# Finger Spoofing Detection classification task

Exam Project for the Machine Learning and Pattern Recognition course

Mario Mastrandrea (s302219)$^1$ and Peter ALHachem (s293885)$^2$
$^{1,2}$Politecnico di Torino

September, 2023

# Contents

# 1 Introduction

This report describes the work carried out by the authors for the exam project of the *Machine Learning and Pattern Recognition* course at Politecnico di Torino, a.y. 2022/2023. The project targets the identification of an optimal Machine Learning (ML) model in the context of a *binary classification task* consisting in the detection of authentic fingerprints. Specifically, we aimed at the development of a classifier capable of detecting whether an image represents an *authentic* fingerprint of an individual or a *spoofed* one, i.e. a fingerprint image that comes as a resultant of applying cloning techniques to the authentic one. The research of such model included the use of diverse pre-processing techniques and Machine Learning classification algorithms, leveraging a dataset of pre-processed fingerprint images, in the form of low-dimensional samples.

In the early stages, an analysis of the *dataset* structure will be illustrated, seeking to comprehend its features' distribution within a well-rounded examination. The next step will contain an analysis of the various *classifiers* aiming at presenting a model able to classify at best our dataset samples with minimum cost. At the end, an *evaluation* of the performances for the best classifiers is presented, showing their results over a test dataset simulating the real population samples.

# 2 The task

The project consists in is a **binary classification task**, i.e. the classification of samples as belonging to one of two possible classes. This is a classical Machine Learning task based on *supervised learning* algorithms, exploiting knowledge derived from previously collected samples whose classes are known. We then employ a dataset of labeled samples to train and evaluate different kind of classification models, adopting different pre-processing strategies to reduce overfitting and improve the performances. We will start the elaborated description of the task with a deepen explanation of the dataset in hand.

## 2.1 The Dataset

Our **dataset** consists of numerical samples representing embeddings of human fingerprints, which are low-dimensional representations of images obtained by means of feature extraction models. For practical purposes, the dataset is represented by synthetic data, which have significantly lower dimensions than in real cases and with no physical interpretation. The embeddings, that represent diverse features of the fingerprint, are **10-dimensional** vectors with **continuous values**. Each data sample is either associated to an *authentic* fingerprint (label 1) or the a *spoofed* fingerprint (label 0). The spoofed fingerprint class samples belong to one of 6 possible sub-classes that are best described by the method used to perform spoofing, however it is important to note that the actual spoofing method (sub-classes) is not available.

The dataset is composed of two different sets: the **training set** and the **test set** (or evaluation set), where the former contains 2325 samples and the latter contains 7704 samples. In the **training set**, with 1525 samples belonging to the *spoofed fingerprint class* and 800 samples present in the *authentic fingerprint class*. On the other hand, the **evaluation set**, the *false class* contains 5304 sample while the *true class* contains 2400 sample. We can deduce that the non-target class is significantly more present than the target one, creating an imbalance in our dataset. The *training set* will be used to train validate the different models using *k-fold cross validation* techniques: the set will be sequentially split into a training set and a validation set, which will be then used to compare the models in hand, for hyperparameter tuning and for score calibration. The test set will remain untouched and will only be used for the evaluation phase.

## 2.2 Target application

The main goal of our project is to optimize the performance of such a classification model for a specific application domain. In fact, despite of the classes distribution in the dataset at our disposal, our **target application** consists of equal *prior probabilities* for the two classes but with different *mis-classification costs*. In particular, the cost of classifying a spoofed fingerprint as authentic is considered 10 times higher the cost of predicting an authentic fingerprint as spoofed. Such a cost imbalance is due to the security vulnerabilities that might be created by those kinds of error, so the risk associated to a malicious person which is erroneously recognized as the authentic one is considered much greater than the one associated to a person not successfully identified.

Since we are dealing with a binary classification problem, these target application information can be summarized by a tuple of just three numerical values:

- the **application prior** $\pi_T$: it represents the prior probability of the *true* class, i.e. the probability of encountering an *authentic* fingerprint, regardless of the nature of the sample (it also contains the information about the prior probability of the *false* class, which is $\pi_F = 1 - \pi_T$);
- the **false negative cost** $C_{fn}$: which is the cost of predicting a *positive* sample as belonging to the *false* class; applied to our target domain, it represents the cost of predicting an *authentic* fingerprint as a *spoofed* one;
- the **false positive cost** $C_{fp}$: likewise, it is the cost of predicting a *negative*

sample as belonging to the *true* class, which means the cost of predicting a *spoofed* fingerprint as an *authentic* one.

These three values take the name of **working point** and they fully represent the information associated to a specific target application. In our case, since we deal with balanced application priors for the two classes, and the false positive cost is 10 times higher than the false negative one, we will consider the following working point: ($\pi_T = 0.5$, $C_{fn} = 1$, $C_{fp} = 10$).

To reduce the complexity of the metric, in binary applications is possible to reduce the working point to a single value, called **effective prior** $\tilde{\pi}$. This corresponds to the value of true application prior $\pi_T$ that, together with unit costs $C_{fn}$ and $C_{fp}$, produces the same results as the original working point. It is computed as:

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

In our case, the effective prior is $\tilde{\pi} = \frac{1}{11} = 0.0\overline{90}$

## 2.3 Method of work

Initially, a **features analysis** will be performed on the dataset in hand. The main purpose behind the examination of the dataset relies on finding distributions and patterns among the classes that can let us identify, with an early assumption, which models can theoretically perform better or worse on the classification task. Furthermore, it allows to explore *dimensionality reduction* techniques in the aim of reducing the complexity of our dataset without losing too much information, and limiting overfitting issues.

The following step consists in analysing different kind of **classification models** using the training dataset and find their optimal configuration. The model families considered are the following: *Multivariate Gaussian (MVG)*, *Logistic Regression (LR)*, *Support Vector Machine (SVM)*, *Gaussian Mixture Model (GMM).* This training phase includes the adoption of cross-validation techniques to try diverse values of hyperparameters (associated with each specific classifier) and different pre-processing strategies (PCA, Z-Norm, feature expansion, etc.) in order to choose the best configuration of our models. After having selected the best parameters for each kind of model, a **comparison** will be held to select the best models among all trained ones.

Next, we will analyze the scores produced by those classifiers, and we will describe possible solutions to perform **score calibration**, in order to obtain models with more robust estimates and therefore applicable to a wider range of target applications.

Lastly, we will **evaluate** our best models over the test set to analyze the correctness of the choices made during the model selection phase and better estimate the costs produced by the use of our candidate models over a real population of samples.

All the steps described so far, as well as the results showed in the following sections have been obtained with the Python scripts attached to this report, containing all the code implementing the mentioned Machine Learning techniques and processing the mentioned fingerprints dataset.

## 3 Features Analysis

In the aim of a thorough examination of the dataset, a feature analysis will be performed with the aim of finding some hidden distributions or patterns that will simplify the search of our optimal model. A feature analysis, is considered important as it will give us an initial confidence of what model candidates might be favorites over others. The dataset considers 10-dimensional embeddings (low-dimensions) as its feature space with each feature holding a continuous value as its representation, forming a **10-dimensional vector** for each sample.

### 3.1 Features distributions

In Figure 1 is shown a matrix of plots containing in its diagonal the **histograms** of the 10 features, and in each cell $(i, j)$ the 2-dimensional **scatter plots** of the $i$-th and the $j$-th features.

An initial observation made on the histograms reveals that some features present an adequate Gaussian distribution within the two classes: **features 3, 4** and mainly **feature 5** on the diagonal show the description observed.

We proceed the observation to showcase that within all the features of the diagonal, it is well highlighted that the *authenticated fingerprint class* portrays a promising description within a **Gaussian distribution** while it cannot be said the same with the *spoofed fingerprint class.* An initial reasoning concerning this behavior would be the association of the spoofed class to 6 diverse sub-classes that, we believe were described around diverse distributions.

Another argument that solidifies this assumption is the observation made on the scatter plots. In fact, we can observe around all the 2-dimensional scatter plots a redundant pattern that showcases a **Gaussian-like distribution** within the *authenticated fingerprint class* while the scarcity of the points on the *spoofed fingerprint class* indicates a difficulty to represent one single area where a similar distribution could be made.

For further confirmation to be made around the preceding assumptions, dimensionality reduction techniques will be presented in the following section.

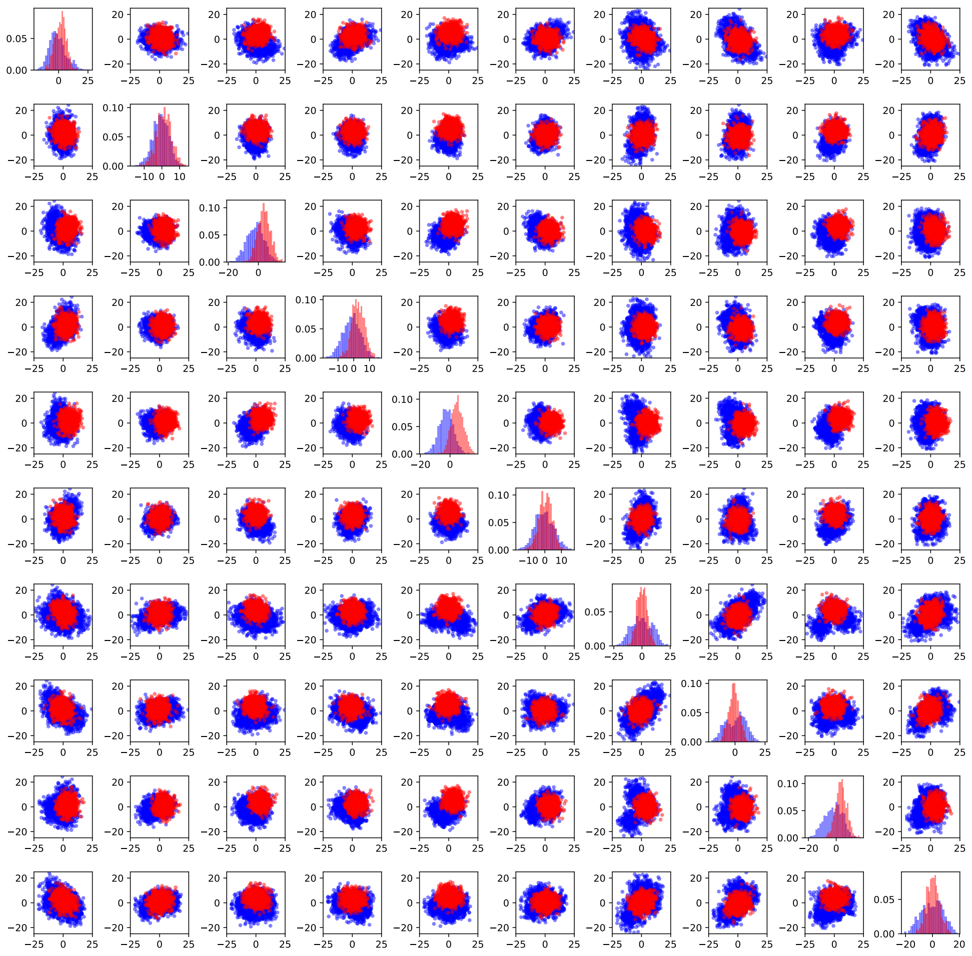
Figure 1: Matrix of histograms and 2D scatter plots of the 10 dataset features

## 3.2 Dimensionality reduction

Reducing the number of dataset dimensions is a fundamental step in every Machine Learning pipeline, in order to reduce the so called *curse of dimensionality.* Having a small number of features, in fact, makes the data less sparse, improving the efficiency of data organization strategies, and also creates advantages for ML algorithms by reducing the number of parameters the models have to estimate and thus limiting the risk of overfitting.

Considering that our dataset is already mapped to a **low-dimensional space**, given the 10-features samples, and considering the good number of samples at our disposal, we expect to already obtain acceptable results even employing the full set of features. However, we will consider in the next sections the use of **PCA** as pre-processing step to analyze the effect of additional *dimensionality reduction* techniques over the results obtained by the different classifiers.

In this section, instead, we will use **PCA** to further investigate the characteristics of our samples, mapping them to a 2-dimensional subspace and visualizing the effect it has on our two classes. This will help us to extract further assumptions on our dataset with the target of solidifying pre-made assumptions made in the previous section. We will also employ **LDA** to map our samples to a 1-dimensional sub-space in order to examine the level of discrimination the two classes have. this will help us predict models that will eventually be optimal considering our dataset. It is important to note that the dimensionality reduction techniques involved are not used as a data pre-processing techniques, at this stage, but rather as indexes to predict working candidate models.

### 3.2.1 Principant Component Analysis

Principal Component Analysis (PCA) is a dimensionality reduction technique that maps $d$-dimensional samples into a lower dimensional space of $m$ features (with $m \ll d$), where $m$ is a hyperparameter of the model which can be selected by means of cross-validation techniques. It is an *unsupervised* method, meaning that it just compresses the dataset size retaining most of its information, regardless of the task for which data will be used. For this reasons, it does not employ the samples' labels to perform the mapping.

An interesting observation of the Principal Component Analysis results is that it finds the directions preserving the *highest variance* of the dataset in the original features space. For this reason, we will first **scale** and **center** our samples before applying PCA (otherwise we would find as most discriminant direction the one connecting the origin to the dataset mean, which is not usually an interesting information).

### 3.2.2 Linear Discriminant Analysis

The **Linear Discriminant Analysis (LDA)** is another data dimensionality procedure, but unlike PCA it is a *supervised* technique employing additional information coming from the samples' labels to optimize the subspace mapping for the specific classification task. It is still based on a hyperparameter $m$, representing the number of dimensions of the subspace identified to reduce the dimensionality of the samples.

