Machine learning pattern recognition

-Fingerprint Spoofing Detection-

Andrea Sillano - s314771 & Riccardo Renda - s310383

Politecnico
di Torino

ANNO ACCADEMICO 2022-2023

Contents

Chapter 1

Introduction

1.1 Abstract

The aim of this report is to analyze the Fingerprint Spoofing Detection dataset. We will exploit some of the most common Machine Learning Models and study how well they perform on the given dataset. Each model will be ranked based on its DCF values. In the end we will obtain the most suitable model for the provided dataset.

1.2 Dataset Overview

The dataset is composed by means of embeddings of fingerprint images either *authentic* or *spoofed*. The dataset is splitted in Train with 2325 samples and Test with 7704 samples, both datasest are unbalanced. The Training one consists of 800 *authentic* and 1525 *spoofed*, the Test instead 2400 *authentic* and 5304 *spoofed*.
The *spoofed* samples belongs to 6 different sub-classes which correspond to different spoofing methods, but informations about the sub-classes are not available. The features don't have any physical interpretation. During our analisys, the training set will be used for training and validating the models while in the end the test set will be exploit only for the evaluation part.

1.3 Feature Analisys

Every sample in the dataset is represented by **10 features**. Initially we will analize the RAW features plotting histograms and scatters diagram:

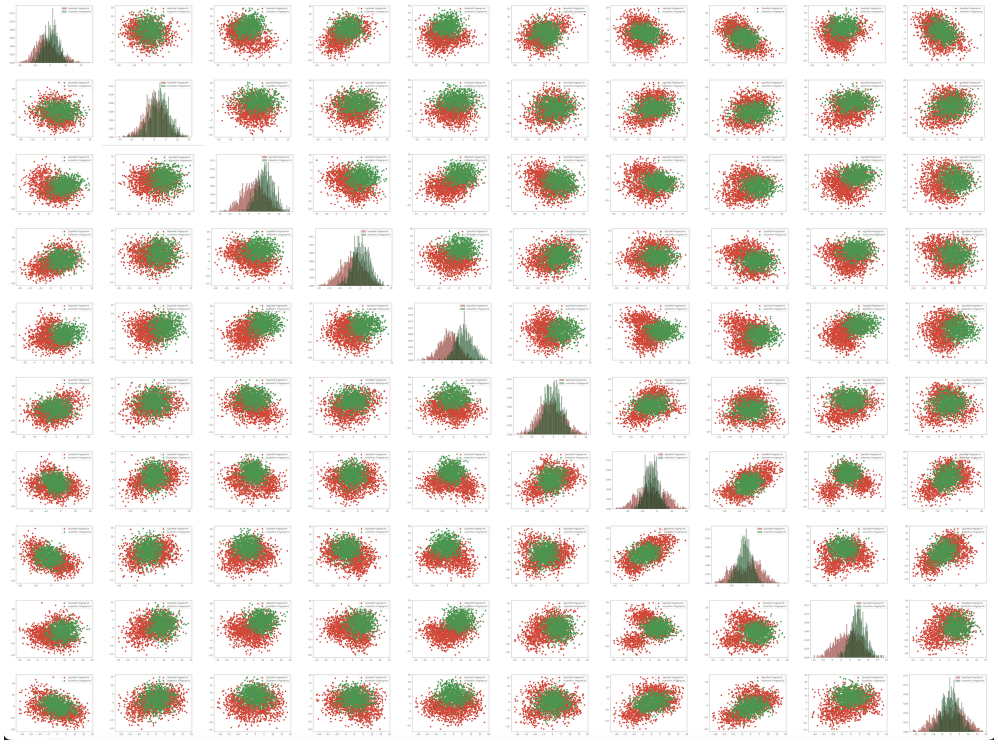

Figure 1.1: Feature Plot

We can see that the features seem to be distributed quite well as a gaussian density without evident outliers; for this reason we wont apply gaussianization preprocessing method, eventually we will only use Z-normalization technique.
In the next step we plott the heatmap of features correlation.

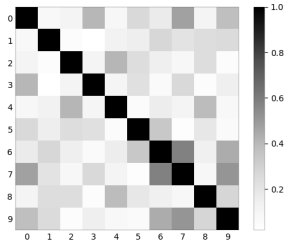

Figure 1.2: All classes

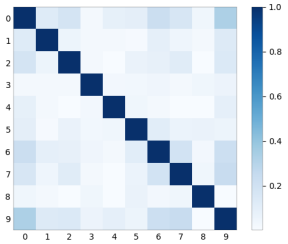

Figure 1.3: Authentic fingerprint

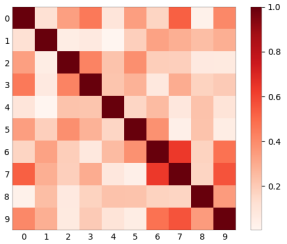

Figure 1.4: Spoofed fingerprint

Figure 1.5: Heatmap - Pearson Correlation

We can observe that generally the features aren't correlated, in particular the authentic ones. On the other hand, in the spoofed samples we can see that features 0-7, 6-7, 6-9 and 7-9 are slightly correlated.

