Machine Learning and Pattern Recognition
Fingerprint Spoofing

Federica Amato s310275
Marco Colangelo s309798

1 Introduction

1.1 Abstract

This report aims to examine a dataset that contains various low-level images of real (defined as Authentic) and fake (defined as Spoofed) fingerprints using different Machine Learning models. First, we will explore the structure of the dataset and try to understand how it is distributed. Then, we will evaluate various classifiers and try to find the best system that can accurately classify our samples with the lowest cost.

1.2 First considerations about the problem

The dataset is composed of samples that represent fingerprint images through low-dimensional representations called embeddings. Each fingerprint is represented by a 10-dimensional vector of continuous real numbers, which is obtained by mapping the images to a lower-dimensional space. Real fingerprints are labeled as 1, while spoofed fingerprints are labeled as 0. The individual components of the embeddings do not have any physical interpretation. Spoofed fingerprint samples can belong to one of six different sub-classes, each representing a different spoofing technique, but the specific technique used is not provided. The target application assumes that the prior probabilities for the two classes are equal, but this does not hold for the misclassification costs. The training set contains 2325 samples and the test set contains 7704 samples, with the fake fingerprint class being significantly overrepresented.

1.3 Features analysis

Here is a series of plot about the several features distributions:

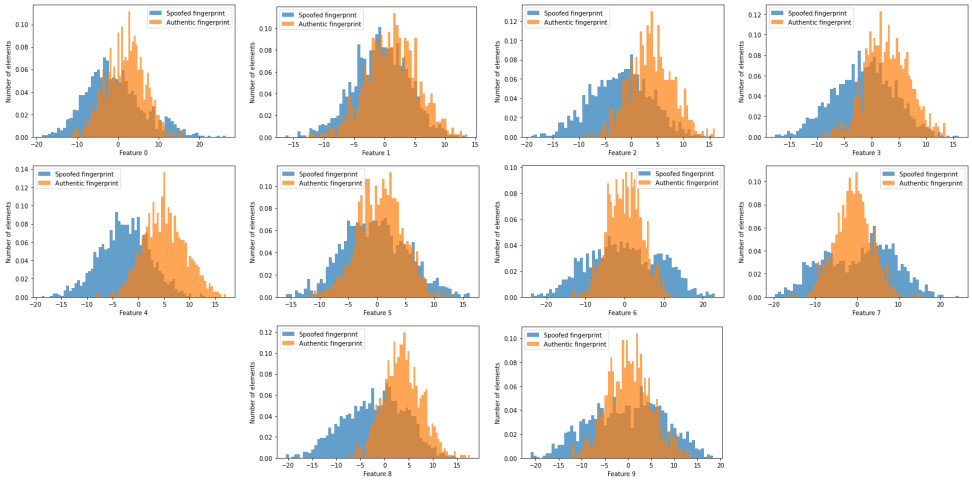


Figure 1: Histograms of the features

You can see how some of these features can easily be described with a Gaussian distribution. Feature 1 (starting to count from 0) is the more evident case. In any case, it is evident that the shape of the distribution to describe the features of the Authentic class is much more similar to a Gaussian form while this is not the case for Spoofed class plots. This can be partly justified by the fact that the samples defined as Spoofed are actually composed of samples obtained with different spoofing techniques. Ergo, the class in question is constructed by putting together elements of different sub-classes. The Feature 4 looks like the one that most easily divides the elements of the two classes. Here are some of the most representative cross freatures plots. It's evident how Authentic class is well described by a Gaussian form and how the position of the Spoofed samples is organized over several clusters (that correspond to several sub-classes):