A limitation characterizing the LDA is that $m$ must be lower than the number of classes $k$: this means that in *binary classification tasks* (as in our case) we can just map our samples to a **1-dimensional space**. We will then use this method for data visualization to just gather further information about our dataset distribution, but we will not employ LDA as a data pre-processing step for our model analysis.

### 3.2.3 PCA and LDA top features analysis

We pre-processed our training set by means of **PCA** with $\boldsymbol{m} = \mathbf{2}$, to obtain a visual representation of the 2 directions responsible for the highest variance of the dataset. In Figure 2 are shown both the histograms and the scatter plot of the top-2 features.

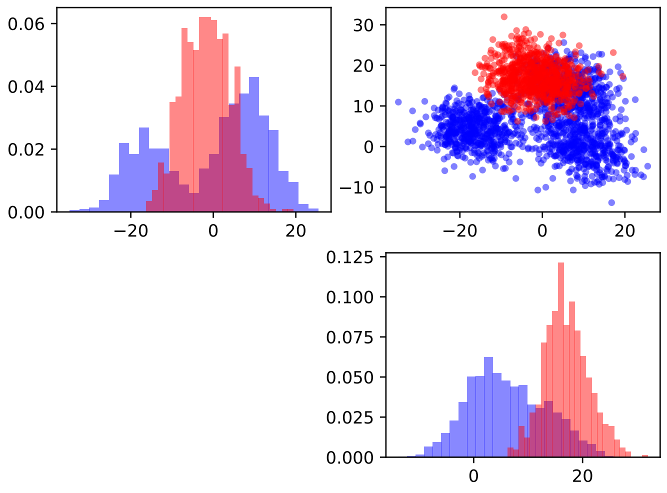

Figure 2: Histograms and scatter plot of PCA top-2 features distributions. **Red** represents the *true* class, i.e. the *authentic fingerprints*, while **blue** represents the *false* class, i.e. the spoofed samples.

As an initial observation is made, the 2-dimensional Principal Component plots confirms our previous assumption that the *authentic fingerprint class* could be represented by a **Gaussian distribution** as the scatter plot intercepts a specific area where the data samples of our target class are represented. However, on the other hand, the *spoofed fingerprint class* is not represented by a Gaussian distribution which means that plain **Gaussian densities** may not be explicitly enough for our non-target class.

Applying instead **LDA** to our dataset, helps us visualize its most discriminant direction and making assumptions about the goodness of linear classification models. In *binary classification tasks* (as in our case), we can just map our samples to a **1-dimensional space** and thus, by mapping the training samples to the most discriminant direction, we obtain the distribution described in Figure 3.

LDA rightfully demonstrates to us that a linear classifier may be able to dis-

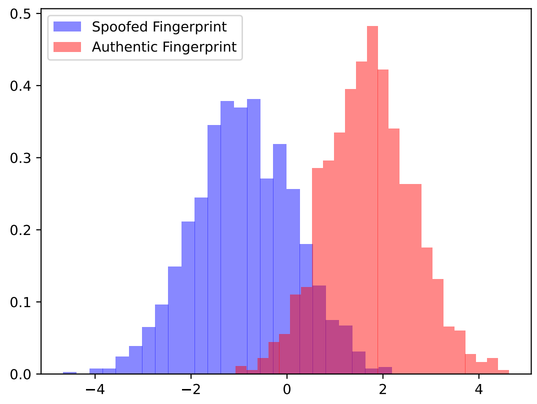

Figure 3: Histogram of LDA most discriminant direction distribution

criminate our classes to a certain extent since there is still an area that might create wrongfully classified samples. However, considering also the analysis done on the previous scatter plot, showing the distribution of the 2 main PCA features, we expect **non-linear classifiers** (like MVG with full covariance, GMMs and SVM) to be significantly better performing than linear ones.

## 3.3 Features correlation

We now shift our attention towards analysing the *Pearson correlation* between the 10 features of our dataset. In Figure 4 are shown the features correlation matrices related both to the entire dataset and to the two separated classes.

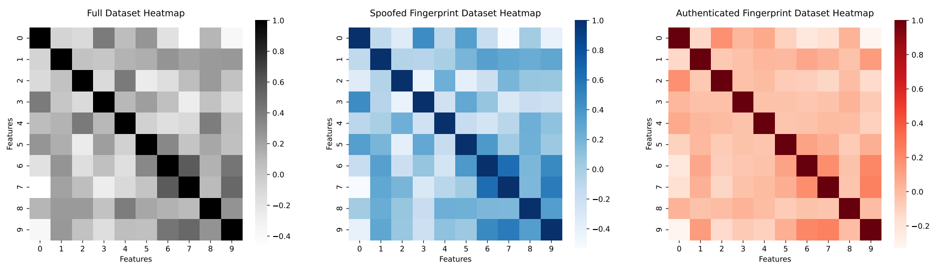

Figure 4: Heatmaps showing *Pearson correlation* among the dataset features

We are able to identify some similar correlations between the three different cases studied. For example, **features 0 and 9** or **features 0 and 7** are always weakly correlated while on the other hand, **features 9 and 7** are always highly correlated. In general, we can state that our dataset present some correlations between its features, even if not necessarily evident.

The correlation, although it follows a generic pattern between the three cases, can be found distinguishable between the full dataset and the specific cases of taking each class alone. It is important to notice from the plots that the correlation of the *authentic fingerprint class* features is weaker than the one interesting the *spoofed fingerprint class*, which coincides with the fact that the non-target class finds its strong correlation from the features being directly impacted by the spoofing method that generated the *spoofed fingerprint class in the first place.* We do expect a PCA dimensionality reduction will not strictly remove important information from our dataset due to the features correlation. However, it should be used carefully due to the weaker correlations present in the target class.

### 3.3.1 Explained dataset variance

To select good candidate values for the PCA data pre-processing step, we can have a look at the Figure 5, showing the **dataset variance** explained as the number of PCA dimensions increases.

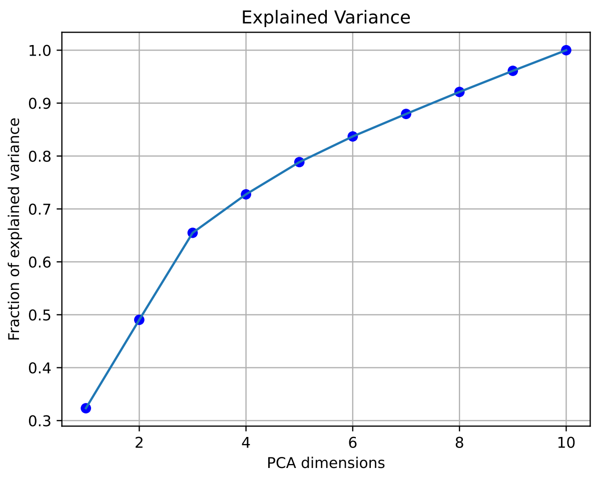

Figure 5: Explained dataset variance vs selected PCA dimensions

We can notice in the graph that almost the 80% of our dataset variance is already explained by just half of the dimensions. We have just 10 features, and all of them provide a good contribute in the overall variance explanation. Anyway, since more than 90% of the variance is already explained by 8 features we will mostly focus on the analysis of the results obtained using **PCA** with $\boldsymbol{m}$=**7,8,9** for our model selection.

# 4 Models analysis

After an elaborate feature analysis where we have explained the diverse behaviors included in our dataset, made assumptions on the optimal classifiers that we esteem adequate for best performances and selected the candidate pre-processing hyper-parameter concerning the fittest dimensionality reduction, we deflect our attention towards proving that our initial assumptions are correct by considering several models that are considered candidates for best classification model.

The models considered in our analysis, also exploring possible variants, are the following ones:

1. Multivariate Gaussian (MVG) classifier:
   - with full covariance
   - with diagonal covariance
   - with tied covariance
2. Logistic Regression (LR):
   - Linear LR
   - Quadratic LR
3. Support Vector Machine (SVM):
   - Linear SVM
   - Polynomial Kernel SVM
   - RBF Kernel SVM
4. Gaussian Mixture Model (GMM):
   - with full covariances
   - with diagonal covariances
   - with tied covariances

Each of these models will be performed over pre-processed data (mainly using **Z-Norm, PCA**) and with further exploration over several hyper-parameter values, by means of *k-fold cross validation.*

With **Z-Score Normalization (Z-Norm)** being an important preprocessing technique performed on the dataset that normalizes the dataset in hand to comply to the form of having a mean equal to 0 while the norm of the standard deviation will be normalized to be equal to 1. Using Z-Norm, we look forward to an easier interpretation of the data used within the models, and a decent effect on the result of some models that are present in our analysis (e.g. LR, SVM).

## 4.1 K-fold Cross Validation

In the model analysis, we employ **k-fold cross-validation** as a crucial step of the model development process. It is a widely adopted technique to test the performance of a classifier adopting different hyperparameters and/or different pre-processing strategies, in order to select the **best model configuration** among all the possible ones.

One simple strategy to perform cross validation consists in splitting the training set into two separate parts: the first one used to train the models, and the second one used to evaluate their metrics, comparing the different configurations. This strategy has the advantage to be very easy to implement and less time-consuming, but it hides the drawback of choosing a 'good' split for the training set, in order to have a big fraction of samples to train the models, but at the same time enough data to obtain good estimates of the evaluation metrics, reflecting as much as possible the real population.

Implementing *k-fold cross-validation*, instead, we divide our training set into *k* equal-sized subsets (or **folds**), which are then iteratively used for the validation. During each iteration, in fact, one fold serves as the *validation* set, while the remaining $k-1$ folds are used for *training* the model. This process is repeated *k* times, with each fold taking a turn as the validation set.

By rotating through all folds, we ensure that every data point in the training dataset is used for validation exactly once. This helps us to gauge the model's performance across various data subsets, providing a more comprehensive evaluation. At the end, after choosing the best configuration of hyperparameters and pre-processing methods, the candidate model is obtained trained it over **all the train samples**, regardless of the folds.

The use of the *k-fold cross-validation* protocol, along with its rigorous training-validation separation within each iteration, ensures that our model is trained on a diverse range of data and is capable of providing reliable results plausible to the ones on unseen samples. For our model analysis we chose to adopt $k = 5$ as number of folds for the cross validation steps.

## 4.2 Evaluation Metrics

The evaluation of classifier performance is a crucial indicator in our study, especially when considering the costs associated with different types of mis-classifications.

We employ **the Detection Cost Function (DCF)** as a comprehensive metric to assess our classifiers' effectiveness over a test dataset. DCF allows us to factor in the costs of false positives and false negatives scores, providing a holistic view of our models' performance. It's important to note that DCF comprises both the False Positive Rate (FPR) and the False Negative Rate (FNR), weighted by misclassification costs. However, merely calculating DCF is far from sufficient.

We focus on the concept of **minimum DCF**, denoted as $DCF_{min}$, as it represents the lower bound for achievable performance by iteratively using all the possible scores of potential thresholds. The *gap* between $DCF_{min}$ and the actual DCF serves as an indicator of **mis-calibration** in our classifiers, where the actual DCF serves as an indicator on the real state of our classifier.

To select the best models, we prioritize those that minimize $DCF_{min}$ since they exhibit the best trade-off between false positives and false negatives and thus providing us with the best models available.

Subsequently, we delve into the crucial task of **score calibration**, aiming to fine-tune the scores produced by our classifiers to align the actual DCF with $DCF_{min}$. This calibration process ensures that our classifiers' predictions are optimized to meet our specific cost considerations and real-world objectives.

It is important to emphasize that **we are considering just a single effective prior** $\tilde{\pi} = \frac{1}{11} = 0.0\overline{90}$ with a single working point provided as $(\tilde{\pi}, 1, 1)$ **for all our following estimations of** $DCF_{min}$. An elaborated explanation of the effective prior is provided within the task section (subsection 2.2).

## 4.3 Multivariate Gaussian Classifiers

The Multivariate Gaussian (MVG) classifier represents a classical example of *generative model* in the context of ML classification tasks.

A **generative model** is a *supervised* probabilistic model which does not directly predict the label of a sample $x$, i.e. it does not model the posterior class probability of the sample $P_{C/X}$. Instead, it tries to separately estimate the density distribution function of each class, and then estimate for a new test sample $x$ its class conditional probabilities $P_{X/C}$ of belonging to each class. Then, leveraging the *Bayes theorem* and application domain information in the form of application prior probabilities $P_C$, it indirectly derives the $P_{C/X}$ and classify a sample choosing the label corresponding to its highest posterior class probability.

The MVG is a model which assumes that samples are distributed according to a **Multivariate Gaussian density**, whose main parameters are the mean $\mu$ and the covariance matrix $\Sigma$.

We can consider the covariance as it is and different for each class, thus training a **MVG with Full Covariance (FC)**, whose separation rules are quadratic curves (so a *non-linear* model). Or else, we can consider two more variants of this classifier:

- we could reduce the effort for estimate those class covariances, assuming the same covariance matrix $\Sigma$ for all the classes, and thus training a **MVG with tied covariance**, which is characterized by *linear* separation rules;
- we can assume features to be uncorrelated (*Naive Bayes assumption*), and training a **Naive Gaussian Classifier** estimating just **diagonal** covariance matrices, again decreasing the number of parameters to estimate and reduce overfitting risk.

Due to the general dataset correlations arising from the previous analysis (subsection 3.3), we expect *Naive Bayes classifiers* to perform poorly with our samples; while the *tied covariance MVG* might obtain better results, due to the similarities between the two classes covariances. But we expect the **MVG with Full Covariance** to be the most promising model, given also the good amount of samples at our disposal.