Chapter 2

Classification

2.1 Introduction

Our application's performances will be evaluated in terms of minimum costs, in order to select the most promising model we have to compute the minDCF for the given working point ($\pi_T = 0.5, C_{fn} = 1, C_{fp} = 10$). This working point has a high C_{fp} since classifing a *spoofed* fingerprint as *authentic* can result in security vulnerabilities. Since the training set is small compared to the test one we adopt a K-Fold protocol with K = 5 in order to get more accurate results. We will also try different priors ($\pi_T = 0.1$, $\pi_T = 0.9$).
As a pre-processing method we will use PCA in order to verify if it can improve our results by mapping samples from a 10^{th} dimensional space to a lower one (we will check m = 9, m = 8, m = 7, m = 6).

2.2 Multivariate Gaussian Model

Guassian Classifier models belongs to Generative classifier class, a sample is assigned to a class by the optimal bayes decision rule, where the label corresponds to the max of the posterior probabilities.

$$c^* = argmax P(C = c_t|x) \tag{2.1}$$

There are different kind of Mutivariate Gaussian models, the ML paramenters that they use are $\mu_c \quad \Sigma_c$ which are the mean and the covariance matrix. But every model has different way of computing those parameters:

- **MVG Full** computes $\mu_c, \quad \Sigma_c$ for every class
- **MVG Naive Bayes** computes $\mu_c, \quad \Sigma_c$ for every class but the covariance matrices are diagonalized
- **MVG Tied** computes $\mu_c, \quad \Sigma_c$ but the covariance matrix is unique for all the classes
- **MVG Tied Naive Bayes** computes $\mu_c, \quad \Sigma_c$ but the covariance matrix is unique for all the classes and diagonalized

All the previous models have different advantages and disadvantages based on the properties of the dataset. We will analyze each aspect afterwards. It is possible to use the same method, for all the Gaussian models, to assign a label for a binary problem like this one. It's called log-lokelihood ratio that produce the score with probabilistic interpretations where higher scores represent preferences for class h_1

$$logr(x) = \frac{P(C = h_1|x)}{P(C = h_0|x)} = log\frac{f_{X|C}(x|h_1)}{f_{X|C}(x|h_0)} + log\frac{P(h1)}{P(h0)} \tag{2.2}$$

thus the optimal solution is given by:

$$log\frac{f_{X|C}(x|h_1)}{f_{X|C}(x|h_0)} \lessgtr -log\frac{\pi}{1-\pi} \tag{2.3}$$

By looking at features correlations and covariances matrices plots we can hypothesize that MVG Full model and MVG Naive Model will outperform the MVG Tied and MVG Naive Tied since those are penalized when features and class have different distributions.

MVG Classifier			
RAW			
PCA	minDCF ($\tilde{\pi} = 0.5$)	minDCF ($\tilde{\pi} = 0.9$)	minDCF ($\tilde{\pi} = 0.1$)
-	0.331	0.111	0.616
9	0.338	0.110	0.629
8	0.329	0.113	0.626
7	0.336	0.113	0.613

Naive Gaussian Classifier			
RAW			
PCA	minDCF ($\tilde{\pi} = 0.5$)	minDCF ($\tilde{\pi} = 0.9$)	minDCF ($\tilde{\pi} = 0.1$)
-	0.472	0.144	0.806
9	0.381	0.115	0.726
8	0.391	0.116	0.732
7	0.388	0.115	0.737

Tied MVG Classifier			
RAW			
PCA	minDCF ($\tilde{\pi} = 0.5$)	minDCF ($\tilde{\pi} = 0.9$)	minDCF ($\tilde{\pi} = 0.1$)
-	0.486	0.184	0.706
9	0.472	0.183	0.700
8	0.476	0.183	0.691
7	0.475	0.181	0.695

Tied Naive Gaussian Classifier			
RAW			
PCA	minDCF ($\tilde{\pi} = 0.5$)	minDCF ($\tilde{\pi} = 0.9$)	minDCF ($\tilde{\pi} = 0.1$)
-	0.551	0.198	0.790
9	0.569	0.201	0.823
8	0.568	0.200	0.815
7	0.572	0.200	0.812

Considerations

As we assumed before the MVG Full model and Naive model have outperformed the other two. Both models scores the best results by applying the PCA pre-processing technique. In particular the MVG Full appears to be the best performing, by using PCA-8 with a score of 0.329.

2.3 Logistic Regression

The Logistic Regression uses a discriminative approach that doesn't model the distribution of the observed samples but directly models the post distribution for the classes. We used two kind of model, Prior-weigth Logistic regression thath produce linear decision boundaries, we also tested the Quadratic Logistic regression that allows us to have non linear boundaries by expanding the feature space. The scores are computed by:

$$s_i = w^T x + b \tag{2.4}$$

This scores are related with the distance of the sample from the boundaries. We have to consider different working points (i.e. with different priors) hence we will use a regularized prior-weighted version of the objective function:

$$R(w,b) = \frac{\lambda}{2}||w||^2 + \frac{\pi_T}{n_T}\sum_{i=1|c_i=1}^{n} log(1+e^{-z_i(w^T x_i+b)}) + \frac{1-\pi_T}{n_F}\sum_{i=1|c_i=0}^{n} log(1+e^{-z_i(w^T x_i+b)}) \tag{2.5}$$

where (w, b) are the model parameters; $\frac{\lambda}{2}||w||^2$ is a regularization term useful to obtain a $\mathbf{w}$ with a lower norm (which reduce the risk of over-fitting the training data).
Finally λ is an hyperparameter called regularization coefficient:

- $\lambda \gg 0$: the norm of $\mathbf{w}$ ($||w||^2$) is reduced there is poor separation of classes.
- $\lambda \approx 0$: poor generalization of unseen data (low accuracy) but good separation of classes on the training set.

