the two classes might be plausible despite the fact that the possibility of error is not zero.

2.2 Principal Component Analysis

The second transformation we apply is the PCA in order to reduce the dimensional space up to the dimensions who preservs most of the informations. Plotting the graph below we see how PCA6 preservs 89.5% of the information.

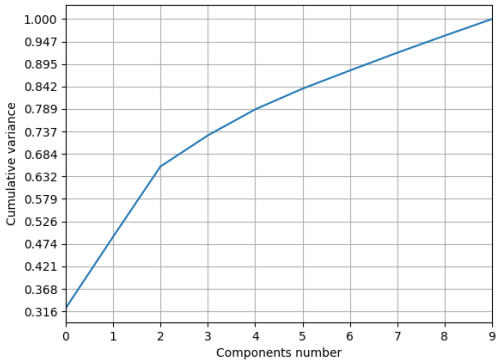

Figure 4: Components number PCA - cumulative variance plot

Again, plotting the features distribution and the cross-features plots as before will lead to the same conclusions we already did.

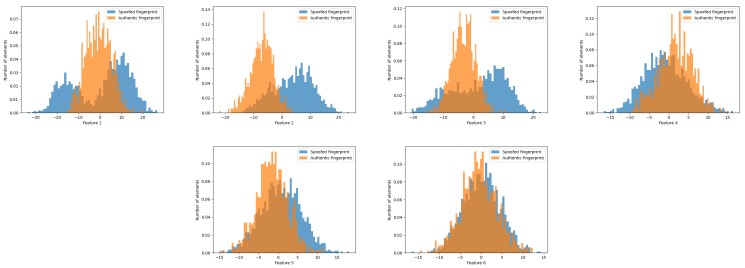

Figure 5: Feature distribitions PCA6

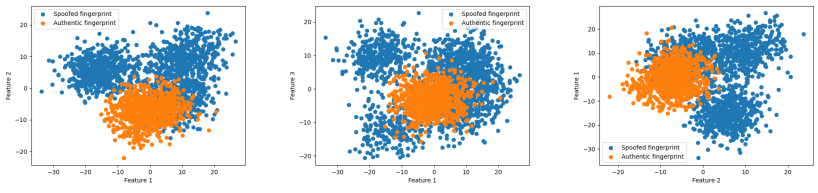

Figure 6: Cross feature plots PCA6

2.3 Pearson Correlation Plots

Pearson Correlation Plots have been used to determine how much the features are correlated with each other:

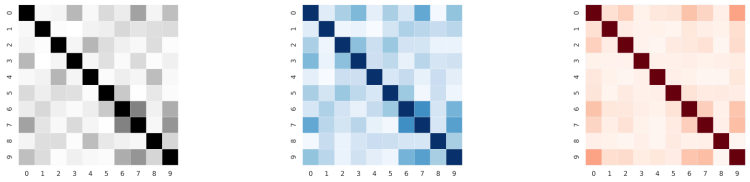

Figure 7: Pearson correlation plots of Whole-Spoofed-Authentic datasets

Analyzing the various heat maps we can see that the features are very poorly correlated with each other although this is not true for all of them (i.e., 6-7 of the Spoofed class) again due to the fact that the Spoofed class is itself composed of multiple clusters.

3 Classification and Validation

In this section, various classifiers trained according to various models explored during the course and subsequently validated will be analyzed. The expectations formulated through the analyses in the previous section will then be compared with the results actually obtained. Specifically, the models used have been:

1. Gaussian Classifiers:
    - Multivariate Gaussian Classifier (Full Covariance Matrix)
    - Multivariate Gaussian Classifier (Naive Bayes)
    - Multivariate Gaussian Classifier (Tied Covariance Matrix)
    - Multivariate Gaussian Classifier (Naive Bayes + Tied Covariance)
2. Logistic Regression:
    - Prior Weighted Logistic Regression
    - Quadratic Logistic Regression
3. Support Vector Machine:
    - Linear SVM
    - SVM with Polynomial Kernel
    - SVM with RBF kernel
4. Gaussian Mixture Models:

- Gaussian Mixture Models (Full Covariance Matrix)
- Gaussian Mixture Models (Naive Bayes)
- Gaussian Mixture Models (Tied Covariance Matrix)
- Gaussian Mixture Models (Naive Bayes + Tied Covariance)

Premises:

1. In order to evaluate the various models, K-Fold cross validation has been adopted, particularly with K=5.
2. In this phase the evaluations are based on minDCF, the actual DCF will be discussed later in this document.
3. The application point, as requested, has been settled to $(\pi,C_{fn},C_{fp})=(0.5,1,10)$, equivalently $\pi_T = \frac{1}{11}$.

3.1 Multivariate Gaussian Models

Multivariate Gaussian Models works under the assumptions that data follows a Guassian distribution:

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

With Σ_c being:

- The full covariance matrix for each class for the standard MVG.
- The diagonal covariance matrix for each class for the Naive Bayes MVG.
- The covariance matrix computed over the whole dataset for the Tied MVG.

Given the analysis conducted in 2 we expect Gaussian models to perform well since that feature distribution exhibit a similar Gassian distribution. Furthermore, based on the correlation analysis conducted trough the Pearson correlation plots we expect also the Naive Bayes to perform sufficiently well although worse than the standard MVG due to some correlations (i.e. 7,8,10). Eventually we expect Tied Gaussian Models to perform worse than the other since the histograms showed that each class is spread differently.