For the sake of our models analysis we will train the three different kinds of classifiers using a *k-fold cross-validation* protocol (with $k = 5$), analyzing also different values of **PCA** dimensions. In Table 1, Table 2 and Table 3 are respectively shown the results concerning MVG with *Full Covariance*, MVG with *Tied Covariance* and MVG with *Diagonal Covariance*, applying different PCA preprocessing strategies.

| PCA | min DCF |
|---|---|
| - | 0.331 |
| **9** | **0.330** |
| 8 | 0.333 |
| 7 | 0.341 |
| 6 | 0.336 |
| 5 | 0.359 |

Table 1: 5-fold CV results of **Full Covariance MVG** with different PCA dimensions

| PCA | min DCF |
|---|---|
| - | 0.486 |
| 9 | 0.492 |
| 8 | 0.485 |
| 7 | 0.484 |
| **6** | **0.483** |
| 5 | 0.490 |

Table 2: 5-fold CV results of **Tied Covariance MVG** with different PCA dimensions

After a thorough explanation of the model functionalities with a clear display on

| PCA | min DCF |
|---|---|
| - | 0.472 |
| 10 | 0.370 |
| 9 | 0.369 |
| **8** | **0.360** |
| 7 | 0.361 |
| **6** | **0.360** |
| 5 | 0.371 |

Table 3: 5-fold CV results of **Diagonal Covariance MVG** (*Naive Bayes*) with different PCA dimensions

the results in hand, two important observations can be concluded. The first observation confirms that our expectation of a **MVG with full covariance** performs better than its other variants, with its best result of *min DCF = 0.330* using PCA=9. The reason behind this performance relies within the significant correlation present in our dataset and the fact that the non-linearity present within the full covariance model (quadratic model) interprets better the considerations of our *spoofed fingerprint class* and its separability from *authentic fingerprint class.*

On the other hand, it is important to highlight the impact **PCA** has on the results in all 3 variants of our classifier, as we can detect a little improvement in the performance within three variants. However with our model candidate considered, we do realize that **PCA** does not have a crucial difference within the *min DCF* comparison made and thus the information lost is not significantly important in this case.

## 4.4 Logistic Regression models

Logistic Regression (LR) presents a clinical representation of *discriminative models* within the ML realm.

**Discriminative models** are designed to capture the boundary or decision boundary that separates different classes or categories in a dataset. Unlike generative models, which aim to model the joint probability distribution of both the input features and the labels, discriminative models focus solely on modeling the conditional probability of labels given the input data.

**Logistic Regression (LR)** relies primarily on classifying precisely the data samples. It completes this task by paying close attention to how likely something belongs to a certain group based on the data it sees. In doing so, Logistic Regression (LR) becomes skilled at forming clear boundaries between different groups, telling us exactly when something belongs to one group or another.

In the realm of LR, two critical variants stand out that will be considered in our analysis:

- **Linear Logistic Regression (LR):** it refers to a classical binary LR where its main target is to discriminate data samples as belonging to a target class or a non-target class using the posterior probability technique.

  Considering the *objective function* of the **Prior-Weighted LR**, where

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

  is considered as more preferable to work with an unbalance dataset as ours. The main target is the minimization of average cross-entropy of our objective function and at the same time the maximization of the maximum likelihood that will present to us the observed labels.

- **Quadratic Logistic Regression (Q-LR):** it is a specialized Logistic Regression model which includes quadratic terms, allowing it to capture non-linear relationships between predictors and the response variable. In Quadratic Logistic Regression, an important aspect is that we can solve the task leveraging the same objective function of the linear case, by applying a proper transformation over the samples, called *feature expansion.* It is formally on applying a function to each sample mapping them to a higher dimensional space. For Quadratic Logistic Regression we employ the transformation:

$$\phi(x) = \begin{bmatrix} vec(xx^T) \\ x \end{bmatrix} \quad (3)$$

  where $vec(X)$ here denotes a transformation that vertically stacks the columns of its input matrix $X$. This function thus expands the feature space by including the outer product of the $x$ vector with itself, in addition to the original $x$ vector. Consequently, we train the logistic regression model using the feature vectors $\phi(x)$ instead of $x$. This enables us to calculate linear separation boundaries for $\phi(x)$, which corresponds to estimating **quadratic separation surfaces** in the original features space.

Notably, our previous best results were achieved with the Multivariate Gaussian (MVG), inherently quadratic. Therefore, we anticipate that Quadratic Logistic Regression will *outperform* the linear version, as our data appears to exhibit better separation when assessed with non linear decision boundaries.

We first consider the effect of **Z-Norm** on the classical LR model, with different values of the regularization parameter $\lambda$. Logistic Regression models, are in fact, greatly affected by the scaling process performed on the dataset, which might leave us with significantly better results when training the model. $DCF_{min}$ values are shown in Figure 6. And from a small observation on this graph we can see a significant increase in performance iterating over several values of $\lambda$ using Z-Score Normalization (Z-Norm) pre-processing technique. However, diving deep in the results of graph we do not see a great impact performed by using Z-Norm as a $DCF_{min}$ greater than 0.3 is still a very bad index in the model analysis.

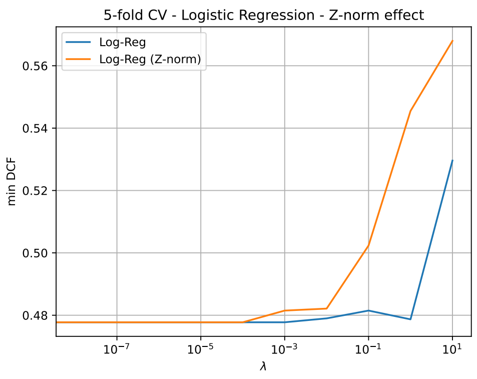

Figure 6: 5-fold CV min DCF of **Logistic Regression** classifier applying (or not) **Z-Norm**

We now shift our attention to the impact PCA and Z-Norm, together, have over classic LR. It is highly crucial to mention that we are following a *greedy* search for the optimal parameters(i.e. we are looking for sub-optimal results changing just one parameter and fixing the others, in turn). Results of LR and Z-Norm, using different PCA strategies, are shown in Figure 7.

An important observation could be gathered that the results using several values of $\lambda$, with the intervention of Z-Norm and PCA does not immensely impact the results performed on the Linear LR as the results are still not exceptionally great. We do believe, at this point, that the bad results displayed have a direct relation with the model being a **Linear LR**.

We try to experiment the training set on a **Quadratic Logistic Regression (Q-LR)** (LR with Quadratic Feature Expansion). The initial experimentation will be made without PCA, by analyzing the effect of the z-norm solemnly. In Figure 8 the $DCF_{min}$ results of Q-LR and the effect of applying Z-Norm.

A full examination on Figure 8 showcases a positive impact of Z-Score Normalization (Z-Norm) on the model considered as the values presented remain within a plausible range. Although Z-Norm worsens the results of $DCF_{min}$ after increasing $\lambda$ the regularization term, the values are still pretty much acceptable with a great

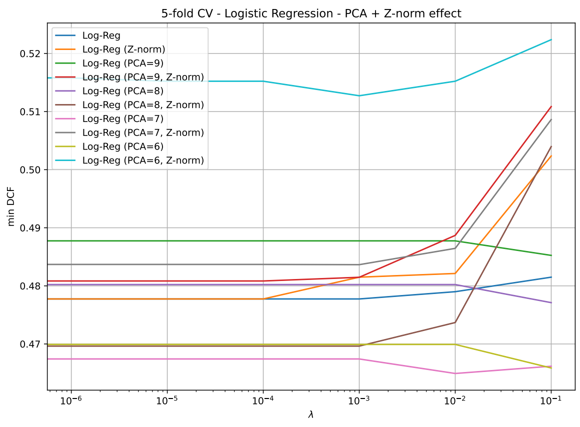

Figure 7: 5-fold CV min DCF of **Logistic Regression** classifier applying (or not) **Z-Norm**, with different values of **PCA**

consideration to a good value with $\lambda = 10^{-2}$.

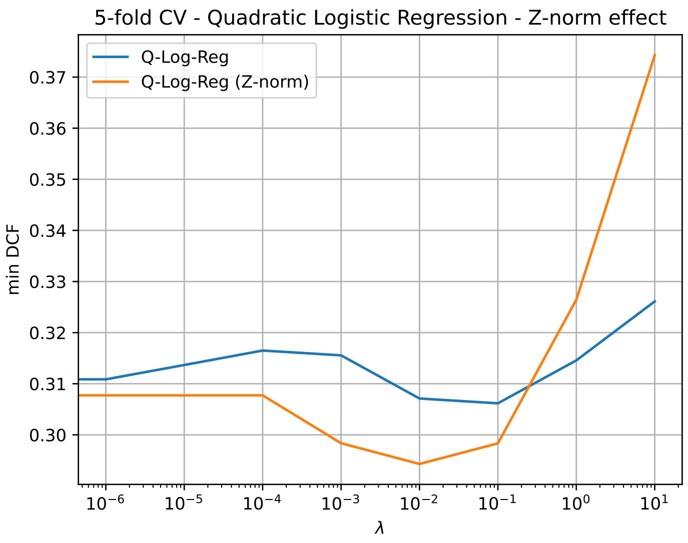

Figure 8: 5-fold CV min DCF of **Quadratic Logistic Regression** classifier applying (or not) **Z-Norm**

Moving forward, we attempt the addition of PCA to the mix as the results are shown in (Figure 9). Our motivation of considering PCA is considered very fruitful as the results keep on improving from considering Z-Norm as the only preprocessing technique within the model and rather having two techniques instead with an honorable mention to the best results of Quadratic Logistic Regression obtained with $\boldsymbol{\lambda = 10^{-4}}$ and **PCA** $=$ **6**, without z-normalization.

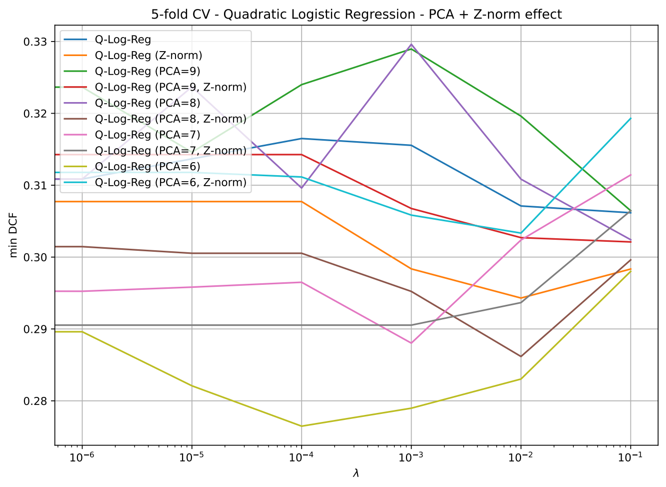

Figure 9: 5-fold CV min DCF of **Quadratic Logistic Regression** classifier applying (or not) **Z-Norm**, with different values of PCA

We continue our search for the best Quadratic Logistic Regression (Q-LR) by considering lower values for PCA in the aim for better results which the graph (Figure 10) shows. The best model remains the one found previously: Quadratic Logistic Regression with $\boldsymbol{\lambda = 10^{-4}}$ and $\textbf{PCA} = \textbf{6}$, without z-normalization. The min DCF is shown in Table 4.

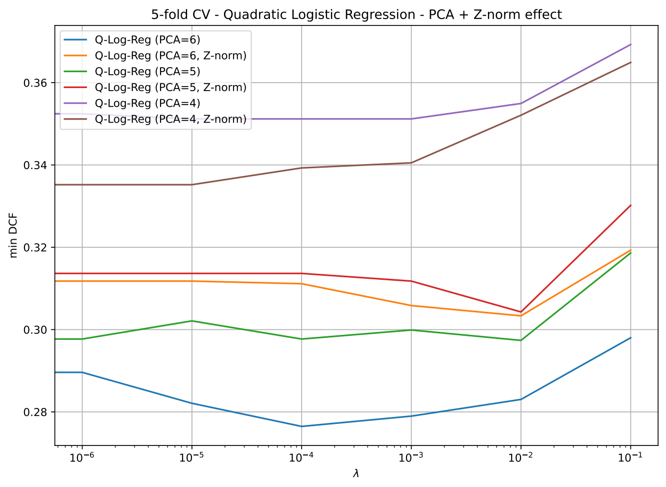

Figure 10: 5-fold CV min DCF of **Quadratic Logistic Regression** classifier applying (or not) **Z-Norm**, with different values of PCA

Remaining with the greedy approach, we try to re-balance our classes during

| PCA | $\lambda$ | min DCF |
|---|---|---|
| **6** | $\mathbf{10^{-4}}$ | **0.276** |

Table 4: 5-fold CV results of **Quadratic Logistic Regression** with $\boldsymbol{\lambda}$**=**$\mathbf{10^{-4}}$ **PCA=6** and no z-norm

training, using a **Prior-Weighted** model. Re-balancing our best model considers using values of prior $\pi_T$ close to our effective prior $\tilde{\pi} \approx 0.091$, so we analyze the cases: $\pi_T \in \{0.05, \tilde{\pi}, 0.2, 0.5\}$. In Table 5 are shown the $DCF_{min}$ values obtained in the different cases.

| $\pi_T$ | $\lambda$ | min DCF |
|---|---|---|
| - | $10^{-4}$ | 0.276 |
| 0.1 | $10^{-4}$ | 0.284 |
| $0.0\overline{90}$ | $10^{-4}$ | 0.280 |
| **0.2** | $10^{-4}$ | **0.275** |
| 0.5 | $10^{-4}$ | 0.279 |

Table 5: 5-fold CV results of **Diagonal Covariance MVG** (*Naive Bayes*) with different PCA dimensions

Considering the values provided in Table 5, all of the above results showcase great outcomes of $DCF_{min}$ with the best one for the Logistic Regression models being obtained by **Quadratic Logistic Regression** with $\boldsymbol{\lambda}$**=**$\mathbf{10^{-4}}$, **PCA=6** with a re-balancing of the original prior with the value of$\boldsymbol{\pi_T}$**=0.2**.