Given the previous information about the peculiarity of each model we can forecast that the Prior-Weighted Linear Logistic Regression (called Logistic Regression) will perform poorly since the classes cannot be separated linearly in an easy way. On the other hand the Quadratic Logistic Regression should perform well thanks to the extended feature space that provides non linear separations. Also Quadratic Logistic Regression will be done with Prior-Weighted solution.

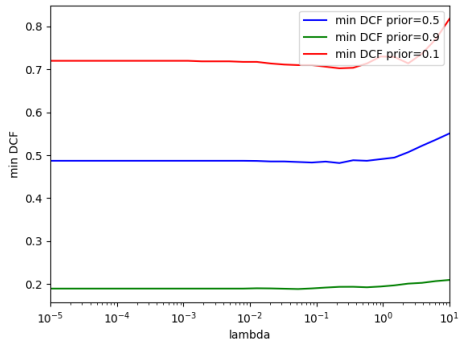

Figure 2.1: Raw Data

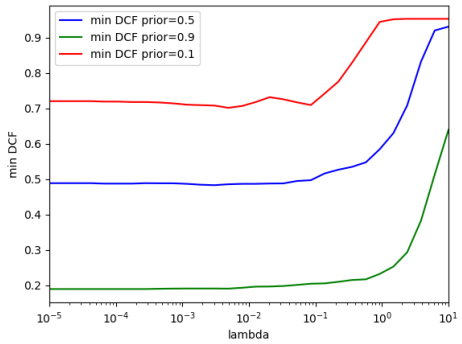

Figure 2.2: Z-Norm

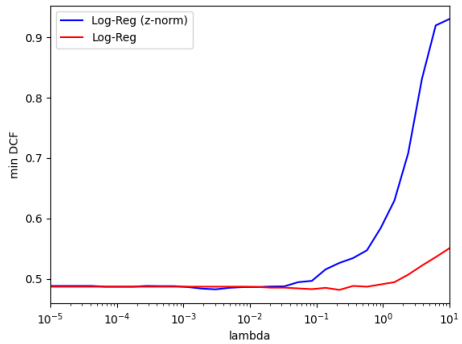

Figure 2.3: RAW vs Z-Norm Plot

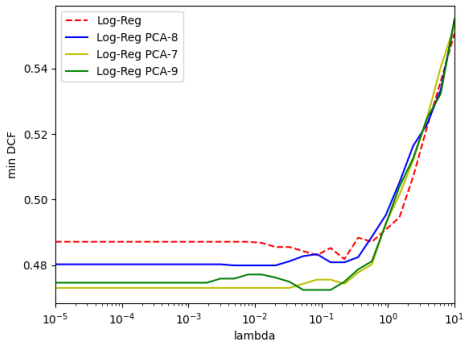

Figure 2.4: PCA comparison Plot

The choice of λ depends on the values of minDCF.
Seeing the plot in Figure 2.1, a good choice for our working point could be $\lambda = 10^{-1}$. The result with PCA-7 produce better score, moreover applying normalization does not seem particularly effective either.

Logistic Regression Classifier - $\lambda = 10^{-1}$						
	RAW			*Z-Normalizzation*		
PCA	minDCF ($\tilde{\pi} = 0.5$)	minDCF ($\tilde{\pi} = 0.9$)	minDCF ($\tilde{\pi} = 0.1$)	minDCF ($\tilde{\pi} = 0.5$)	minDCF ($\tilde{\pi} = 0.9$)	minDCF ($\tilde{\pi} = 0.1$)
-	0.483	0.191	0.711	0.503	0.205	0.720
9	0.472	0.193	0.700	0.496	0.188	0.759
8	0.481	0.191	0.694	0.500	0.188	0.765
7	0.476	0.191	0.696	0.495	0.187	0.758

2.3.1 Quadratic Logistic Regression

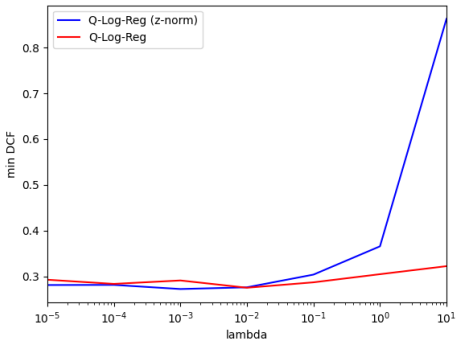

Figure 2.5: RAW vs Z-Norm Plot

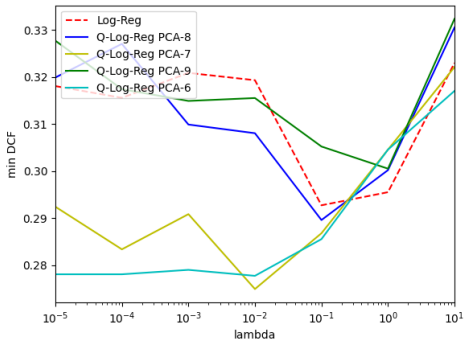

Figure 2.6: PCA comparison Plot

The choice of λ depends on the values of minDCF.
Seeing the plot in Figure 2.5, a good choice for our working point could be $\lambda = 10^{-2}$. The result is heavily influenced by the use of PCA so we choose the model with PCA-7 as shown in Figure 2.6, moreover applying normalization does not seem particularly effective.