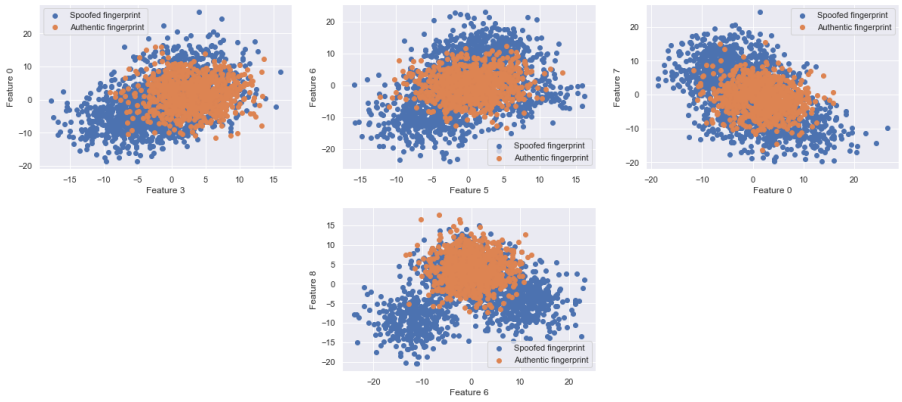


Figure 2: Cross features plot

A transformation that we can use is the Linear Discriminant Analysis (LDA), this preprocessing step allows us to understand if the features are linearly discriminable, so this means that if a linear and/or Gaussian model may perform better than no-gaussian and/or no-linear model. LDA founds C-1 directions over to plot our features where C is the number of classes (in our case we have only 1 dimension because we trait a binary classification problem). looking at the graph we see how it is possible to use linear separation methods for the two classes but the separation would not be without errors.

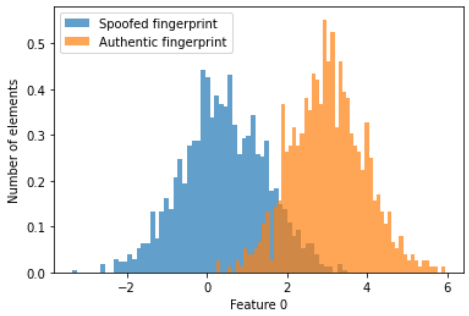


Figure 3: LDA application with m = 1

We focus now on the correlation between the features using Pearson correlations plots.

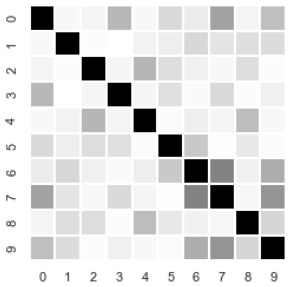


Figure 4: Whole dataset

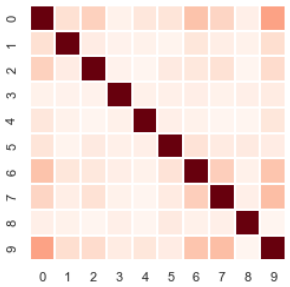


Figure 5: Authentic class

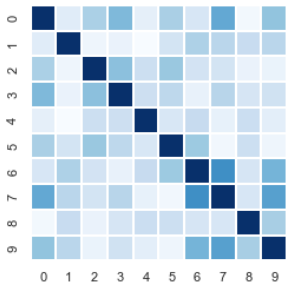


Figure 6: Spoofed class

Looking at the heatmaps, it can be deduced that almost all features are poorly correlated, especially if we look at the Authentic class plot. However, in the plot bound to the whole dataset and in the plot for the Spoofed class we can see some strictly correlated features. This may be reconducted to the several clusters of the Spoofed class. For example, features 6,7 and 9 seem to be quite correlated. this suggests that we may have benefit if we map data from 10-dimensional to 7-dimensional or 6-dimensional space in order to reduce the number of parameters to estimate for a model.

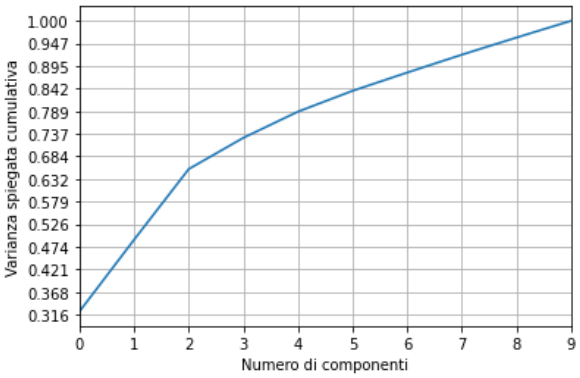


Figure 7: Cumulative variance over PCA dimensions

From the graph we see how with pca 6 we store up to 89.5% of the information, so we will try our models applying PCA to reduce up to 6 dimensions

2 Quadratic Logistic Regression

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

$$\phi(x) = \begin{pmatrix} \text{vec}(xx^T) \\ x \end{pmatrix} \tag{1}$$

function 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.
Until now the best results were obtained with MVG (which is a quadratic model),so we expect that quadratic LR perform better than linear version, because our data seems to be better separated by quadratic rules.