## 4.5 Support Vector Machines

We will now shift our attention to another type of classifier that is widely used in ML algorithms: the **Support Vector Machine (SVM)** model.

The constraint of risk minimization present within Logistic Regression can be extended to a more generic constraint that considers *margin separation* with the consideration of a specific margin between the two classes. This practical margin will be ultimately be guided by points that surround closely the hyperplane created called **support vectors**, which leaves some sort of liberty to a mis-classified points to be considered within the lost function within a certain range of acceptability and not being firmly labeled as wrong predicitions which leaves, to an extent a level of freedom not present within the LR model. This technique is publicly known as *Support Vector Machine (SVM)*. The objective function within SVM will solemnly rely on the dual formulation because of its simplicity by re-reducing the complexity of the problem down to the number of samples present in our dataset.

What makes SVM so favorable is that it allows us to compute **non-linear separation hyperplanes** without the specific need to expand features (unlike LR), but just levering **kernel functions**. A *kernel function* is a function that computes the dot product between two samples in an expanded feature space by means of simple dot products, and therefore being very efficient. We can thus express the SVM objective function in terms of dot products, and just apply a kernel function to them in order to look for a different kind of separation surface.

The SVM dual objective function is:

$$J_D(\boldsymbol{\alpha}) = -\frac{1}{2}\boldsymbol{\alpha}^T\mathbf{H}\boldsymbol{\alpha} + \boldsymbol{\alpha}^T\mathbf{1} \tag{4}$$

This equation assumes that $\boldsymbol{\alpha}$ is a vector with elements $\alpha_i$ for $i$ in the range from 1 to $n$, where $0 \leq \alpha_i \leq C$ for all $i$, and that $\sum_{i=1}^{n} \alpha_i z_i = 0$.

Different SVM variants that will consider the linearity and non-linearity models are presented as the following:

- **Linear Support Vector Machine (SVM)**: At their core, Linear SVM seeks to find the optimal hyperplane that best separates two classes within a dataset. This hyperplane is determined in such a way that it maximizes the margin, which is the distance between the hyperplane and the nearest data points of each class. By mapping the data into a higher-dimensional space, they transform complex nonlinear problems into linear ones, effectively finding linear decision boundaries that can separate different classes with great accuracy.
- **Kernel Support Vector Machine (SVM)**: The dual formulation of SVM in 4 relies on data samples and their dot products, expressed as $H_{ij} = z_i z_j x_i^T x_j$.

  This property streamlines score computation without the need for explicit feature expansion. What's essential is the ability to efficiently calculate dot products between training and test samples. With a function capable of performing these dot product calculations effectively in the expanded space, we can leverage a *kernel function*, denoted as $k$, both during training and when scoring new data points.

  This approach, $k(x_i, x_j) = \Phi(x_i)^T\Phi(x_j)$, empowers us to identify a linear separation boundary within the augmented feature space, corresponding to a non-linear separation boundary in the original feature space.

  We will consider two types of **kernel SVM models**, using the following kernels:

– **Radial Basis Function (RBF)**:

$$k(x_1, x_2) = e^{-\gamma \|x_1 - x_2\|^2} \tag{5}$$

– **Polynomial Kernel Function**:

$$k(x_1, x_2) = (x_1^T x_2 + c)^d \tag{6}$$

We start our analysis of SVM with the results obtained by its Linear variance. We use cross validation to test different values of the hyperparameters $C$ and to test different values of PCA, while also considering the analysis on the effect of Z-Norm pre-processing (Figure 11). Our prediction concerning Support Vector Machine (SVM) aligns with the considerations performed on Logistic Regression (LR) as we do firmly believe that the non-linear versions of SVM (i.e. Polynomial SVM and RBF kernel SVM) will significantly outperform the Linear SVM.

With An initial inspection of Figure 11, we can confirm our prediction as the results of $DCF_{min}$ are generally poor with a major consideration that both pre-processing techniques,Z-Norm or PCA, do not have an impact or improvement of the results of the Linear model.

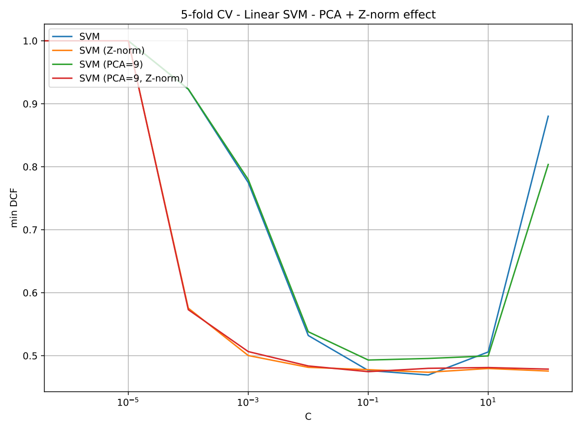

Figure 11: 5-fold CV min DCF of **Linear SVM** classifier applying (or not) **Z-Norm**, with PCA=9

We move forward with a deep examination of the SVM with 2D Polynomial kernel, adopting different PCA strategies and evaluating again the effect of Z-Norm. In Figure 12, we display the results of $DCF_{min}$.

An initial scan of the graph in hand showcase a significant improvement in the results of $DCF_{min}$ with the best results are considered for the parameter $\boldsymbol{C}$ =

$\mathbf{10^{-2}}$ with **PCA**=**8** while **Z-Norm** is being adopted, as $\boldsymbol{DCF_{min}}$=**0.268**. We do observe promising results for the SVM hyperparameter $C$, being between $10^{-3}$ and 1 (included), in both the linear or kernel case. We will currently shift our focus to consider solemnly these values of $C$.

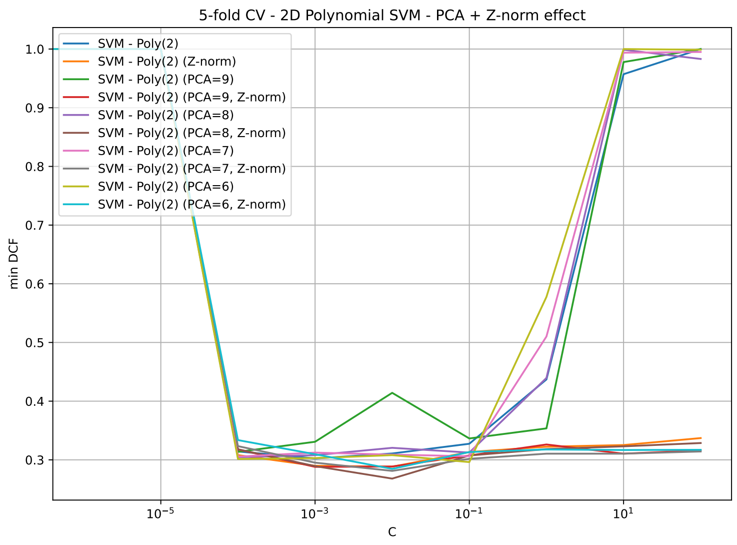

Figure 12: 5-fold CV min DCF of **2D Polynomial kernel SVM** applying (or not) **Z-Norm**, with different values of PCA

We further our attention on the non linear SVM models, analyzing **3D polynomial kernel** and **Radial Basis Function (RBF) kernel SVM**.

With the 3D Polynomial kernel, we consider different configurations of the hyperparameter $C$ as well as some different variation of PCA, we do not forget to apply the effect of z-norm on the model. In Figure Figure 13 can be found the results of $DCF_{min}$.

An initial impression could be presented regarding **Z-Score Normalization**, as the graph showcases better values computed for $DCF_{min}$ when it is applied. We do also observe an obvious point made within the figure as hyper-parameter $C = 10^{-3}$ performs promising results, obtaining a $DCF_{min}$=**0.291** with Z-Norm and **no PCA**, as the best result for **3D Polynomial kernel SVM**.

We will consider the same conditions for RBF SVM as we will focus on $C$ values between $10^{-3}$ and 1. Following a *greedy* approach, based on the most promising configuration of the Polynomial SVM, we will start the analysis of the case with PCA=8, applying or discarding Z-Norm.

Knowing that RBF SVM considers a hyper-parameter $\gamma$, the best results, ac-

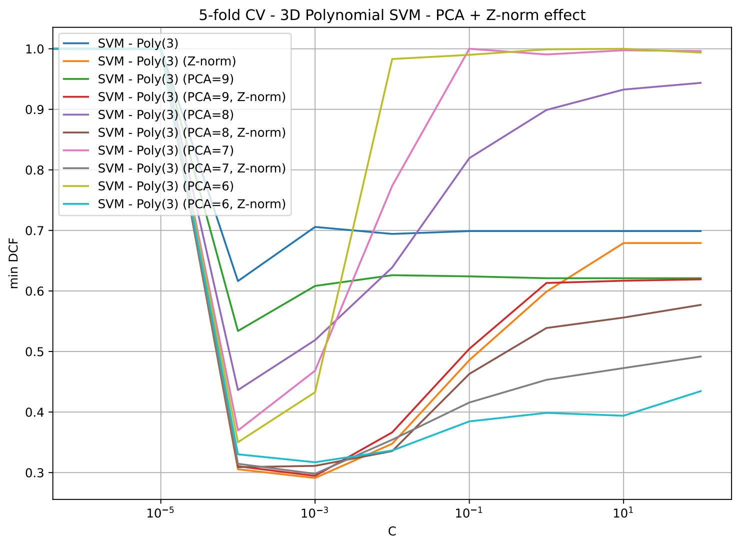

Figure 13: 5-fold CV min DCF of **3D Polynomial kernel SVM** applying (or not) **Z-Norm**, with PCA=8

cording to the graph, is when the hyper-parameter consider a value of $10^{-3}$. Adding to it the values of the hyperparameter $C$=1 and PCA=8 (no Z-Norm), we obtain the most optimal $DCF_{min}$=0.297 out of all the proposed models.

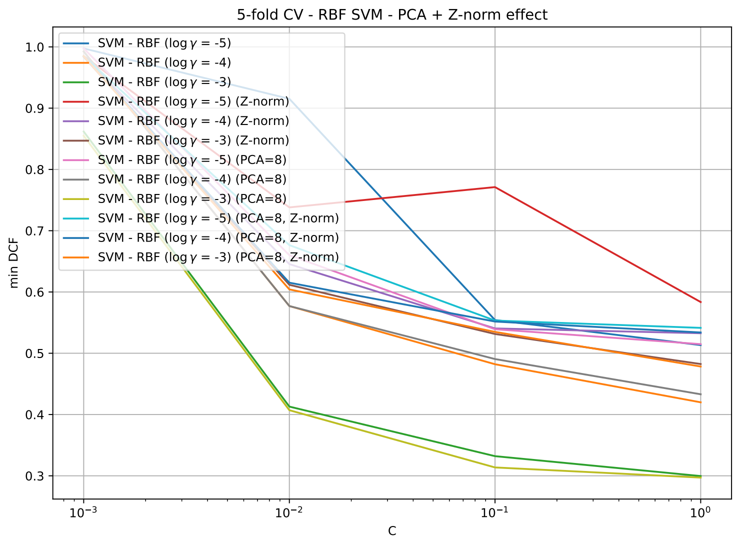

Figure 14: 5-fold CV min DCF of **RBF kernel SVM** applying (or not) **Z-Norm**, with PCA=8, and $\log\gamma = -3, -4, -5$.

We decide to further analyze this configuration by still considering a greedy approach, exploring higher values of $C$ for $\gamma = 10^{-3}$ and different values of PCA in (Figure 15). We remark that the value of PCA=8 remains the best model for $\gamma = 10^{-3}$, however we detect a better result for $DCF_{min}$ for $C$=10 as it seems that the increase in the value of the hyper-parameter has a direct relation with the increasing performance of the model.

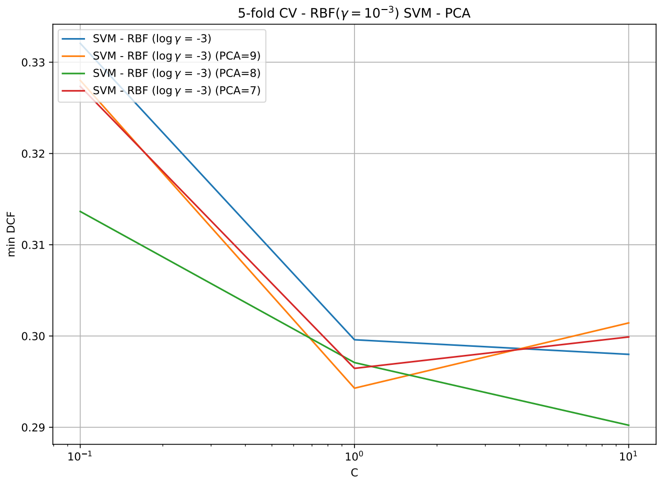

Figure 15: 5-fold CV min DCF of **RBF kernel** ($\gamma = 10^{-3}$) **SVM**, with PCA=7,8,9

In order to test the assumption made above, we decide to analyze higher values of $C$ shown in (Figure 16). We do find that the graph portrayed falsify our assumption on hyper-parameter $C$ as the values of $DCF_{min}$ worsen drastically for the same configuration considered.