Quadratic Logistic Regression Classifier - $\lambda = 10^{-2}$						
	RAW			*Z-Normalizzation*		
PCA	minDCF ($\tilde{\pi} = 0.5$)	minDCF ($\tilde{\pi} = 0.9$)	minDCF ($\tilde{\pi} = 0.1$)	minDCF ($\tilde{\pi} = 0.5$)	minDCF ($\tilde{\pi} = 0.9$)	minDCF ($\tilde{\pi} = 0.1$)
7	0.275	0.100	0.495	0.276	0.098	0.512

Considerations

The hypothesis on the models are correct, since the Prior-Weghted Linear Logistic Regression didn't performed well, instead the Quadratic Logistic Regression seems to be best performing model until now with a score of 0.275.

2.4 Support Vector Machine

SVM Linear

The risk minimization problem seen in Logistic Regression can be generalized, allowing the separation of the samples up to a margin. This strategy is the Support Vector Machine (SVM) and to solve the SVM problem we can consider the dual formulation instead of the primal; indeed the dual is easier to optimize because its complexity depends only on the number of samples and we can compute non-linear hyperplanes without having to expand the features:

$$J^D(\alpha) = -\frac{1}{2}\alpha^T H\alpha + \alpha^T \mathbf{1} \tag{2.6}$$

where $0 \leq \alpha_i \leq C$, $\forall i \in \{1, ..., n\}$ and $\sum_{i=1}^{n} \alpha_i z_i = 0$.
In adding, there is also a balanced version of SVM where different values of C is used for each class in the box constraint of the dual formulation:

$$0 \leq \alpha_i \leq C, \forall i \in \{1, ..., n\}$$
$$where \quad Ci = \begin{cases} C\frac{\pi_T}{\pi_T^{emp}} if i \in h_1 \\ C\frac{\pi_F}{\pi_F^{emp}} if i \in h_0 \end{cases} \tag{2.7}$$

We will study the first model which is SVM Linear: our classes can't be divided in an optimal way by linear model so we expect SVM Linear to perform worse than the non-linear one. Anyway in the table below there are some combinations of parameters C and K for RAW data:

SVM Linear $\tilde{\pi} = 0.5$			
	K = 0.1	K = 1	K = 10
C = 0.01	0.919	0.532	0.473
C = 0.1	0.780	0.479	0.479
C = 1	0.533	0.468	0.475
C = 10	0.472	0.464	0.473
$\tilde{\pi} = 0.9$			
C = 0.01	0.379	0.216	0.187
C = 0.1	0.319	0.186	0.188
C = 1	0.208	0.184	0.188
C = 10	0.200	0.189	0.236
$\tilde{\pi} = 0.1$			
C = 0.01	0.990	0.680	0.716
C = 0.1	0.964	0.670	0.713
C = 1	0.685	0.720	0.717
C = 10	0.669	0.719	0.717

We can see that the best combination is (C=10, K=1), so we'll choose K=1 and now we can follow plotting minDCF for SVM Linear case, in a first analisys we compare unbalanced vs balanced SVM:

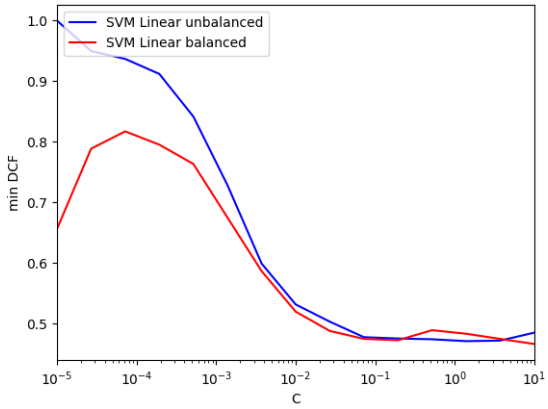

Figure 2.7: unbalanced vs balanced

Even if the balanced version performs better for low values of C, we can see that the best result is for C=10 in the unbalanced version so we will use this model and this value for C. Now we will study PCA variants of the SVM Linear classifier also with RAW and Z-normalized data:

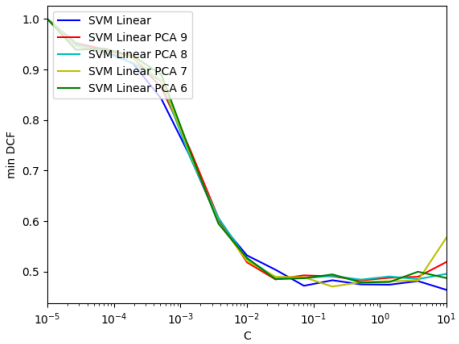

Figure 2.8: RAW vs PCA Plot

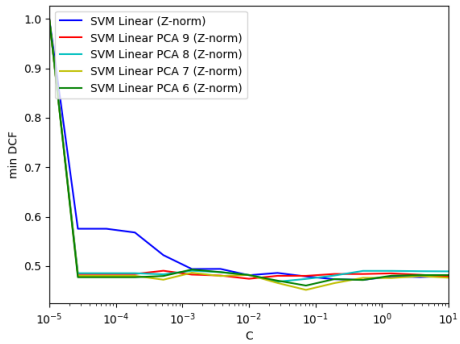

Figure 2.9: Z-norm comparison Plot

With respect to using PCA, we can see that with or without PCA it doesn't affect so much the result so we won't use it. Also using Z-normalization isn't particularly effective so we will use RAW data on future analisys.