3.1.1 Results

As expected, the MVG model yielded the best results, indicating that quadratic models perform better on our dataset. Looking at the results obtained, we are able to see that PCA helps us greatly improve the performance of the model.

Standard	Naive Bayes	Tied	Tied Naive Bayes
0.331	0.472	0.486	0.551
0.336	0.360	0.483	0.549
0.341	0.361	0.484	0.541
0.333	0.360	0.485	0.544
0.330	0.369	0.492	0.543

Table 1: minDCF of Gaussian Models

However, not having substantial differences between the various minDCFs, we opted to verify with other models which could be the candidate PCA for the analysis.

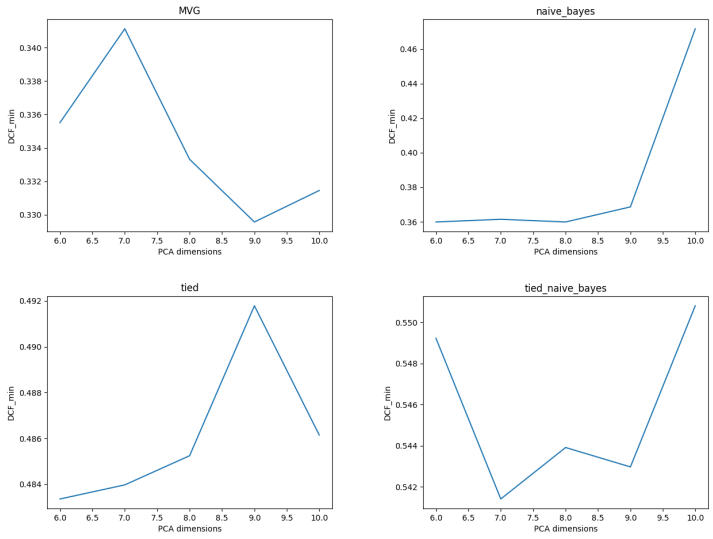

Figure 8: How PCA affects minDCF

3.2 Logistic Regression

Since classes are unbalanced has been applied a prior-weighted regularized version of the objective function of the Logistic Regression problem:

$$J(\omega, b) = \frac{\lambda}{2}||\omega|| + \frac{\pi_T}{n_T} \sum_{i=1|c_i=1}^{n_T} \log(1 + e^{-z_i s_i}) + \frac{1 - \pi_T}{n_F} \sum_{i=1|c_i=0}^{n_T} log(1 + e^{-z_i s_i})$$

with (ω, b) are the model parameters, $\frac{\lambda}{2}||w||^2$ a regularization term that helps obtaining a ω with lower norm and λ a hyper-parameter called regularization coefficient for which:

- $\lambda >> 0$: poor separation of classes but small norm.
- $\lambda \sim 0$: good separation of classes but poor generalization of unseen data.

In order to choose a good value of the hyper-parameter have been plotted various minDCF related to λ. Various values of π_T have been used, applying PCA for several m values and Z-norm.

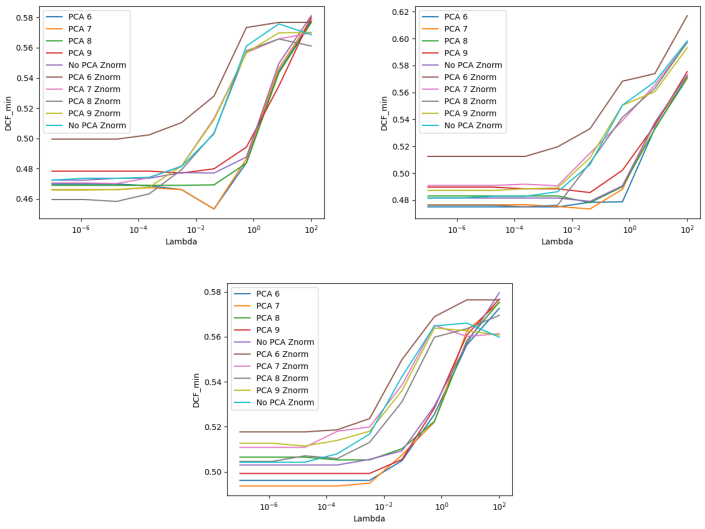

Figure 9: Logistic Regression with $\pi_T = 0.1, 0.5, 0.9$

3.2.1 Results

We see how due to the unbalance between classes $\pi_T = 0.1$ gives the best minDCF. Furthermore, the best values for λ are between 0.1 and 0.2 as we expected. PCA helps to reduce the minDCF meanwhile Z-norm is not useful.