We can now finally represent the model being the most performing: **RBF kernel SVM** obtains $\boldsymbol{DCF_{min}}$=**0.290**, with $\boldsymbol{C}$=**10**, $\boldsymbol{\gamma}$=$\mathbf{10^{-3}}$, **PCA=8** and no Z-Norm.

In the following table (Table 6) we perform a brief summary of the best SVM classifiers so far. We will only consider the best among the 3 (SVM with 2D polynomial kernel) and try to re-balance classes during training.

| SVM model | C | PCA | z-norm | min DCF |
|---|---|---|---|---|
| 2D poly | $10^{-2}$ | 8 | yes | **0.268** |
| 3D poly | $10^{-3}$ | no | yes | 0.291 |
| RBF ($\gamma = 10^{-3}$) | 10 | 8 | no | 0.290 |

Table 6: Best SVM classifiers

Similar to the case of Quadratic Logistic Regression, we use training true prior

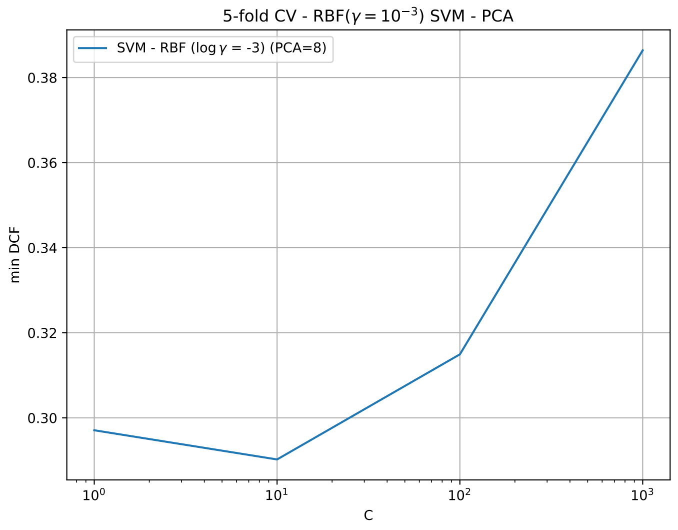

Figure 16: 5-fold CV min DCF of **RBF kernel** ($\gamma = 10^{-3}$) **SVM**, with PCA=8

values $\pi_T \in \{0.05, \tilde{\pi}, 0.2, 0.5\}$. The results are shown in Table 7.

Re-balancing the classes in our case does not improve the performances of the model: the best results are still obtained without considering a re-balancing ($DCF_{min} = 0.268$), although a prior of $\pi_T = 0.2$ displays near-enough results ($DCF_{min} = 0.274$).

| $\pi_T$ | min DCF |
|---|---|
| - | **0.268** |
| 0.05 | 0.306 |
| $0.0\overline{90}$ | 0.291 |
| 0.2 | 0.274 |
| 0.5 | 0.276 |

Table 7: Performance of the **best 2D Polynomial SVM** model with and without class rebalancing, using different training true priors $\pi_T$

.

## 4.6 Gaussian Mixture Models

Lastly we will shift our focus on considering **Gaussian Mixture Model (GMM)** as potential candidates for best model classifiers.

**Gaussian Mixture Model (GMM)** is another example of *generative classifier* that allows us to approximate the density of a random variable (R.V.) $X$ when the exact density of $X$ is unknown. The GMM density is represented as a weighted sum of $G$ Gaussian distributions:

$$X \sim \text{GMM}(M, S, w) \implies f_X(x) = \frac{1}{G}\sum_{g=1}^{G} w_g \mathcal{N}(x|\mu_g, \Sigma_g) \tag{7}$$

Here, $M$, $S$, and $w$ correspond to the means $(\mu_1, \ldots, \mu_G)$, covariance matrices $(\Sigma_1, \ldots, \Sigma_G)$, and weights $(w_1, \ldots, w_G)$ of the individual Gaussian components, respectively.

The GMM classifier therefore estimates the distributions of the two classes, assuming them as a weighted sum of Gaussian densities of respectively $G_T$ and $G_F$ sub-components; then, using the same approach as the standard Multivariate Gaussian classifier, computes the class posterior probabilities of the test samples for the two different classes, and assigns the label with the highest probability. The two numbers of components for the two classes are hyperparameters of the model which have to be supposed in advance.

In the following analysis we will adopt the **LBG** algorithm, to iteratively build families of classifiers, at each iteration doubling the number of components of one class to estimate the results of a new configuration. Furthermore, to not incur in degenerate solutions, we will constrain the eigenvalues of the covariances to be at least **0.01**.

The convenience of using GMM rotates around the consideration of a **Gaussian distribution** represented by diverse clusters called components. This comes in very handy with the assumption made of a potential multi-regional distribution present within the *spoofed fingerprint dataset* as it is collected based on 6 sub-classes on different spoofing techniques, while the *authentic fingerprint dataset* already emitted a promising **Gaussian distribution** showcased directly within the feature analysis.

Leveraging our data sources, there is a firm believe that higher components within the *spoofed fingerprint class* will entertain better performances on the model while the same is not necessary with the target class. In particular, with the spoofed class coming from 6 different clusters, while the authentic class coming from a single homogeneous cluster, we expect to have better results with 4-8 components for the *false* class, and very few components for the *true* class.

We start our analysis the results obtained by GMM, by selecting values of sub-components with the values 1,2,4,8,16 for both classes. We will carefully analyze all the possible combinations for both classes.

We perform this analysis again by considering PCA=9 and PCA=8. The values of PCA adopted are presented as the most promising data processing strategies in the MVG classifier. In Table 8 are shown the results **without PCA**.

| $G_T$ | $G_F$ | min DCF |
|---|---|---|
| 1 | 1 | 0.331 |
| 1 | 2 | 0.283 |
| **1** | **4** | **0.252** |
| **1** | **8** | **0.263** |
| 1 | 16 | 0.306 |
| 2 | 1 | 0.333 |
| 2 | 2 | 0.280 |
| **2** | **4** | **0.260** |
| **2** | **8** | **0.265** |
| 2 | 16 | 0.304 |
| 4 | 1 | 0.338 |
| 4 | 2 | 0.289 |
| **4** | **4** | **0.266** |
| 4 | 8 | 0.281 |
| 4 | 16 | 0.316 |
| 8 | 1 | 0.369 |
| 8 | 2 | 0.326 |
| 8 | 4 | 0.310 |
| 8 | 8 | 0.324 |
| 8 | 16 | 0.355 |
| 16 | 1 | 0.441 |
| 16 | 2 | 0.417 |
| 16 | 4 | 0.364 |
| 16 | 8 | 0.358 |
| 16 | 16 | 0.389 |

Table 8: 5-fold CV - GMM with 1,2,4,8,16 components (with Full Covariance) using LBG, **no PCA**

The best results are obtained by considering **sub-components 1-2** for the *true* class and **sub-components 4-8** for the *false* class. It is important to note that this observation complies with the initial assumption made previously.

Keeping a consistent technique with the greedy approach, we will shift our study on the 2 candidate values for the sub-components for the respective classes, found in our previous analysis.

We know observe the effect of applying PCA on these initial model configurations, using PCA = 8 (Table 10) and PCA = 9 (Table 9).

With a short examination of the graph, we are able to notice that PCA does not improve the performance of the model in a significant way, but it instead removes useful information from the samples producing slightly worse results.

| $G_T$ | $G_F$ | min DCF |
|---|---|---|
| **1** | **4** | **0.262** |
| 1 | 8 | 0.291 |
| 2 | 4 | 0.278 |
| 2 | 8 | 0.304 |

Table 9: 5-fold CV - GMM with 1,2,4,8,16 components (with Full Covariance) using LBG, **PCA=9**

| $G_T$ | $G_F$ | min DCF |
|---|---|---|
| **1** | **4** | **0.267** |
| **1** | **8** | **0.267** |
| 2 | 4 | 0.272 |
| 2 | 8 | 0.272 |

Table 10: 5-fold CV - GMM with 1,2,4,8,16 components (with Full Covariance) using LBG, **PCA=8**.

Since the configuration with $G_T$=1 components for the *true* class and $G_F$=4 components for the *false* class explores a minimal cost consistently, giving the best results without PCA for ($\boldsymbol{DCF_{min}}$=**0.252**), we now decide to tackle this configuration only.

Keeping the greedy approach, we know consider some variants of the GMM classifier, which represent similar variants of the standard MVG model. We do intend to specifically estimate the class distributions considering not only Full Covariance sub-components, but Tied Covariance and Diagonal Covariance ones, for both classes. However, we do not expect to obtain huge improvements on the previous results, with our PCA analysis already showing that the GMM classifier do not overfit the features.

In Table 11 are shown all the possible results for the combinations of these variants, with $G_T$=1 and $G_F$=4. Considering just 1-GMM component for the *target* class, the tied model and the full covariance model will display equal results, thus the diagonal covariance variant is deeply considered.

With the different variants taken into consideration in our final study of GMM, it is clear in our case that the best performance remains given by the **Full Covariance (FC)** variant for both classes, with a cost $\boldsymbol{DCF_{min}}$=**0.252**. This satisfactory results coincides well with the Pearson correlations analyzed in the feature analysis part as, although the *target* class does interpret a weaker correlation than the *non-target* one, the general visualization still showcases a similar correlation between them.

| $G_T$ var. | $G_F$ var. | min DCF |
|---|---|---|
| **1 (FC)** | **4 (FC)** | **0.252** |
| 1 (FC) | 4 (TC) | 0.265 |
| 1 (FC) | 4 (DC) | 0.273 |
| 1 (FC) | 4 (TDC) | 0.281 |
| 1 (DC) | 4 (FC) | 0.284 |
| 1 (DC) | 4 (TC) | 0.290 |
| 1 (DC) | 4 (DC) | 0.274 |
| 1 (DC) | 4 (TDC) | 0.284 |

Table 11: 5-fold CV - GMM **variants** with $G_T$=1 and $G_F$=4 components using LBG, **no PCA**. FC = Full Covariance, TC = Tied Covariance, DC = Diagonal Covariance, TDC = Tied Diagonal Covariance.

## 5 Model Comparisons

After a thorough model analysis performed, we compare the optimal results found for all the different classifiers studied. Table 12 contains the best models identified, for each genre of classifiers, with its specific configuration of hyperparameters and pre-processing techniques.

| | min DCF | |
|---|---|---|
| MVG - FC<br>(PCA = 9) | 0.330 | |
| **Q-LR**<br>($\lambda = 10^{-4}$, PCA = 6, $\pi_T = 0.2$) | **0.275** | [3] |
| **2D-Poly SVM**<br>($C = 10^{-2}$, PCA = 8, z-norm) | **0.268** | [2] |
| 3D-Poly SVM<br>($C = 10^{-3}$, no PCA, z-norm) | 0.291 | |
| RBF SVM<br>($C = 10$, $\gamma = 10^{-3}$, PCA = 8, no z-norm) | 0.290 | |
| **GMM**<br>($G_T = 1$ (FC) - $G_F = 4$ (FC), no PCA) | **0.252** | [1] |

Table 12: Best model for each family of classifiers.

### 5.1 Most promising models

We decide to further our studies by selecting the three best model out of all the possible candidates available. The top-3 candidates, with 1 referring to the best, are the following:

1. **Gaussian Mixture Model with $G_T = 1$ and $G_F = 4$** (no PCA), with a **minimum DCF of 0.252** [1].
2. **2D Polynomial SVM** with $C = 10^{-2}$, PCA=8 and Z-Norm with a **minimum DCF of 0.268** [2].
3. **Quadratic Logistic Regression** model, with $\lambda = 10^{-4}$, PCA=6 and $\pi_T$=0.2 with a **minimum DCF of 0.275** [3].

To reflect on the goodness of classifiers considering diverse target applications, compare the performance of our top-3 models using a **Detection Error Tradeoff (DET) plot**, showing the relationship between False Negative Rate and False Positive Rate in (Figure 17).

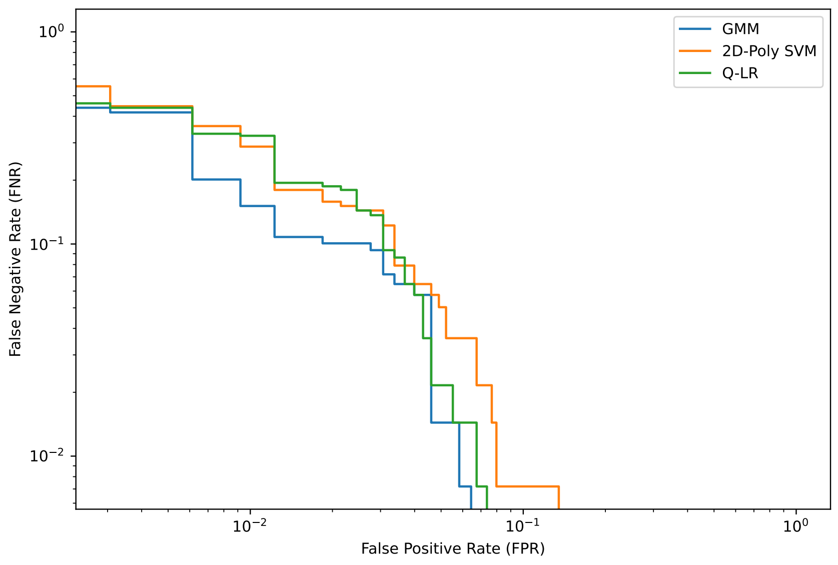

Figure 17: Best 3 models **DET plot** (axis are in log-scale).