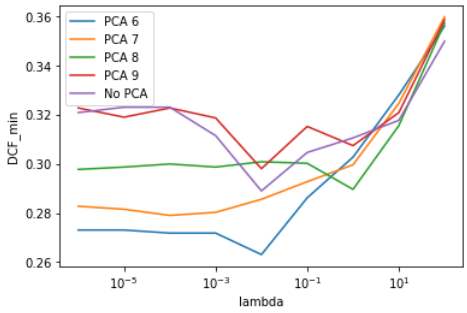


Figure 8: piT = 0.1

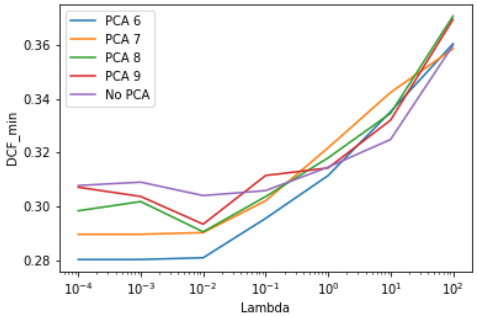


Figure 9: piT = 0.33

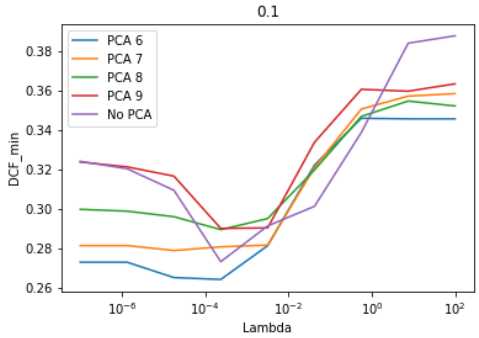


Figure 10: piT = 0.1 + Z-Norm

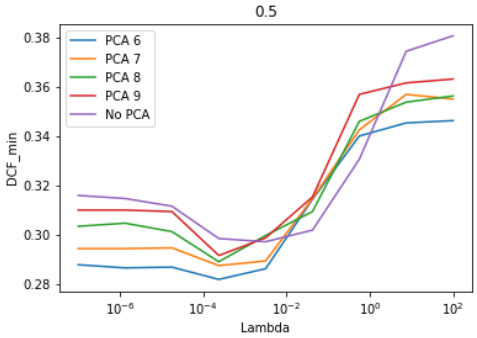


Figure 11: piT = 0.33 + Z-Norm

Due to the lack of performance improvement with piT=0.5 and piT0.9, we tried now with piT=0.33 because similar to the ratio of the number of Authentic samples over the Spoofed ones, but also in this case we did not notice any improvement compared with piT=0.1. We report just the results for piT=0.1 and piT=0.33. Z-Norm data are not put in the tables because we considered them as not relevant in this case. We report just one comparison for piT=0.1 between the PCA=6 configuration and PCA=6+Z-Norm configuration because of them good (and similar) performances. We report the most relevant configurations below:

	piT=0.1	piT=0.33
$\lambda = 10^{-5}$	0.273	**0.280**
$\lambda = 10^{-4}$	0.272	**0.280**
$\lambda = 10^{-3}$	0.272	**0.280**
$\lambda = 10^{-2}$	**0.263**	0.281
$\lambda = 10^{-1}$	0.286	0.295
$\lambda = 1$	0.303	0.311
$\lambda = 10^{-1}$	0.328	0.335
$\lambda = 10^{-2}$	0.356	0.360

Table 1: PCA = 6

	piT=0.1	piT=0.33
$\lambda = 10^{-5}$	0.323	0.308
$\lambda = 10^{-4}$	0.323	0.309
$\lambda = 10^{-3}$	0.311	**0.305**
$\lambda = 10^{-2}$	**0.289**	0.306
$\lambda = 10^{-1}$	0.305	0.314
$\lambda = 1$	0.310	0.325
$\lambda = 10^{-1}$	0.318	0.325
$\lambda = 10^{-2}$	0.350	0.360

Table 2: PCA = None

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

Table 3: PCA=6 with piT=0.1 - Comparison between Z-Norm and No Z-Norm

Applying PCA with m=6 and without Z-Norm is still the best solution. The results are slightly different between Raw features and Z-Scored because we have done a feature expansion operation that computes the dot product inside another embedding space, so the model is sensitive to the transformation of data. We obtain the best model with a value of chosen 10^2 and no Z-Norm applied. We can conclude that classes are better separated with quadratic decision rules. Now we'll test the remaining models.