λ	No Z-Norm	Z-Norm
10^{-5}	0.474	**0.472**
10^{-4}	**0.474**	**0.474**
10^{-3}	**0.474**	**0.474**
10^{-2}	0.481	**0.477**
10^{-1}	0.503	**0.487**
1	0.561	**0.550**
10^{1}	**0.574**	0.581

Table 2: minDCF of LR models with $\pi_T = 1$, PCA (m = 6)

3.3 Quadratic Logistic Regression

As we saw in Gaussian models, analysis quadratic models seems to perform better than others so we can anticipate that Quadratic Logistic Regression will give better results. Quadratic logistic regression is a type of regression model that allows us to estimate the probabilities of belonging to a binary class as a function of continuous or categorical explanatory variables, including quadratic terms of continuous variables. This allows to capture any non-linear effects of explanatory variables on the response variable. Another aspect to consider in quadratic logistic regression is the transformation of explanatory variables. We can use the function below:

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

to create an expanded feature space that includes the outer product of the x vector with itself and the original x vector. In this way, we can train the logistic regression model using the $\phi(x)$ feature vectors instead of x. This allows us to calculate linear separation rules for $\phi(x)$, which corresponds to estimating quadratic separation surfaces in the original space.

3.3.1 Results

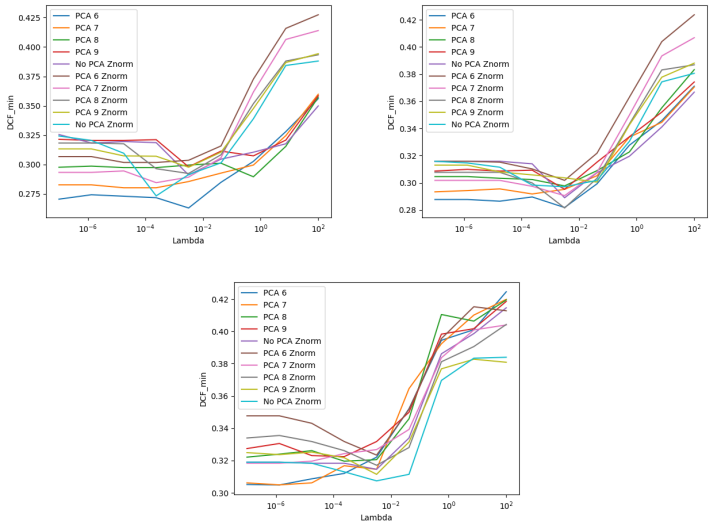

Figure 10: minDCF of Quadratic LR models with $\pi_T = 0.1, 0.5, 0.9$

Again, the best solution is given by $\pi_T = 0.1$, PCA (m=6) and without Z-Norm. As we expected, Quadratic Logistic Regression model performs better than the LR model.

λ	No Z-Norm	Z-Norm
10^{-5}	0.273	**0.265**
10^{-4}	0.272	**0.264**
10^{-3}	**0.272**	0.281
10^{-2}	**0.263**	0.322
10^{-1}	**0.286**	0.346
1	**0.303**	0.346
10^{1}	**0.328**	0.346
10^{2}	0.356	**0.346**

Table 3: minDCF of Quadratic LR models with $\pi_T = 0.1$, PCA (m = 6)

3.4 Support Vector Machine

he problem of minimizing risk in Logistic Regression can be extended to a more general problem that allows for the separation of samples up to a certain margin. This approach is known as Support Vector Machine. To solve the SVM problem, we can use the dual formulation, which is easier to optimize because its complexity depends only on the number of samples. This also allows

us to compute non-linear hyperplanes without having to explicitly expand the features:

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

with

$$0 \leq \alpha_i \leq C, \forall i \in [1, ..., n]$$

and

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

In this section will be analyzed the Linear SVM model which can be obtained by solving the primal problem:

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

where

$$\hat{x_i} = \begin{bmatrix} x_i \\ K \end{bmatrix} \tag{1}$$

$$\hat{\omega} = \begin{bmatrix} \omega \\ b \end{bmatrix} \tag{2}$$

3.4.1 Results

Since we do not expect the linear SVM to give any benefits, we simply tested it with various C and only K = 1.

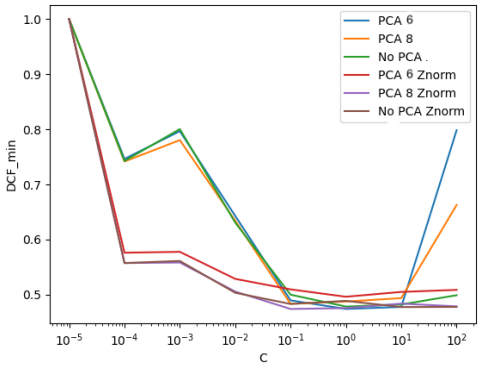

Figure 11: minDCF using Linear SVM

As shown in the table, we have the best value for the PCA (m = 8) case, with $C = 10^{-1}$. In this model, as shown in the table below Z-norm helps getting lower minDCF values.

C	PCA 6	PCA 8	No PCA
10^{-5}	1.000	1.000	1.000
10^{-4}	0.746	**0.741**	0.796
10^{-3}	0.796	**0.780**	0.800
10^{-2}	0.642	0.634	**0.630**
10^{-1}	0.489	**0.483**	0.500
1	**0.474**	0.487	0.478
10^{1}	**0.477**	0.493	0.482
10^{2}	0.798	0.662	**0.498**

Table 4: minDCF of LinearSVM models without Z-Norm

C	PCA 6	PCA 8	No PCA
10^{-5}	1.000	1.000	1.000
10^{-4}	0.576	**0.557**	**0.557**
10^{-3}	0.577	**0.558**	0.561
10^{-2}	0.528	0.505	**0.503**
10^{-1}	0.509	**0.473**	0.483
1	0.504	0.483	**0.477**
10^{1}	**0.477**	0.493	0.482
10^{2}	0.508	0.478	**0.477**

Table 5: minDCF of LinearSVM models with Z-Norm

3.5 Support Vector Machine - Polynomial Kernel

The dual SVM formulation depends on the samples through dot products:

$$H_{ij} = z_i z_j x_i^T x_j$$

This property allows us to calculate scores without having to explicitly expand the features. All we need is the ability to compute dot products between training and test samples. If we have a function that can efficiently calculate dot products in the expanded space, then we can use a kernel function, k, for both training and scoring.