By taking a look at the graph, we can clearly detect a better performance shown by the Gaussian Mixture Model (GMM) model, in fact, while varying the thresholds considered, we can see that the model kept an acceptable trade-off between the False Positive Rate (FPR) and False Negative Rate (FNR). The performance of the GMM model is also showcased by a steeper curve presented by the graph where it can be considered as a good model for applications that require lower FNR.

# 6 Model Calibration

To perform the model evaluation of the diverse models considered, we have only considered the **Minimum Detection Cost Function (DCF)** for the target application prior $\tilde{\pi} = 0.0\overline{90}$ as a reference metric in our study. However this metric considers the threshold which gives the best result of the classifiers, among all the test scores produced by the model during the *cross-validation* phase.

To confirm if this threshold coincides with the theoretical one and thus complies with the practical result, we will consider **the actual DCF** which refers to the actual cost produced by our classifier over the evaluated samples. The gap between the minimum DCF and the actual one is related to the **mis-calibration** of the model, which is due to multiple factors, like the non-probabilistic interpretation of the scores for some models (like SVM), but also **overfitting** or **underfitting** issues for the classifier.

Since our target application requires primarily just one working point ($\pi_T = 0.5, C_{fn} = 1, C_{fp} = 10)$), we could consider halting our analysis, and solemnly adopt the best classifiers found so for with respect to the $DCF_{min}$, using the *threshold* corresponding to the optimal DCF. However, using classifiers with fixed thresholds has the disadvantage of making them exclusively useful for the application prior used to compute those thresholds. We would like to have a more robust model which comes in handy in slightly different contexts, with different working points.

For the reasons stated above, we perform a **score calibration** phase to reduce the gap between $DCF_{min}$ and actual DCF for our best classifiers, making them more robust.

## 6.1 Prior-Weighted LR best models calibration

The **score calibration** technique will revolve around a **Prior-weighted Logistic Regression (LR)**: it is considered as a second model that is fed with the initial scores computed by our different candidate classifiers. Using the prior weights in hands, it learns to adjust the raw scores to provide calibrated ones. It then uses these new scores to calculate our evaluation metrics (i.e. Minimum DCF and actual DCF) due to its practical reflection as true likelihoods of a data sample belonging to one of our two classes. It is important to highlight that this technique is still performed under our traditional **K-fold cross validation**.

To train the LR calibrator, we will employ 3 values of prior close to our effective prior $\tilde{\pi}$, so $\pi_T \in [0.05, 0.0\overline{90}, 0.2]$.

In Table 13 are shown the values of minimum DCF, actual DCF and error rate for our best classifiers, considering the target prior $\tilde{\pi} = 0.0\overline{90}$. We will refer to them as just GMM, 2D-Poly SVM and Q-LR for simplicity purposes, however we are still considering to the models denoted as [1], [2] and [3] as in Table 12.

| | min DCF | actual DCF | error rate |
|---|---|---|---|
| GMM | 0.252 | 0.258 | 0.054 |
| 2D-Poly SVM | 0.268 | 0.390 | 0.045 |
| Q-LR | 0.275 | 0.306 | 0.063 |

Table 13: Minimum DCF, actual DCF and error rates values obtained by 3 best classifiers for $\boldsymbol{\tilde{\pi} = 0.0\overline{90}}$ in 5-fold CV phase

Considering our target application, GMM [1] can be portrayed as well-calibrated since the difference between $DCF_{min}$ and actual DCF is minimal. As second step, the Q-LR model [3] display some less considerable mis-calibration with the respect to our evaluation metrics as it is still considered an acceptable model. However, on the other hand, the SVM classifier [2] has shown some evident mis-calibration that could not be neglected. This does align however with our initial thought that SVM will show mis-calibration signals due to its non-probabilistic interpretation.

We shift our attention by employing a **Bayes error plot** of the 3 classifiers, considerably showing the mis-calibration at hand within the proposed candidates (Figure 18).

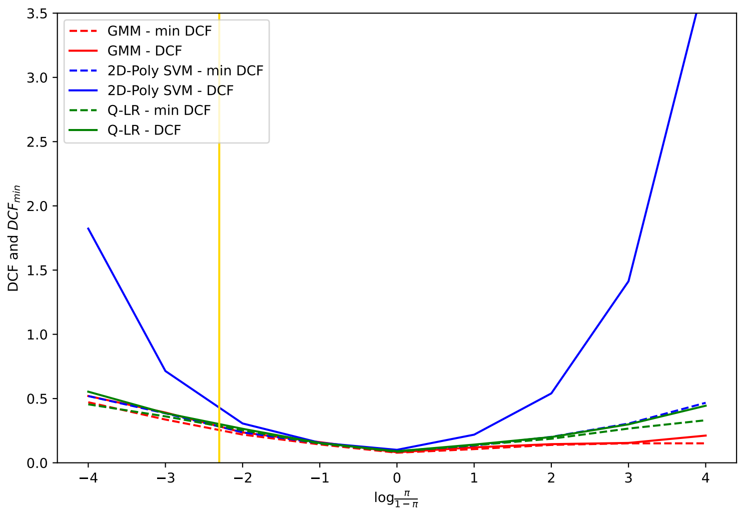

Figure 18: Best 3 (not calibrated) models *Bayes error plot.*

As stated within our analysis on Table 13, both GMM and Q-LR appear much calibrated, with the initial one being considered as a better calibrated model, specifically taking into consideration the SVM. In fact, the latter model showcases a considerably relevant area of curvature between actual DCF and min DCF, which is blatantly demonstrated for unbalanced classification tasks ($\tilde{\pi} \ll 0.5$ or $\tilde{\pi} \gg 0.5$). In yellow, we provide the plotted results of the effective prior of our target application $\tilde{\pi} = 0.0\overline{90}$ with a **prior log odd = -2.302**, where we can distinctively observe that the results displayed occupies the same conditions as the all the range of effective priors with a small consideration towards an ameliorated condition of the model [1] and [3] with respect to higher effective priors.

In Table 14 are shown the results obtained by our best 3 models **after score calibration**, with the 3 mentioned training priors $\pi_T \in [0.05, 0.0\overline{90}, 0.2]$.

| | $\pi_T$ | min DCF | actual DCF |
|---|---|---|---|
| GMM | 0.05 | 0.255 | 0.268 |
| | $0.0\overline{90}$ | 0.255 | 0.268 |
| | 0.2 | 0.253 | 0.268 |
| 2D-Poly SVM | 0.05 | 0.272 | 0.294 |
| | $0.0\overline{90}$ | 0.272 | 0.299 |
| | 0.2 | 0.271 | 0.293 |
| Q-LR | 0.05 | 0.281 | 0.302 |
| | $0.0\overline{90}$ | 0.281 | 0.308 |
| | 0.2 | 0.280 | 0.305 |

Table 14: Best 3 models performances after **Prior-Weighted LR score calibration** using $\pi_T = [0.05, 0.0\overline{90}, 0.2]$. Min DCF and actual DCF values for $\tilde{\pi} = 0.0\overline{90}$.

Considering each candidate model separately, we can directly observe that a variation of the training prior, within the calibration phase, does not significantly affect the results of our models as the difference between these metric remains of the order of $10^{-3}$.

Now if we dive deeper within our target application ($\tilde{\pi} = 0.0\overline{90}$), the GMM model [1] obtains an actual DCF of **0.268**, which is considered as slightly worse than prior to performing calibration (0.258). However the remaining models showcase the benefit from the calibration with a better performance with an actual DCF = **0.293** (for $\pi_T = 0.2$) considering the 2D-Poly SVM [2] instead of 0.390, and an actual DCF = **0.302** (for $\pi_T = 0.05$) instead of 0.306 considering the Q-LR [3].

We choose to employ $\boldsymbol{\pi_T}$**=0.2** for our final models, and we analyze their new performances, after the score calibration phase, with another *Bayes error plot* in Figure 19 (which has the same scale as the previous plot in Figure 18).

We do observe that all 3 candidate classifiers have been calibrated for a wider range of working points, with a special consideration going towards SVM which originally was the worst performance wise.

And to better distinguish the behaviour of the three different models, a detailed plot can be visualized in Figure 20, where we can further elaborate on the improvement the three candidates have performed. And with the yellow line we can identify our target prior $\tilde{\pi} = 0.0\overline{90}$ where we can evidently distinguish a trend that explains the concept of mis-calibration being not significantly evident however visible-enough to provide a clear explanation of the general situation of the 3 calibrated models.

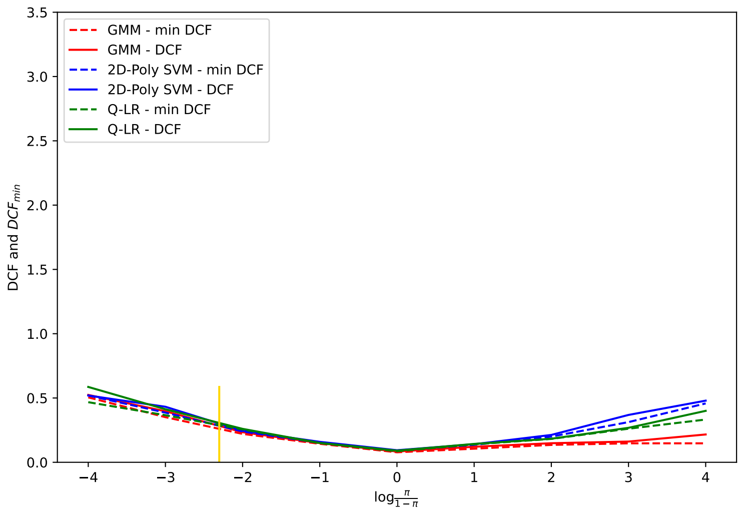

Figure 19: *Bayes error plot* of the best 3 **calibrated** models with $\pi_T = 0.2$.

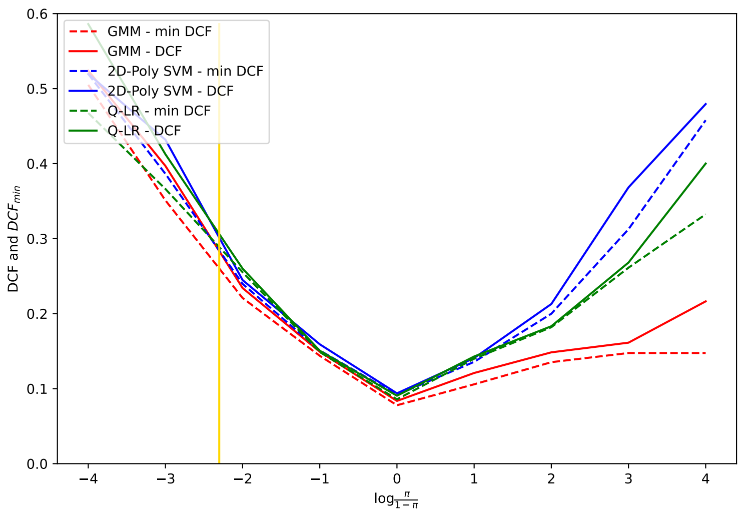

Figure 20: *Bayes error plot* of the best 3 **calibrated** models with $\pi_T = 0.2$ - Detail.

## 6.2 Fusion of the best models

We now shift our attention by studying a **fusion model** that considers all possible combinations of the top-3 candidate models whether it is a 2-model implementation combination or a 3-model implementation.

The first thought that comes while analysing the results in Table 15, is a fast comparison between the model performances considering the minimum DCF, where it is important to highlight that the presence of the **GMM model [1]** within the fusion model increases significantly the results of the performance.

However, although the difference in performance is quite narrow between the fusion models where GMM is present, we do actually visualize it's importance when it comes to a comparison in the actual DCF, with as the model without GMM has shown a higher possibility of overfitting the samples.

We do also signalize that the fusion models do remain quite calibrated as the concept of fusion considers a calibration process where the scores are repeatedly closer to their real values, and thus the mis-classification stayed within an acceptable order of $10^{-3}$.

Conceptualizing the results, we were able to demonstrate that our target application reaches an optimal outcome with a combination of the 3-model, model [1] + [3] or with considering **GMM** solemnly. We can then assume that performing a fusion on the model will be quite relevant for an optimal performance, as the 2-model fusion mentioned, which provides satisfactory results for calibration, is favorable as a potential candidate with an honorable consideration of the two remaining models as backup candidates.

| | min DCF | actual DCF |
|---|---|---|
| [1] GMM | 0.253 | 0.268 |
| [2] 2D-Poly SVM | 0.271 | 0.293 |
| [3] Q-LR | 0.280 | 0.305 |
| [1] + [2] + [3] | **0.253** | 0.267 |
| [1] + [2] | 0.254 | 0.267 |
| [1] + [3] | 0.256 | **0.260** |
| [2] + [3] | 0.278 | 0.286 |

Table 15: Best 3 **calibrated** models (with Prior-Weighted LR) and **fusion combinations** performances using training $\pi_T = 0.2$. Results for the target application $\tilde{\pi} = 0.0\overline{90}$.

If we take a look at the Bayes error plots referenced in Figure 21 and Figure 22, we can ultimately observe a great calibration process performed within the three candidate models cited earlier. A minimal representation of mis-classification is visible within the various models so we can conclude that calibration has been really effective. We can evidently notice within a detailed inspection, that the 3-model fusion is the model where calibration performs the worst, evidently as through our precedent analysis, we discovered that score calibration does not efficiently provide well-rounded results when the SVM model is included.

It is important to emphasize the great calibration process performed by our candidate models, specifically [1] + [3], with a wider range of effective priors it was able to present close results to what our theoretical threshold presented, with a small mis-calibration present just for effective priors $\tilde{\pi} \gg 0.5$, and for $\tilde{\pi} \ll 0.5$. In yellow, we provide the plotted results of the effective prior of our target application $\tilde{\pi} = 0.0\overline{90}$, as the results show a visible mis-calibration present within our target application.