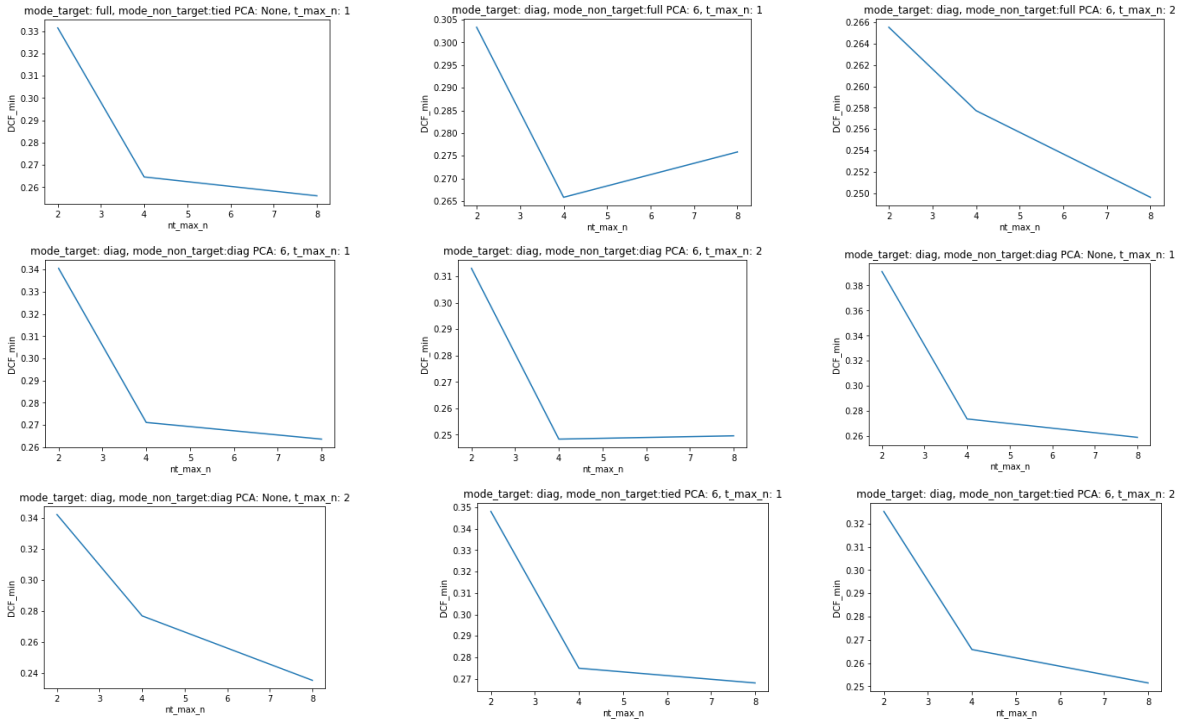
3 Gaussian Mixture Models

The last type of classifier we test is the Gaussian mixture model. This assumes that it is more convenient 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'

We know that Spoofed samples are distributed into 6 different sub-classes, so we expected more optimistic 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.

We report just some of the most significant plots:



So we reported some of the most iconic plots. Among some of them you can find some of the best performance values. It is interesting to note that the best results are obtained by the number of 8 components for the non-target class, which confirms what was previously assumed. 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 value of PCA to None for the best configuration deviates from what has been stated so far.
We decided to report just the results for the best configurations, so the cases with Target class samples represented by a Diagonal Covariance Gaussian

Components/(TargetMode-NonTargetMode)	Diag-Full	Diag-Diag	Diag-TiedFull
(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	**<u>0.248</u>**	0.266
(8,2)	0.250	**0.249**	0.251

Table 4: minDCF for Target mode = DiagCov and PCA6

Components/(TargetMode-NonTargetMode)	Diag-Full	Diag-Diag	Diag-TiedFull
(1,2)	**0.303**	0.391	0.353
(1,4)	**0.284**	0.273	0.290
(1,8)	0.296	**0.259**	0.278
(2,2)	**0.270**	0.342	0.321
(2,4)	**0.248**	0.276	0.268
(2,8)	0.252	**<u>0.235</u>**	0.253

Table 5: minDCF for Target mode = DiagCov and PCANone

We find 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.