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

This enables us to find a linear separation surface in the expanded space, which corresponds to a non-linear separating surface in the original feature space. In this section will be analyzed the application of SVM with the Polynomial Kernel function:

$$k(x_1, x_2) = (x_i^T x_j + c)^d$$

3.5.1 Results

With d = 2 we obtained the results shown below:

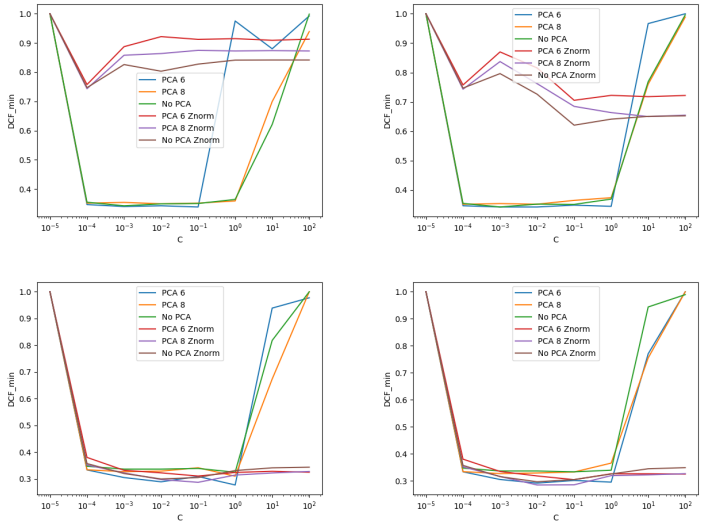

Figure 12: minDCF plots for several C values, SVM with Polynomial Kernel Function

In the tables below is reported the best values obtained. Overall the minimum is for PCA (m=8) and Z-Norm applied ofr hyper-parameter values (c, K) = (1, 1).

C	PCA (m=6)	PCA (m=8)	No PCA
10^{-5}	1.000	1.000	1.000
10^{-4}	0.334	0.334	0.347
10^{-3}	0.305	0.327	0.336
10^{-2}	**0.289**	0.329	0.336
10^{-1}	0.306	0.341	0.339
1	0.675	0.310	0.325
10^{1}	0.938	0.673	0.818
10^{2}	1.000	1.000	1.000

Table 6: minDCF of SVM with Polynomial Kernel, No Z-Norm, K=0, c=1

C	PCA (m=6)	PCA (m=8)	No PCA
10^{-5}	1.000	1.000	1.000
10^{-4}	0.380	0.352	0.357
10^{-3}	0.332	0.323	0.320
10^{-2}	0.323	0.298	0.300
10^{-1}	0.311	**0.287**	0.305
1	0.324	0.314	0.331
10^{1}	0.329	0.322	0.341
10^{2}	0.325	0.328	0.344

Table 7: minDCF of SVM with Polynomial Kernel, Z-Norm, K=0, c=1

C	PCA (m=6)	PCA (m=8)	No PCA
10^{-5}	1.000	1.000	1.000
10^{-4}	0.334	0.334	0.347
10^{-3}	0.305	0.327	0.337
10^{-2}	**0.292**	0.329	0.337
10^{-1}	0.302	0.333	0.334
1	0.295	0.366	0.340
10^{1}	0.770	0.755	0.944
10^{2}	1.000	1.000	0.999

Table 8: minDCF of SVM with Polynomial Kernel, No Z-Norm, K=1, c=1

C	PCA (m=6)	PCA (m=8)	No PCA
10^{-5}	1.000	1.000	1.000
10^{-4}	0.381	0.352	0.357
10^{-3}	0.335	0.316	0.316
10^{-2}	0.318	**0.285**	0.297
10^{-1}	0.304	**0.285**	0.305
1	0.326	0.319	0.325
10^{1}	0.326	0.322	0.345
10^{2}	0.325	0.327	0.349

Table 9: minDCF of SVM with Polynomial Kernel, Z-Norm, K=1, c=1

3.6 Support Vector Machine - Radial Basis Kernel

In this case the kernel function is the Radial Basis Kernel Function (RBF):

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

3.6.1 Results

The analysis has been made for varius values of $\gamma = 0.1, 0.01, 0.001$ and C.