2.4.1 SVMs Quadratic

The dual formulation depends on the samples through dot products:

$$H = z_i z_j x_i^T x_j \tag{2.8}$$

Thanks to this, we haven't to explicitly extend the features in the expanded spaces but only to compute dot products. We need a function able to compute efficiently dot products in the expanded spaces:

$$k(x_1, x_2) = \Phi(x_1)^T \Phi(x_2) \tag{2.9}$$

k is called kernel function and with it both training and scoring can be performed. This will allow us to compute a linear separation surface in the expanded space which corresponds to a non-linear separation surface in the original one. In particular we will use two different type of kernel functions:

- **Polynomial**: $(x_i^T x_j + c)^d$
- **Radial Basis Function (RBF)**: $e^{-\gamma ||x_i - x_j||^2}$

SVM Polynomial

As we have done for SVM Linear we will select the best value for K seeing the best result in the table for RAW data:

SVM Polynomial $\tilde{\pi} = 0.5$			
	K = 0.1	K = 1	K = 10
C = 0.01	0.331	0.341	0.330
C = 0.1	0.356	0.341	0.324
C = 1	0.546	0.476	0.506
C = 10	0.954	0.994	0.935
$\tilde{\pi} = 0.9$			
C = 0.01	0.123	0.122	0.108
C = 0.1	0.122	0.117	0.119
C = 1	0.212	0.240	0.202
C = 10	0.774	0.790	0.589
$\tilde{\pi} = 0.1$			
C = 0.01	0.701	0.717	0.689
C = 0.1	0.664	0.667	0.679
C = 1	0.827	0.875	0.681
C = 10	0.964	0.994	0.935

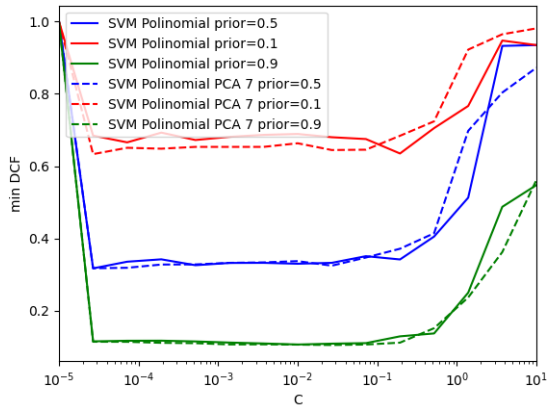

Figure 2.10: SVM Polynomial different prior

As we can see the lowest value is obtained by the combination (C=0.1, K=10), so thanks to this we can plot the minDCF graphs (we have also choose costant = 0 and degree = 2), also as we hypotized using PCA doesn't change the estimation:

SVM RBF

In the end we study SVM RBF choosing the parameters from the table to check different combinations for RAW data:

SVM RBF $\tilde{\pi} = 0.5$			
	K = 0.1	K = 1	K = 10
C = 0.01	0.441	0.413	0.395
C = 0.1	0.349	0.332	0.336
C = 1	0.302	0.300	0.303
C = 10	0.293	0.298	0.302
$\tilde{\pi} = 0.9$			
C = 0.01	0.171	0.146	0.142
C = 0.1	0.124	0.122	0.121
C = 1	0.097	0.100	0.101
C = 10	0.089	0.089	0.096
$\tilde{\pi} = 0.1$			
C = 0.01	0.669	0.664	0.708
C = 0.1	0.607	0.613	0.649
C = 1	0.593	0.575	0.575
C = 10	0.580	0.599	0.601

After that we choose K=0.1 and $log(\gamma) = -3$; we plot minDCF:

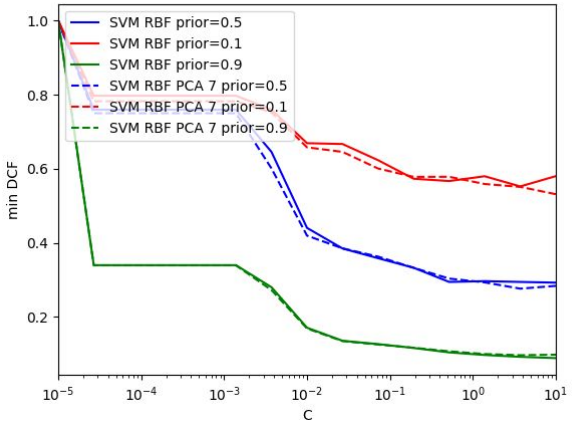

Figure 2.11: SVM RBF different prior

To strenghten our inizial assumptions we can see in detail if rebalancing or z-norm improve the value:

SVM RBF $\tilde{\pi} = 0.5$			
	K = 0.1	K = 1	K = 10
C = 10	0.293	0.298	0.302
SVM RBF Balanced $\tilde{\pi} = 0.5$			
C = 10	0.295	0.303	306
SVM RBF Z-norm $\tilde{\pi} = 0.5$			
C = 10	0.465	0.451	0.462
SVM RBF PCA 9 $\tilde{\pi} = 0.5$			
C = 10	0.292	0.297	0.298
SVM RBF PCA 8 $\tilde{\pi} = 0.5$			
C = 10	0.285	0.288	0.290
SVM RBF PCA 7 $\tilde{\pi} = 0.5$			
C = 10	0.284	0.288	0.285

Re-balancing doesn't seem effective for SVM in this task, also z-norm has similar behaviour, the best score is obtained by SVM RBF with PCA 7.