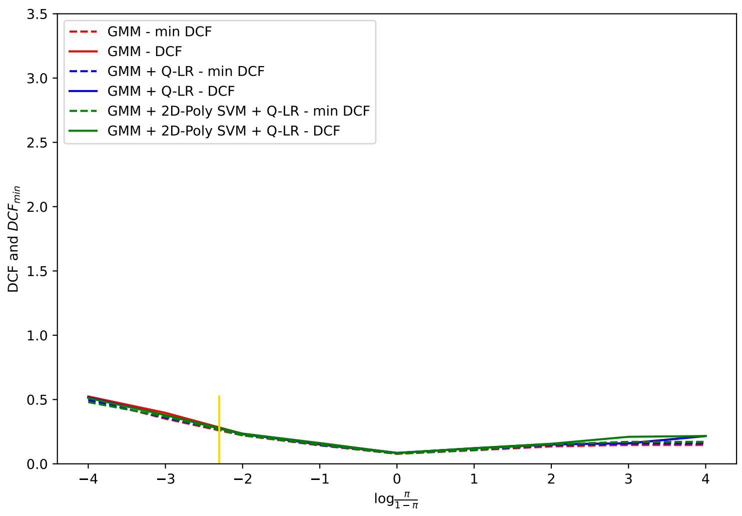

Figure 21: *Bayes error plot* of the **best 3 selected fusion models** (with $\pi_T = 0.2$), as **candidate models**. In yellow our application prior $\tilde{\pi}$.

## 7 Model Evaluation

We proceed to the testing phase of our study where we will analyze the performance of our models using an unseen **testing dataset**, which has never been used so far in the process of model selection. The goal now is to estimate how these models could behave in real-world scenarios, making predictions over unseen samples. The choices made so far to select the best models configurations, in fact, could not be always optimal for the real population. Later, we will try to compute an estimate of the

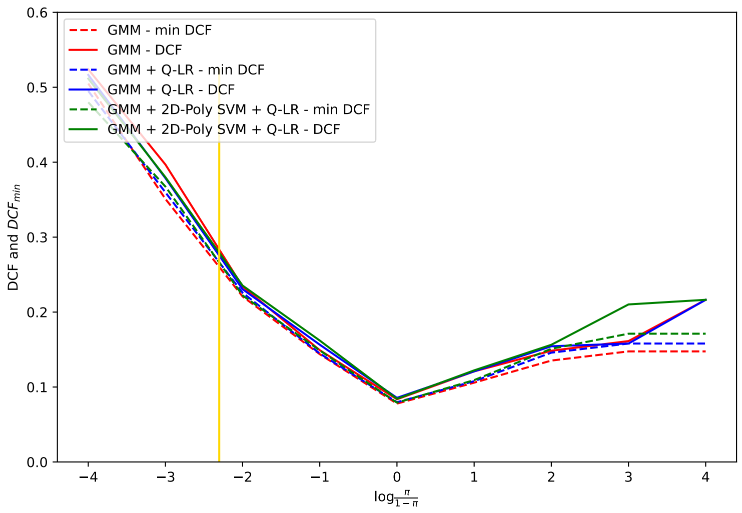

Figure 22: *Bayes error plot* of the **best 3 selected fusion models** (with $\pi_T = 0.2$), as **candidate models** - Detail.

costs produced by different models over the test set, in order to validate (or not) the choices made during the model selection phase.

Our main attention will be focused on the three main candidates for best model as we will evaluate their performance using our traditional metrics of:

- **Minimum Detection Cost Function (min DCF):** as a metric to compare the most optimal model out of our candidates.
- **Actual Detection Cost Function (actual DCF):** as a metric to portray the practical performance of our models in hand. This metric will also be decisive as an index of mis-calibration of our models on unseen data.

As considered for our model analysis, our main objective will be to consider the values over our target application, having an effective prior of $\tilde{\pi} = 0.0\overline{90}$. However, we do intend to deliver a robust model potentially useful for a wider range of applications, so we are still considering the effectiveness of the models over different working points.

## 7.1 Best models evaluation

As we elaborate on the results computed in Table 16, we can perceive that the GMM model is the most promising out of all the candidate models included with $\boldsymbol{DCF_{min}}$=**0.220** and a satisfactory performance, in practice, with an actual **DCF=0.227**. On the other hand, our considered fusion candidate [1] + [3] has slightly displayed

| | validation | | evaluation | |
|---|---|---|---|---|
| | min DCF | actual DCF | min DCF | actual DCF |
| [1] GMM | 0.253 | 0.268 | **0.220** | **0.227** |
| [2] 2D-Poly SVM | 0.271 | 0.293 | 0.257 | 0.278 |
| [3] Q-LR | 0.280 | 0.305 | 0.237 | 0.249 |
| [1] + [2] + [3] | 0.253 | 0.267 | 0.224 | **0.239** |
| [1] + [2] | 0.254 | 0.267 | 0.224 | **0.241** |
| [1] + [3] | 0.256 | 0.260 | **0.222** | **0.243** |
| [2] + [3] | 0.278 | 0.286 | 0.237 | 0.243 |

Table 16: Best 3 **calibrated** models (with Prior-Weighted LR) and **fusion combinations** performances using training $\pi_T = 0.2$ **over the evaluation set**. Results for the target application $\tilde{\pi} = 0.0\overline{90}$.

some lesser performance with an **actual DCF = 0.243** even though of a promising consideration with $DCF_{min} = 0.222$ would say otherwise, as the interpretation of the results were expected to be closer to what was portrayed by our GMM model.

However, as a general view considered on all the results, our three selected candidates ([1]+[3], [1]+[2]+[3] and [1]) performed significantly well in the evaluation phase with a notable mis-classification present in our fusion models, which satisfies the norm of acceptable results anyways.

We will now focus on analyzing the *Bayes error plots* of our diverse model candidates to further visualize their performance on our **evaluation dataset**.

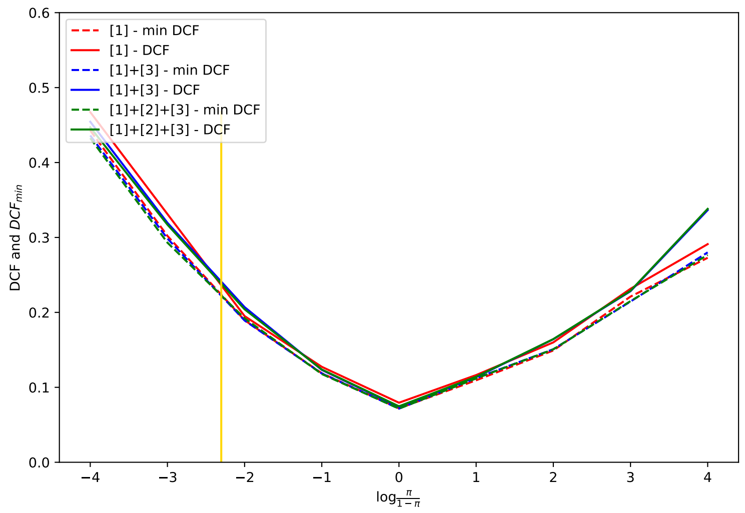

Figure 23: *Bayes error plot* of the **3 best fusion models** ([1]+[3], [1]+[2]+[3], [1]) over the **evaluation set**. In yellow our target application $\tilde{\pi}$.

Considering a wide range of working points, we can perceive that all three candidate models shown great calibration results as the difference between the minimum DCF and actual DCF is reduced to a minimal with the consideration of a small mis-calibration occurring for high value of the effective prior, specifically noticed for the fusion models. The yellow line provide the plotted results of the effective prior of our target application $\tilde{\pi} = 0.0\overline{90}$ and we can deduce from our graph the imminent presence of mis-calibration still present within our target application.

## 7.2 Post-evaluation analysis

In this section, we will perform a full analysis of the decisions adopted in the model selection phase, as it will be a valuable index to comprehend the potential considerations in terms of hyperparameter selections or pre-processing strategies were entitled to promising model candidatures.

However, we will just consider the best models selected in subsection 5.1, (i.e. [1], [2] and [3]), consequently responsible for the implementation of our optimal fusion models ([1]+[3] and [1]+[2]+[3]). This study targets the direct understanding if better choices within their configuration would have affected the fusion classifiers' results over the evaluation set. The minimum Detection Cost Function (DCF) will solemnly be our metric of reference as part of the analysis.

### 7.2.1 Best Q-LR model

We initiate our study by considering the **hyperparameter** variation on the evaluation set with the Q-LR [3] model, the results are shown in Figure 24.

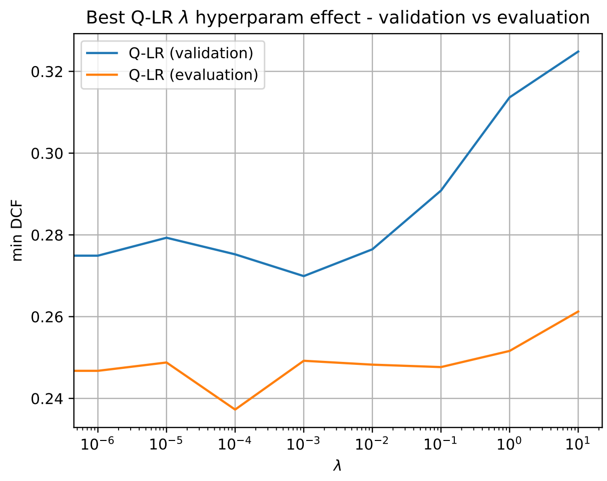

Figure 24: $DCF_{min}$ results of the **best Q-LR** model [3] over **validation and evaluation set**, for different values of **hyperparameter** $\lambda$

We detect that our choice of $\boldsymbol{\lambda} = \mathbf{10^{-4}}$on the training set, performed by the means of a greedy search of the configuration, was not theoretically optimal. And specifically adopting class re-balancing with $\pi_T = 0.2$, the best $DCF_{min}$ is obtained for $\lambda = 10^{-3}$. However, this latter selection of the $\lambda$ resulted in an optimal $\boldsymbol{DCF_{min}}$=**0.237** over the evaluation set. Thus, this hyperparameter will be kept for further analysis on the configurations of **PCA** and **Z-Norm**.

| PCA dim | validation | | evaluation | |
|---|---|---|---|---|
| | just PCA | PCA + Z-Norm | just PCA | PCA + Z-Norm |
| - | 0.308 | 0.299 | 0.277 | 0.274 |
| 9 | 0.317 | 0.315 | 0.283 | 0.264 |
| 8 | 0.304 | 0.310 | 0.269 | 0.252 |
| 7 | 0.298 | 0.287 | 0.258 | 0.258 |
| **6** | **0.275** | 0.308 | **0.237** | 0.273 |

Table 17: $DCF_{min}$ results of the **best Q-LR** model [3] over **validation and evaluation set**, for different **PCA** and **Z-Norm** strategies.

Through the results displayed in Table 17, we can firmly deduce that our initial consideration of **PCA = 6** and **no Z-Norm** was an optimal selection as the values presented within the minimum DCF clearly showcase the increase in performance when adopting this configuration.

In Table 18, we will inspect the effect of **class re-balancing** has on the results made on the evaluation set and compare it with the results obtained on the validation set.

| $\pi_T$ | validation | evaluation |
|---|---|---|
| - | 0.276 | 0.236 |
| 0.05 | 0.284 | 0.261 |
| $0.0\overline{90}$ | 0.280 | 0.254 |
| **0.2** | **0.275** | **0.237** |
| 0.5 | 0.279 | 0.240 |

Table 18: $DCF_{min}$ results of the **best Q-LR** model [3] over **validation and evaluation set**, for different **class re-balancing** strategies.

With a deterministic analysis, we observe that our choice of training the Q-LR model with a prior $\boldsymbol{\pi_T} = \mathbf{0.2}$, among the diverse examined class re-balancing priors was also optimal as it resulted with a $\boldsymbol{DCF_{min}}$=**0.237** for the evaluation set, which complies with the original result obtained on the validation set being the most performing within all the re-balancing strategies.

Following our traditional greedy strategy, the best model for our evaluation set aligns with model [2] (i.e. Quadratic Logistic Regression model with $\lambda = 10^{-4}$, PCA=6, no Z-Norm and training prior $\pi_T = 0.2$) present in our model selection phase.

### 7.2.2 Best SVM model

In this section we consider our **Support Vector Machine (SVM)** model [2] we have obtained in the model analysis section, and by varying the value of **hyperparameter** C, we analyze the effect the variance of this hyperparameter provides over both the validation and evaluation set.

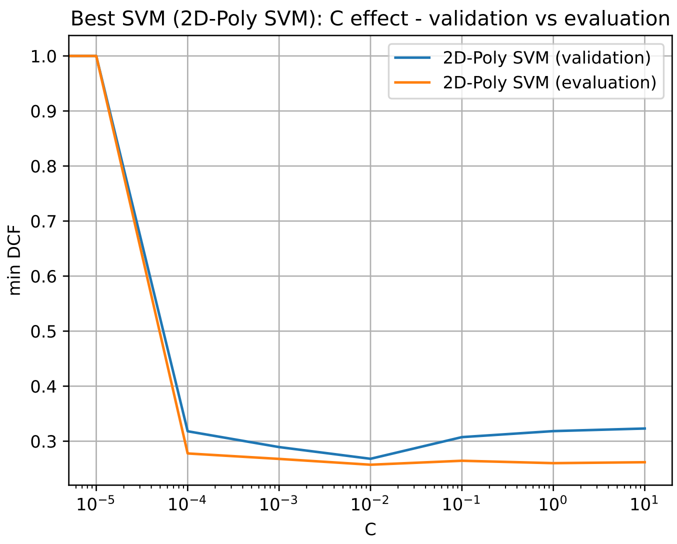

Figure 25: $DCF_{min}$ results of the **best SVM** model [2] over **validation and evaluation set**, for different values of **hyperparameter** C

According to Figure 25, we can clearly emphasize on the similar behavior the model performs whether it is on the validation or evaluation set. Now referring to the hyperparameter variation showcased, we can declare the our initial consideration for the validation set made for a hyper-parameter $\boldsymbol{C = 10^{-2}}$aligns with the optimal value of the hyperparameter found for an optimal $DCF_{min}$ = **0.268**.