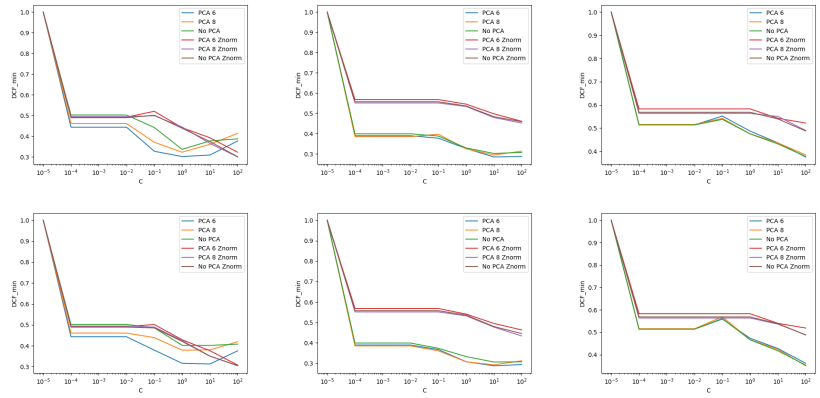

Figure 13: minDCF plots, SVM with RBF Kernel Function

As shown in the table below, the best configuration we had is for (γ, K) = (0.001, 0) and (γ, K) = (0.001, 1). In this case Z-Norm does not help.

γ / C	10^{-4}	10^{-3}	10^{-2}	10^{-1}	1	10^{1}
10^{-4}	0.514	0.514	0.514	0.552	0.487	0.436
10^{-3}	0.390	0.390	0.390	0.378	0.326	**0.285**
10^{-2}	0.443	0.514	0.443	0.327	0.301	0.309

Table 10: minDCF of SVM with RBF Kernel, PCA (m=6), K=0

γ / C	10^{-4}	10^{-3}	10^{-2}	10^{-1}	1	10^{1}
10^{-4}	0.514	0.514	0.514	0.559	0.473	0.427
10^{-3}	0.390	0.390	0.390	0.367	0.308	**0.287**
10^{-2}	0.443	0.443	0.443	0.379	0.316	0.313

Table 11: minDCF of SVM with RBF Kernel, PCA (m=6), K=1

3.7 Gaussian Mixture Models

GMM assumes that it is more con- venient to represent data with Gaussian distributions composed of multiple components (or clusters). Results on the first Gaussian models suggested discrete performance for the MVG and Naive Bayes models. In particular, the latter performed slightly worse due to the slight correlation between some classes. However, since this was not so prevalent, the performance between MVG and NB was almost comparable. So we expect to get

similar results if not better than those just described for Gaussian models. We approached the problem by trying different combinations of models and number components for Non-Target class (Spoofed) and Target Class (Authentic). In particular, we used:

- 1,2) components for class 'Authentic'
- (2,4,8) components for class 'Spoofed'

3.7.1 Results

We know that Spoofed samples are distributed into 6 different sub-classes, so we expected more opti- mistic results when representing Spoofed fingerprints with a higher number of components. We then tried to configure our model trying with all the combinations of the Full-covariance model, Diagonal- covariance model and Tied Full-covariance model for both Target-class and Non-Target class. Because not so useful during the previous analysis, we did not apply the Z-norm for the GMM models. Some of the most significant plots below:

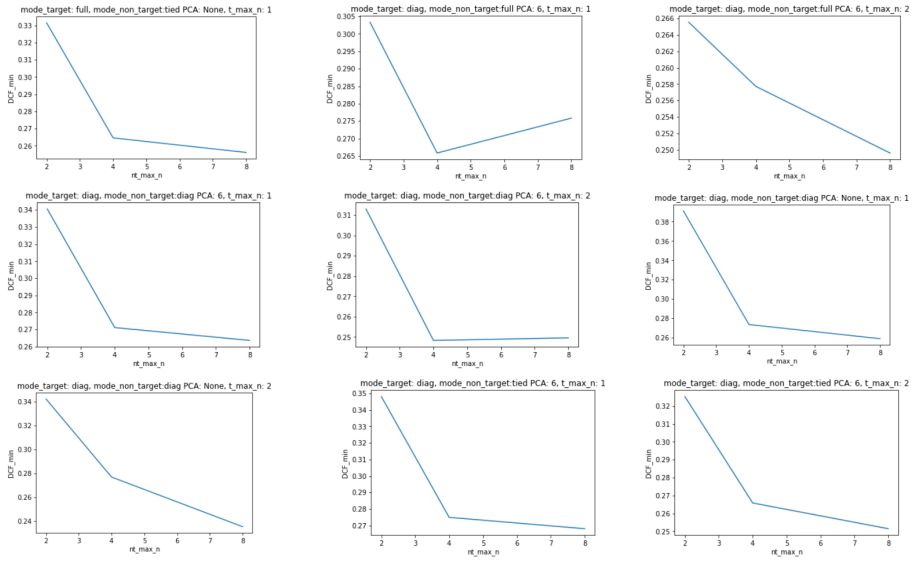

Figure 14: Most relevants minDCF with GMM models plots

Among some of them, you can find some of the lowest DCF values. It is interesting to note that the best results are obtained by applying 8 components to the non-target class, which confirms what was previously assumed about the Spoofed Fingerprint distribution over several sub-classes. In addition, performance is better for the number of components equal to 2 for the target class. A value however low enough to confirm the analysis at the beginning of the report

about the Gaussian distribution of Authentic samples. The same can be said when you notice that the best performance is obtained for the configuration with Diag Cov for both classes. Only the absence of PCA for the best configuration deviates from what has been stated so far. The best results we obtained are reported below, using PCA (m=6) and No PCA.