4 Final Considerations about Validation

We resume the best three models' information with the following table:

Models	minDCF
GMM Target Mode=DiagCov e Non-Target Mode=DiagCov (2,8), PCA=None	0.235
Quadratic Logistic Rregression lambda = 10^{-2}, PCA = 6, piT = 0.1	0.263
SVM with kernel RBF, KSVM = 0, gamma = 10^{-3}, C = 10, PCA = 6, piT = 0.1	0.285

Table 6: Best Models

5 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. Consider "DCF" string in the legend as a reference to "actDCF".

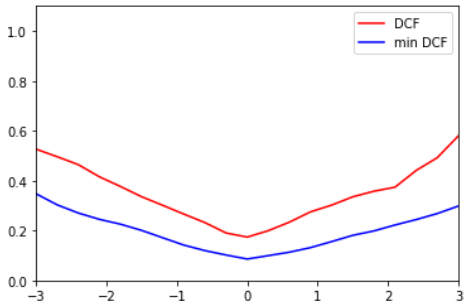


Figure 12: Quad LR model not calibrated

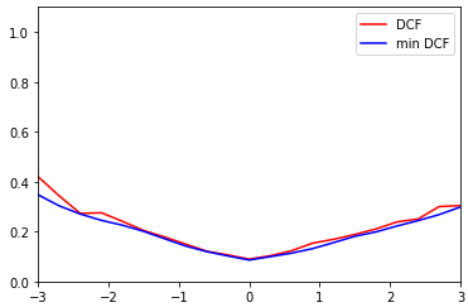


Figure 13: Quad LR model calibrated

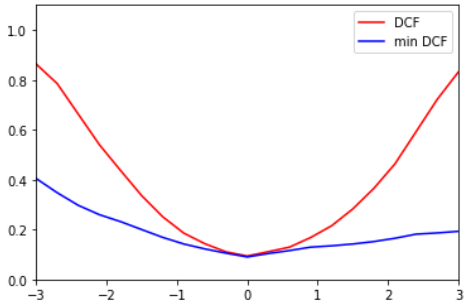


Figure 14: SVM RBF model not calibrated

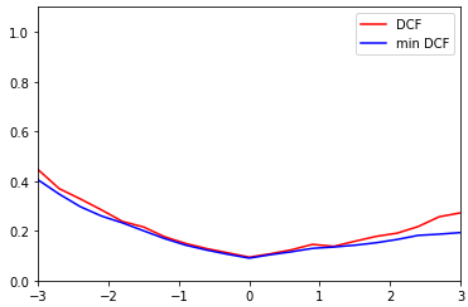


Figure 15: SVM RBF model calibrated

Although the GMM model did not need calibration, we tried to apply it anyway. In fact, the results remained almost unchanged.

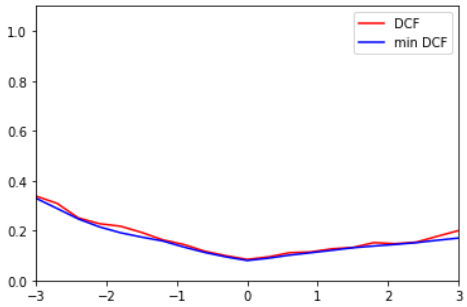


Figure 16: GMM model not calibrated

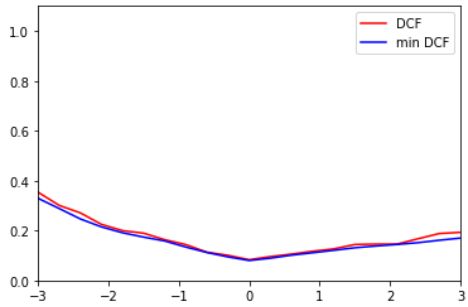


Figure 17: GMM model calibrated

As we can see LR and RBF SVM have the score not calibrated and if we make calibration, we can obtain some improvements. Looking at these Bayes Error plots we can also have an idea about how our best models perform in other working points.

In conclusion, we consider the previously chosen GMM model

6 Evaluation

Now let's move on to the testing phase and check if the choices made previously during the Valuation phase were optimal or not. We also make a comparison by testing models by applying hyperparameters of values discarded previously. We will carry out the tests only for the 3 best models: we will not analyze the linear models of Logistic Regression and SVM nor will we analyze the Gaussian models (as already integrated in GMM)
The training and the test will be carried out not through K-Fold but directly using the Test set and the whole Training set

6.1 Logistic Regression

We first consider the logistic regression models along for the same interval that we use during the training. Our best model, during the training was LR trained with piT = 0.1 so we will consider the plot for this type of model:

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-Norm