Considerations

In the previous plots we can compare the three different model:
Although for low values of C, SVM Polynomial is better than the other two model, we can see that the best result on this dataset is obtained by SVM RBF with C=10. Plus, the SVM Linear performs poorly for every value of C. These results confirm what we hypothesized at the beginning: linear model aren't good for this case study while quadratic models perform better, in particular SVM RBF with PCA 7 outperforms the other two models with a score of 0.284.

2.5 Gaussian Mixture Model

Gaussian Mixture Models belong to the Generative model group, they are useful when it's not possible to represent class distribution with a single Gaussian distribution. Gaussian Mixture can be generalized to any distribuition with a certain level of precision. They allow to partition the training set in subsets that can be described as Gaussian density. Since target class is well described by Gaussian density and non-target class is divided in 6-subsets classes we consider models with different components for target and non-target class, with less components for target in order to get better results.

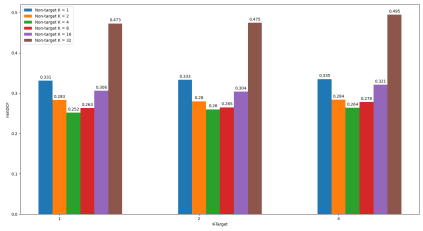

Figure 2.12: GMM Full

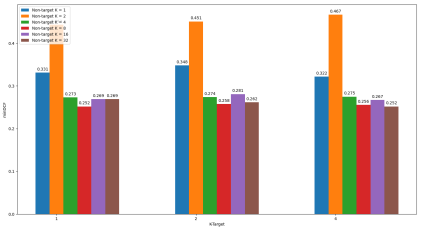

Figure 2.13: GMM Naive

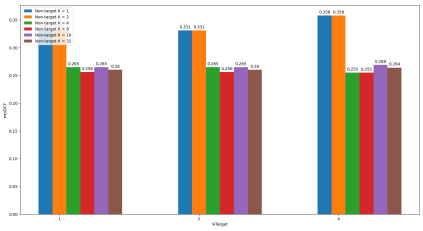

Figure 2.14: GMM Tied

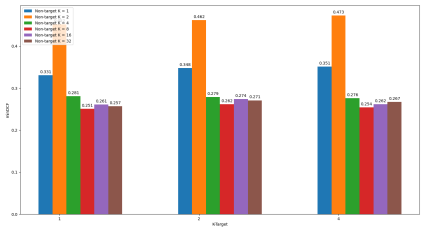

Figure 2.15: GMM Naive Tied

From the figures above it is possible to choose the best combination for the number of target component and non target one. We further analyze every model by appling PCA.

GMM FULL - 1 T 4 NT			
RAW			
PCA	minDCF ($\tilde{\pi} = 0.5$)	minDCF ($\tilde{\pi} = 0.9$)	minDCF ($\tilde{\pi} = 0.1$)
-	0.252	0.085	0.471
9	0.261	0.089	0.475
8	0.260	0.084	0.477
7	0.252	0.083	0.489

GMM NAIVE - 1 T 8 NT			
RAW			
PCA	minDCF ($\tilde{\pi} = 0.5$)	minDCF ($\tilde{\pi} = 0.9$)	minDCF ($\tilde{\pi} = 0.1$)
-	0.250	0.083	0.547
9	0.264	0.085	0.565
8	0.270	0.081	0.559
7	0.262	0.083	0.554

GMM TIED - 1 T 8 NT			
RAW			
PCA	minDCF ($\tilde{\pi} = 0.5$)	minDCF ($\tilde{\pi} = 0.9$)	minDCF ($\tilde{\pi} = 0.1$)
-	0.255	0.083	0.530
9	0.252	0.087	0.536
8	0.261	0.080	0.531
7	0.254	0.082	0.510

GMM TIED NAIVE - 1 T 8 NT			
RAW			
PCA	minDCF ($\tilde{\pi} = 0.5$)	minDCF ($\tilde{\pi} = 0.9$)	minDCF ($\tilde{\pi} = 0.1$)
-	0.251	0.081	0.519
9	0.258	0.083	0.539
8	0.260	0.082	0.537
7	0.252	0.084	0.544

Considerations

The results met our initial prediction, in fact GMM outperforms MVG models because each component is represent with a better Gaussian distribution, and the most promising model is the GMM Naive since by looking at the features and classes distributions this model can get better results by splitting the training set with 1 component for target and 8 for non target.

Chapter 3

Scores calibration

3.1 Scores calibration

The only metric used to validate models' performances until now was the minDCF. But the actual cost we have to pay for the class assignment depends on the goodness of the threshold applied. While the minDCF is computed by using the scores as the threshold in a dynamic way, the actDCF is calculated using a static criteria:

$$t = -log\frac{\pi_T}{1-\pi_T} \tag{3.1}$$

The difference between the minDCF and actDCF is meant to be 0 in an optimal case. However, if there is a gap, it is due to **miscalibrated scores** and, to have more accurate analisys, calibration is needed.
We can now proceed to compute the error Bayes plot for the following models:

- MVG - PCA 8
- Q-Log-Reg with $\lambda = 0.01$ PCA - 7
- SVM RBF with $\gamma = 0.001, \quad K = 0.1, \quad C = 10$ and PCA 7
- GMM 1C - 8C