Number of components (T/NT)	Diag-Full	Diag-Diag	Diag-Tied
(2,1)	**0.303**	0.340	0.348
(4,1)	**0.266**	0.271	0.275
(8,1)	0.276	**0.264**	0.268
(2,2)	**0.265**	0.313	0.325
(4,2)	0.258	**0.248**	0.266
(8,2)	0.250	**0.249**	0.251

Table 12: minDCF with Target mode = DiagCov, PCA (m=6)

Number of components (T/NT)	Diag-Full	Diag-Diag	Diag-Tied
(2,1)	**0.303**	0.391	0.353
(4,1)	**0.284**	0.273	0.290
(8,1)	0.296	**0.259**	0.278
(2,2)	**0.270**	0.342	0.321
(4,2)	**0.248**	0.276	0.268
(8,2)	0.252	**0.235**	0.253

Table 13: minDCF with Target mode = DiagCov, No PCA

The best configuration with Diagonal Covariance applied to both Target class and Non- Target class and 2 components for the Target class and 8 components to represent the Non-Target class, without PCA. This is an absolutely consistent result with what we have seen so far, considering the level of correlation between the features, the Gaussian distribution of the Target class and the number of sub-classes (remembering it is 6, using 8 components for Non-Target class seems quite reasonable) in which the samples of the non-target class are divided. We note that with this latest model, we were able to find the minimum DCF among all models.

3.8 Conlusions on validation step

In conclusion, we consider the previously chosen GMM model as the best solution for our problem:

Model	params	PCA	minDCF
SVM with polynomial kernel function	K=1, c=1, Z-Norm, $C = 10^{-1}$	8	285
SVM with RBF kernel function	$\gamma = 0.001$, K=0, C=10	6	285
Quadratic Logistic Regression	$\lambda = 0.01$, $\pi_T = 0.1$	6	0.263
Gaussian Mixture Models	Components (T/NT) = (2,8) Modes (T/NT) = Diag/Diag	None	**0.235**

Table 14: Best validation models

4 Calibration

To evaluate the different models, so far we have used only the minimum cost of detection (minDCF), which however depends on a threshold. To understand if the threshold is the theoretical one, we will use a metric called actual DCF (actDCF). The method that we will adopt is based on Logistic Regression, which works like a relation of verisimilitude to posterior, so we can obtain the calibrated score by simply subtracting the theoretical threshold. To estimate the parameters of the calibration function, we will use a K-Fold approach, since the number of samples we have is limited. We take just the best 3 models to calibrate.

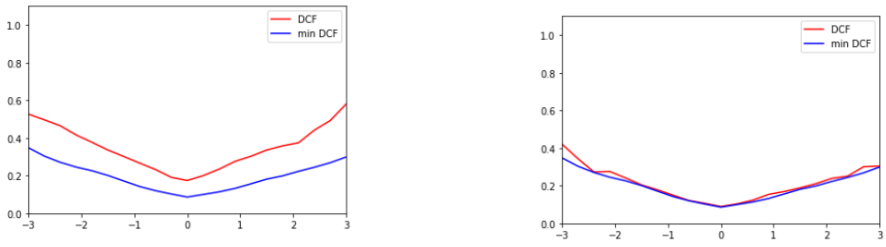

Figure 15: Quadratic Logistic Regression before and after calibration

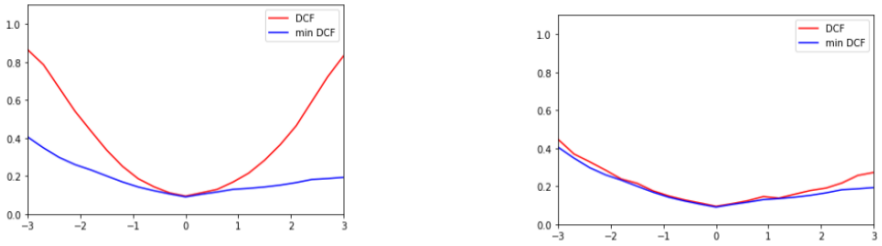

Figure 16: SVM with RBF Kernel Funciton before and after calibration

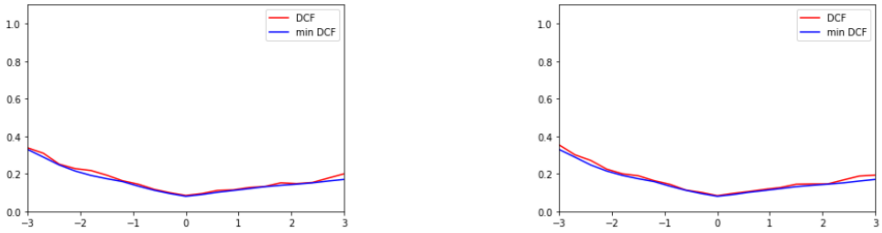

Figure 17: GMM before and after calibration

Quadratic Logistic Regression and SVM shows some imporvements after calibration meanwhile GMM stands pretty much the same. These Bayes Error plots we can also have an idea about how our best models perform in other working points.

5 Evaluation

The next analysis conducted in this paper will be to test the model previosly trained and validated on the test set. Will be carried out just the best three models: the Quadratic Linear Regression, the SVM-RBF and the Gaussian Mixtures Models.