We then will compare the impact pre-processing techniques, **PCA** and **Z-Norm**, and deduce the most optimal configuration available (Table 19). It is important to note that the values were computed according the best value for the hyperparameter $C = 10^{-2}$.

It is important to highlight that the best configuration found within the validation set remains the original configuration found in the analysis step where **PCA = 8 and Z-Norm is used** with $DCF_{min}$ = **0.268**. However, applying the diverse con-

| PCA dim | validation | | evaluation | |
|---|---|---|---|---|
| | just PCA | PCA + Z-Norm | just PCA | PCA + Z-Norm |
| - | 0.311 | 0.288 | 0.262 | 0.252 |
| 9 | 0.414 | 0.289 | 0.259 | 0.257 |
| 8 | 0.320 | **0.268** | 0.259 | 0.257 |
| 7 | 0.309 | 0.281 | 0.253 | 0.263 |
| **6** | 0.308 | 0.285 | **0.245** | 0.270 |

Table 19: $DCF_{min}$ results of the **best SVM** model [3] over **validation and evaluation set**, for different **PCA** and **Z-Norm** strategies.

figurations present to the evaluation set, we were able to locate a new optimum with **PCA = 6 and no Z-Norm** applied. We will thus employ this novel pre-processing configuration found in the next step.

Next up we will discover whether the selection of omitting **class re-balancing** in the validation set does hold when we consider re-balancing within the evaluation set. The values of the priors used comply with with the values used previously. Results are showed in Table 20.

| $\pi_T$ | validation | evaluation |
|---|---|---|
| - | 0.308 | 0.245 |
| 0.05 | 0.359 | 0.266 |
| $0.0\overline{90}$ | 0.321 | 0.251 |
| 0.2 | **0.296** | 0.250 |
| **0.5** | 0.302 | **0.240** |

Table 20: $DCF_{min}$ results of the **best SVM** model [2] over **validation and evaluation set**, for different **class re-balancing** strategies.

Within our original computation on the validation set, we were able to deduce a training prior $\pi_T = 0.2$ as an optimal result for $DCF_{min} = 0.296$. However, this result does not coincide with the observation made on the evaluation set, as it was found that a re-balancing performed with a prior $\boldsymbol{\pi_T = 0.5}$ was more effective with a $\boldsymbol{DCF_{min} = 0.240}$.

We can now formally interpret a new configuration that is quite diverse from the one found with the SVM model [2] that portrays a better performance classifier on the evaluation set. This model is introduced as: **SVM with 2D-Polynomial Kernel, $\boldsymbol{C = 10^{-2}}$, with PCA=6 and no Z-Norm adopted, with $\boldsymbol{\pi_T = 0.5}$**.

### 7.2.3 Best GMM model

We now shift our attention to our most promising candidate, the **Gaussian Mixture Model (GMM)** model [1]. Within this section, we will consider the number of components out of the list [1,2,4,8,16] where all the combinations possible will be used for both the *spoofed fingerprint class* and the *authentic fingerprint class.*

| $G_T$ | $G_F$ | validation | evaluation |
|---|---|---|---|
| 1 | 1 | 0.331 | 0.276 |
| 1 | 2 | 0.283 | 0.245 |
| **1** | **4** | **0.252** | **0.220** |
| **1** | **8** | **0.263** | **0.228** |
| 1 | 16 | 0.306 | 0.292 |
| 2 | 1 | 0.333 | 0.288 |
| 2 | 2 | 0.280 | 0.245 |
| **2** | **4** | **0.260** | **0.229** |
| **2** | **8** | **0.265** | **0.237** |
| 2 | 16 | 0.304 | 0.296 |
| 4 | 1 | 0.338 | 0.294 |
| 4 | 2 | 0.289 | 0.243 |
| 4 | 4 | 0.266 | 0.239 |
| 4 | 8 | 0.281 | 0.246 |
| 4 | 16 | 0.316 | 0.288 |
| 8 | 1 | 0.369 | 0.316 |
| 8 | 2 | 0.326 | 0.269 |
| 8 | 4 | 0.310 | 0.253 |
| 8 | 8 | 0.324 | 0.265 |
| 8 | 16 | 0.355 | 0.285 |
| 16 | 1 | 0.441 | 0.330 |
| 16 | 2 | 0.417 | 0.299 |
| 16 | 4 | 0.364 | 0.273 |
| 16 | 8 | 0.358 | 0.277 |
| 16 | 16 | 0.389 | 0.307 |

Table 21: $DCF_{min}$ results of the **best GMM** model [1] over **validation and evaluation set**, with **different number of clusters** for the two classes (**no PCA**).

As we have previously observed within our analysis of the GMM model [1], considering a number of components for the *target class* equal to **1-2** and a number of components for the *non-target class* equal to **4-8** is extremely efficient whether we use the validation or evaluation set, which resulted as an optimal model considered with [1,4] component of a $\boldsymbol{DCF_{min}} = \mathbf{0.252}$, for the validation set, and $\boldsymbol{DCF_{min}} =$ **0.220**in the evaluation set. This does align with our initial assumption that the *true class* is sufficiently represented with a maximum pair of gaussian-like components while the *false class* distinctively needs further consideration regarding the number of components.

After a thorough investigation of the optimal combination of the pair number of components, we move on to consider the precedent combination explained to evaluate the optimal pre-processing strategy involved within the evaluation set. We will, in fact, variate the number of dimensions in **PCA** between 8 (Table 23) and 9 (Table 22), with the previously fixed configuration to see its impact on the evaluation set.

| $G_T$ | $G_F$ | validation | evaluation |
|---|---|---|---|
| **1** | **4** | **0.262** | **0.219** |
| 1 | 8 | 0.291 | 0.227 |
| 2 | 4 | 0.278 | 0.224 |
| 2 | 8 | 0.304 | 0.231 |

Table 22: $DCF_{min}$ results of the **best GMM** model [1] over **validation and evaluation set**, with **PCA=9**.

| $G_T$ | $G_F$ | validation | evaluation |
|---|---|---|---|
| **1** | **4** | **0.267** | **0.218** |
| 1 | 8 | 0.267 | 0.219 |
| 2 | 4 | 0.272 | 0.218 |
| 2 | 8 | 0.272 | 0.222 |

Table 23: $DCF_{min}$ results of the **best GMM** model [1] over **validation and evaluation set**, with **PCA=8**.

With an primal observation made, both dimensionality reduction methods have produced close to optimal values when considering of validation set or evaluation set. However, it is notable to mention that with a **PCA = 8** we are able to find a better performance on the evaluation set with a $\boldsymbol{DCF_{min}} = \mathbf{0.218}$with a configuration ($G_T$=1, $G_F$=4) as number of components for the two classes.

We now finally turn our attention with a fixed configuration of a number of components equal to 4 for the *non-target class* and 1 for the *target class*, and with a PCA value fixed at 8, to investigate the performance of possible **variants** of the Gaussian Mixture Model (GMM) (Table 24).

In spite of the observation made within the validation that portrays the configuration $G_T = 1$ with Full Covariance (FC) and $G_F = 4$ with Diagonal Covariance (DC) as the optimal configuration, we were able to demonstrate through the evaluation set that the configuration $\boldsymbol{G_T}$**=1 with Full Covariance (FC)** and $\boldsymbol{G_F}$**=4 with Full Covariance (FC)**, using PCA=8, is the most performing with a $\boldsymbol{DCF_{min}} = \mathbf{0.218}$.

| $G_T$ var. | $G_F$ var. | validation | evaluation |
|---|---|---|---|
| 1 (FC) | 4 (FC) | 0.267 | **0.218** |
| 1 (FC) | 4 (TC) | 0.280 | 0.228 |
| 1 (FC) | 4 (DC) | **0.260** | 0.229 |
| 1 (FC) | 4 (TDC) | 0.274 | 0.257 |
| 1 (DC) | 4 (FC) | 0.270 | 0.234 |
| 1 (DC) | 4 (TC) | 0.275 | 0.238 |
| 1 (DC) | 4 (DC) | 0.269 | 0.243 |
| 1 (DC) | 4 (TDC) | 0.279 | 0.264 |

Table 24: $DCF_{min}$ results of the **best GMM** model [1] over **validation and evaluation set**, with **PCA=8** and **different gaussian combinations** for the two classes.

Thus we can finally deduce the final model of the GMM being: **Gaussian Mixture Model (GMM) with $G_T$=1 with Full Covariance (FC), $G_F$=4 with Full Covariance (FC) and PCA=8**.

### 7.2.4 Best models optimal configurations over evaluation set

In Table 25 we recap the results obtained over the evaluation set by our best 3 models and their fusion, both adopting the configurations choices coming from the model selection phase, and the ones related to the post-evaluation analysis.

| | chosen config. | | opt. config. on eval. set | |
|---|---|---|---|---|
| | min DCF | actual DCF | min DCF | actual DCF |
| [1] GMM | **0.220** | **0.227** | **0.218** | 0.235 |
| [2] 2D-Poly SVM | 0.257 | 0.278 | 0.240 | 0.248 |
| [3] Q-LR | 0.237 | 0.249 | 0.237 | 0.249 |
| [1]+[2]+[3] | 0.224 | **0.239** | **0.218** | **0.220** |
| [1]+[2] | 0.224 | 0.241 | **0.216** | **0.220** |
| [1]+[3] | **0.222** | **0.243** | **0.218** | **0.220** |
| [2]+[3] | 0.237 | 0.243 | 0.237 | 0.245 |

Table 25: $DCF_{min}$ and actual DCF results (for target application $\tilde{\pi} = 0.0\overline{90}$) of the **3 best models and their fusion** over the **evaluation set**, for both our **chosen configuration** and the **optimal configuration for the evaluation set**.

Finally, we can deduce, first and foremost that in comparison with the original configuration of the candidates and their fusion complements their optimal configuration has outperformed them with a significant margin, with a notable emphasis on the fusion models show a strong calibration index and with close to nothing mis-classification found. We can, thus consider the fusion, considering the optimal configuration found in subsection 7.2 the most optimized models within our study.

## 8 Conclusion

After a detailed verification made on the evaluation set, we will swiftly perform a brief overview on the results obtained for our candidate models within this report.

In the model analysis section, we were able to detect three potential candidate models that perform adequately with our training dataset and they were the following with their full configuration:

1. **Gaussian Mixture Model with $G_T = 1$ (FC) and $G_F = 4$(FC)** (no PCA), with a **minimum DCF of 0.252** [1];
2. **2D Polynomial SVM** with $C = 10^{-2}$, **PCA=8** and **Z-Norm** with a **minimum DCF of 0.268** [2];
3. **Quadratic Logistic Regression** model, with $\lambda = 10^{-4}$, **PCA=6** and $\pi_T$**=0.2** with a **minimum DCF of 0.275** [3].

We were able to distinguish a plausible advantage found within the classification model of the GMM. However we still needed to visualize the performance of our candidates, in practice, with the consideration of another evaluation metric **actual DCF**. We were able to identify the SVM model as a strong mis-calibrated model while the other two performed quite smoothly.

Then the consideration of **score calibration** came-in handy as the our actual DCF slowly converged closer to the ideal scenario present.

Additionally, we assisted the calibration of single models to an augmented solution where we detect the calibration performed by fusion of all the possible combination models. We were able to deduce the following calibrated models as the most performing models in terms of reduction of mis-classification, in decreasing order of performance:

- **A 2-combination fusion concerning the candidates: [1]+[3]**; with a **actual DCF of 0.260**
- **A 3-combination fusion of all our candidates: [1]+[2]+[3]** with a **actual DCF of 0.267**;
- **Gaussian Mixture Model with $G_T = 1$ (FC) and $G_F = 4$(FC)** (no PCA), with a **actual DCF of 0.268** [1].

As the performance surely increased and the margin of mis-classification tends to become narrower, it was time to see if our candidate models and all their possible fusions are able to be as performing as they were on the training data but now considering an unseen set of data called **evaluation set**. We were able to conclude that the candidate models sated above came in with satisfactory result with an honorable mention of the **Gaussian Mixture Model (GMM)** model [1] with a $DCF_{min}$ = 0.220 and actual DCF = 0.227. But at the end, all the models were, in general, able to increase their classification performance and reduce mis-classification.

At the end, and to further our analysis on the decisions taken with respect to hyperparameter selection and pre-processing conditions, we decided to check if the optimal models chosen in the training phase remained the same in the evaluation phase. We were able to visualize that some models were not optimally configured and thus there is some configurations that do perform better than the originally deduced ones. The most performant out of all these optimal models, considering the evaluation set, are the following:

- **[1]+[3]** with $\boldsymbol{DCF_{min}}$**=0.218** and **actual DCF=0.220**.
- **[1]+[2]+[3]** with $\boldsymbol{DCF_{min}}$**=0.218** and **actual DCF=0.220**.
- **[2]+[3]** with $\boldsymbol{DCF_{min}}$**=0.216** and **actual DCF=0.220**.

# List of Acronyms