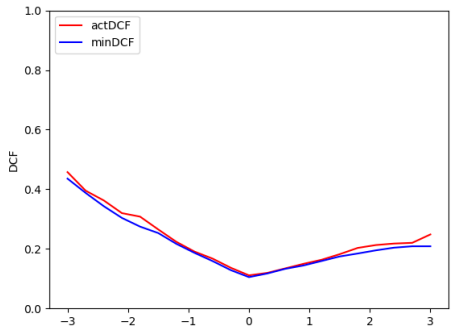

Figure 3.1: min vs act MVG

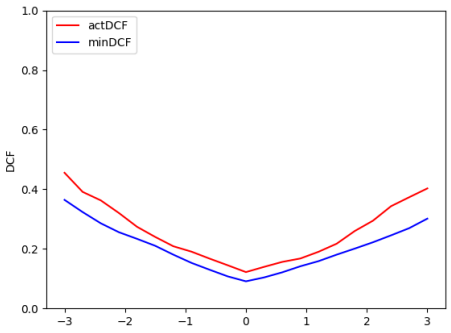

Figure 3.2: min vs act Quadratic LogReg

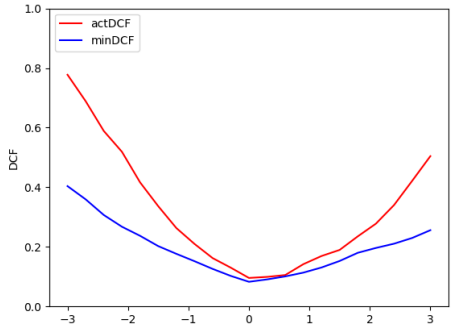

Figure 3.3: min vs act SVM RBF

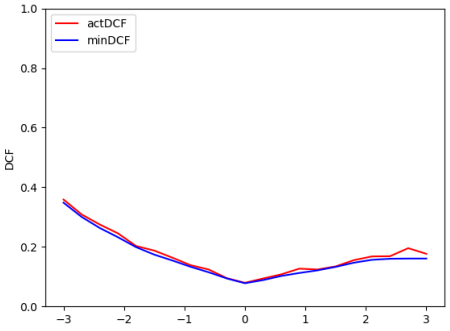

Figure 3.4: min vs act GMM

As we expected the actDCF for MVG and GMM models seems to be pretty similar to the minDCF and it doesn't require to be calibrated. On the other hand, the actDCF for quadratic LogReg and SVM RBF models has a larger gap and a score calibration technique is required, in particular SVM RBF is the model with the higher miss-calibration. The miscalibration is due to the fact that non probabilistic model generally tends to produce uncalibrated scores.
Score calibration will allow us to compute a transformation function which map scores $\mathbf{s}$ from uncalibrated to calibrated ones s_c. We employ prior-weighted Log-Reg to re-calibrate all the models, in particular we pass the scores to the LogReq weighted in order to obtain the values of α and β. Also this process is done through K-fold.
If we assume this function as a linear function in s we can write:

$$f(s) = \alpha s + \beta \tag{3.2}$$

$f(s)$ can be view as a log-likelihood ratio for the two class.; we can rewrite the class posterior for a prior $\tilde{\pi}$:

$$log\frac{P(C = h_1|s)}{P(C = h_0|s)} = \alpha s + \beta + log\frac{\tilde{\pi}}{1-\tilde{\pi}} \tag{3.3}$$

The log posterior ratio of the Logistic Regression can be obtained by some modification to the f(s) function: first of all s is interpreted as features and we write $\beta^{'} = \beta + log\frac{\tilde{\pi}}{1-\tilde{\pi}}$. After this, we only need to train the model to learn the value of α and $\beta^{'}$. So the final equation for calibrating the score will be:

$$f(s) = \alpha s + \beta - log\frac{\tilde{\pi}}{1-\tilde{\pi}} \tag{3.4}$$

We select $\tilde{\pi} = 0.7$ since our dataset is unbalanced, and we choose $log(\lambda) = -4$ As we can see on the following plots, we were able to well-calibrate the scores; below we can see some parallelism minDCF/actDCF in detail:

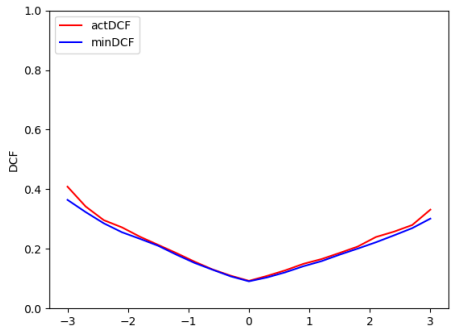

Figure 3.5: Calibration Quadratic LogReg

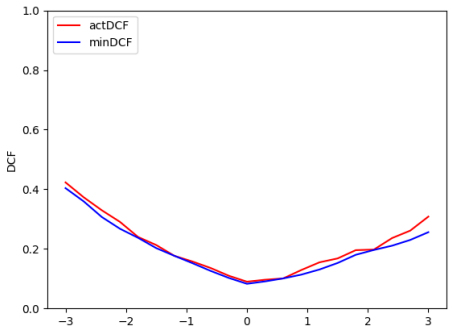

Figure 3.6: Calibration SVM RBF

Employing score calibration we were able to calibrate the scores reducing the gap between minDCF and actDCF for Quadratic LogReg and SVM RBF classifiers.