5.1 Quadratic Logistic Regression

During the validation step we had the best value for $\pi_T = 0.1$. Will be considered that value for the plot below:

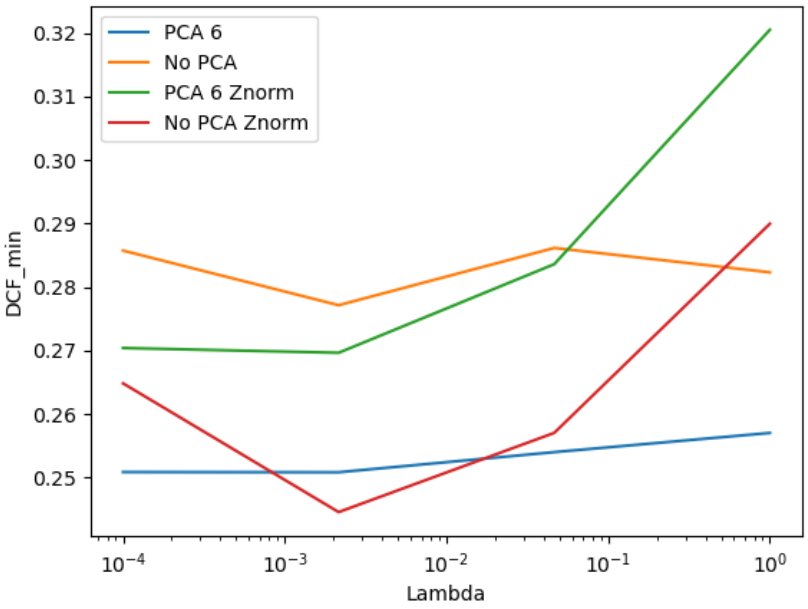

Figure 18: Quadratic Logistic Regression on Test Set

From the plot, it is possible to notice as, although the values of DCF are slightly different, the trends of the models are practically the same if we make a comparison with results obtained during Evaluation and those during Validation. We also applied Z-Norm to the model trained on PCA6 which continues to perform worse than the version without Z-Nor, although we notice a slight improvement if we apply No PCA with Z-Norm, which is a different behaviour from the validation set. The value $\lambda = 10^{-2}$ still remains the best one to use.

λ	PCA (m=6)	No PCA	PCA (m=6) + Z-Norm	No PCA + Z-Norm
10^{-3}	0.251	0.285	0.270	**0.246**
10^{-2}	0.250	0.276	0.270	**0.244**
10^{-1}	**0.254**	0.286	0.283	0.257
1	**0.257**	0.282	0.320	0.290

Table 15: minDCF with Quadratic Logistic Regression on Test Set

5.2 Support Vector Machine - Polynomial Kernel Funciton

In this case we use $c \in [0, 1]$ for the polynomial's kernel setting $K_{svm} = 1$.

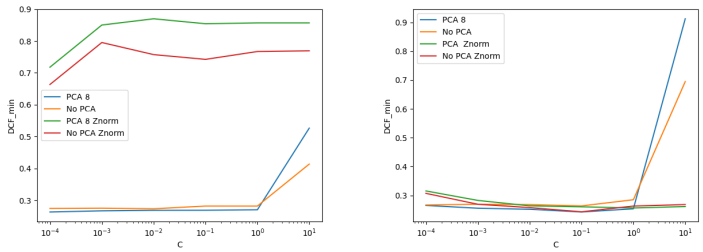

Figure 19: minDCF with SVM - Polynomial Kenrel Function on Test Set with $c \in [0, 1]$

Analyzing this values, the results seems to be consistent with the ones obtained during the previous analysis over the training set, with d = 2, c = 1 and $C = 10^{-1}$ being the best hyper-parameters and PCA 8. However, we made a small error because this time Z-Norm does not improve the performance.

C	no Z-Norm	Z-Norm
10^{-4}	0.270	0.674
10^{-3}	0.270	0.793
10^{-2}	**0.266**	0.714
10^{-1}	0.272	0.603
1	0.322	0.590
10^{1}	0.798	0.593

Table 16: PCA (m=8), K=1, c=0

C	no Z-Norm	Z-Norm
10^{-4}	**0.274**	0.663
10^{-3}	0.276	0.785
10^{-2}	0.278	0.668
10^{-1}	0.281	0.558
1	0.284	0.570
10^{1}	0.554	0.578

Table 17: No PCA , K=1, c=0

C	no Z-Norm	Z-Norm
10^{-4}	0.265	0.315
10^{-3}	0.255	0.282
10^{-2}	0.252	0.263
10^{-1}	**0.242**	0.260
1	0.253	0.256
10^{1}	0.913	0.261

Table 18: PCA (m=8), K=1, c=1

C	no Z-Norm	Z-Norm
10^{-4}	0.267	0.307
10^{-3}	0.269	0.269
10^{-2}	0.268	0.257
10^{-1}	0.263	**0.243**
1	0.285	0.263
10^{1}	0.695	0.268

Table 19: No PCA , K=1, c=1

5.3 Support Vector Machine - Radial Basis Function

In this step we fixed the K value to 0, changing γ values in order to choose the best hyper-parameter.

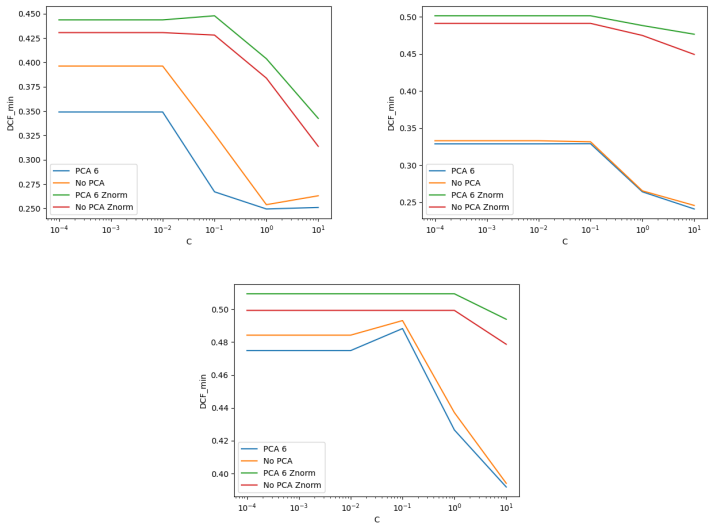

Figure 20: minDCF with SVM - Polynomial Kenrel Function on Test Set with $\gamma \in [0.01, 0.001, 0.0001]$

Even in this case, the results seems to be consistent with the ones obtained during the previous analysis over the training set,with $\gamma = 10^{-3}$ and C = 10 being the best hyperparameters and PCA 6 without Z-normalization being our best choice.

γ / C	10^{-4}	10^{-3}	10^{-2}	10^{-1}	1	10^{1}
10^{-4}	0.475	0.475	0.375	0.490	0.429	0.386
10^{-3}	0.329	0.329	0.329	0.327	0.259	**0.241**
10^{-2}	0.349	0.349	0.349	0.277	0.247	0.252

Table 20: PCA (m=6), No Z-Norm

γ / C	10^{-4}	10^{-3}	10^{-2}	10^{-1}	1	10^{1}
10^{-4}	0.484	0.484	0.484	0.496	0.433	0.391
10^{-3}	0.333	0.333	0.333	0.330	0.264	**0.245**
10^{-2}	0.396	0.396	0.396	0.321	0.255	0.262

Table 21: No PCA, No Z-Norm

γ / C	10^{-4}	10^{-3}	10^{-2}	10^{-1}	1	10^{1}
10^{-4}	0.509	0.509	0.509	0.509	0.509	0.493
10^{-3}	0.502	0.502	0.502	0.502	0.489	0.475
10^{-2}	0.444	0.444	0.444	0.447	0.400	**0.337**

Table 22: PCA (m=6), Z-Norm

γ / C	10^{-4}	10^{-3}	10^{-2}	10^{-1}	1	10^{1}
10^{-4}	0.499	0.499	0.499	0.499	0.499	0.478
10^{-3}	0.491	0.491	0.491	0.491	0.472	0.450
10^{-2}	0.430	0.430	0.430	0.428	0.381	**0.312**

Table 23: No PCA, Z-Norm

5.4 Gaussian Mixture Models

In this section will be evaluated on the test the GMM model previously trained. We will plot some combination of components and modes for the Non Target class in order to compare the results with the validation phase.

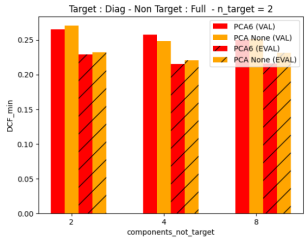

Figure 53: DiagCov-FullCov with 2 Target components

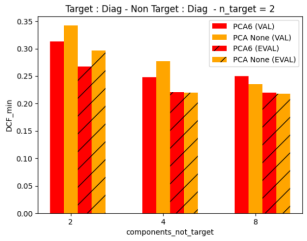

Figure 54: DiagCov-DiagCov with 2 Target components

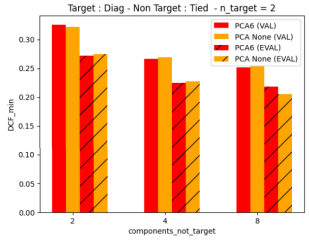

Figure 55: DiagCov-TiedCov with 2 Target components

We can notice some small differences compared to what we saw during the validation phase. In general, however, the trends of the models are very close

to what was previously seen and predicted. The solution chosen, however, is a sub-optimal solution because in this case, the performance of the model that describes the Target class with a Diagonal Covariance matrix and the non Target class with a Tied Covariance matrix performs slightly better.

Modes / Components	(T/NT) = (2,2)	(T/NT) = (2,4)	(T/NT) = (2,8)
Diag-Full	0.232	0.220	**0.215**
Diag-Diag	0.296	**0.219**	0.231
Diag-Tied	0.274	0.227	**0.205**

Table 24: minDCF with GMM on Test Set

6 Conclusion

The result obtained in the evaluation phase confirms what we analyzed in the sections before. However, some behaviours such as the No PCA + Z-Norm for the Quadratic Logistic Regression, the application of Z-Norm for the SVM with polynomial kernel function or the Diag-Tied modes for the GMM are slightly different between the validation and the test set. The best globally solution would be using a Diag-Tied mode for the GMM with an optimal minDCF value of 0.205.