Chapter 4

Wrap up

4.1 Wrap up

In this part from each category we picked the best preforming one.

	minDCF ($\tilde{\pi} = 0.5$)	minDCF ($\tilde{\pi} = 0.9$)	minDCF ($\tilde{\pi} = 0.1$)
MVG Full $PCA = 8$	0.329	0.113	0.626
Quadratic-Log-Reg $\lambda = 0.01$ $PCA = 7$	0.275	0.100	0.495
RBF SVM $PCA = 7$ $log\gamma = -3$ $C = 10$	0.284	0.089	0.580
GMM NAIVE T = 1C NT = 8C	0.250	0.083	0.547

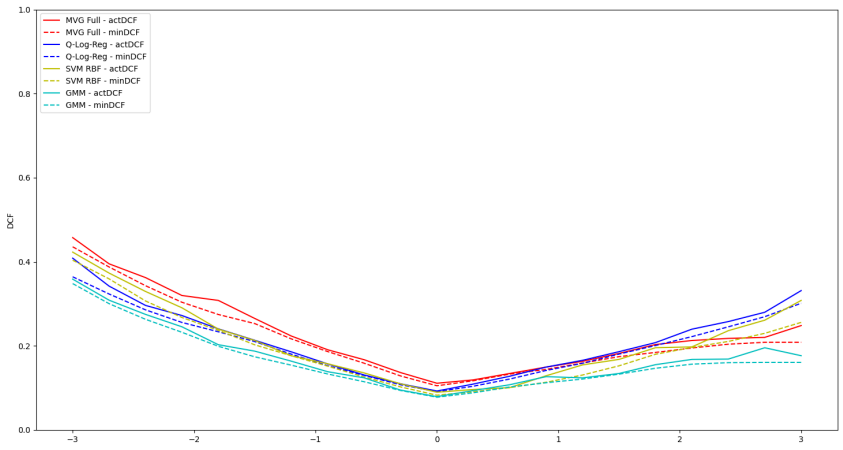

Figure 4.1: Model comaparis by actual and min DCF

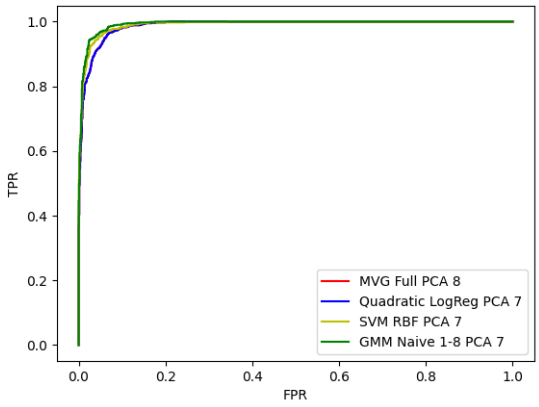

Figure 4.2: ROC curve plot

In the pictures above we can see that all the models are almost well calibrated after calibration, in fact calibration loss has been significantly reduced for SVM RBF and Q-Log models. As the pictures show, GMM model proves to be the best among the others since it has the lowest DCF.

Final pick

Given the previous analisys is safe to say that the best performing model over the given dataset is the **GMM Naive** model with 1 component for target class and 8 components for non target class. So we expect this model to outperform the others in evaluation part.

Chapter 5

Evaluation

5.1 Introduction

We will now focus on evaluation part. For the validation we used k-fold approach but for evaluation we will use the whole test set. Plus, we will use minDCF as the metric; our goal is to verify if the assumptions made on the chosen models in the validation part are met.

5.2 Multivariate Gaussian Model

MVG Classifier			
RAW			
PCA	minDCF ($\tilde{\pi} = 0.5$)	minDCF ($\tilde{\pi} = 0.9$)	minDCF ($\tilde{\pi} = 0.1$)
-	0.276	0.083	0.545
9	0.272	0.082	0.569
8	0.271	0.083	0.571
7	0.273	0.084	0.581

Naive Gaussian Classifier			
RAW			
PCA	minDCF ($\tilde{\pi} = 0.5$)	minDCF ($\tilde{\pi} = 0.9$)	minDCF ($\tilde{\pi} = 0.1$)
-	0.352	0.111	0.718
9	0.310	0.087	0.635
8	0.315	0.087	0.638
7	0.313	0.087	0.632

Tied MVG Classifier			
RAW			
PCA	minDCF ($\tilde{\pi} = 0.5$)	minDCF ($\tilde{\pi} = 0.9$)	minDCF ($\tilde{\pi} = 0.1$)
-	0.455	0.176	0.789
9	0.461	0.177	0.783
8	0.456	0.178	0.773
7	0.459	0.178	0.776

Tied Naive Gaussian Classifier			
RAW			
PCA	minDCF ($\tilde{\pi} = 0.5$)	minDCF ($\tilde{\pi} = 0.9$)	minDCF ($\tilde{\pi} = 0.1$)
-	0.480	0.177	0.829
9	0.483	0.175	0.864
8	0.484	0.176	0.865
7	0.482	0.174	0.857

Considerations

The obtained result are aligned with the ones in validation. The best model proved to be the MVG Full covariance and by looking at the Tied models the linear rules again are not suitable for this application. Both for validation and evaluation PCA pre processing technique seems effective. Both parts uses PCA-8 that returns the best performances.

5.3 Logistic Regression

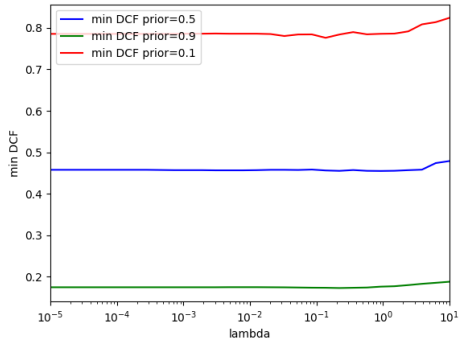

Figure 5.1: Raw Data

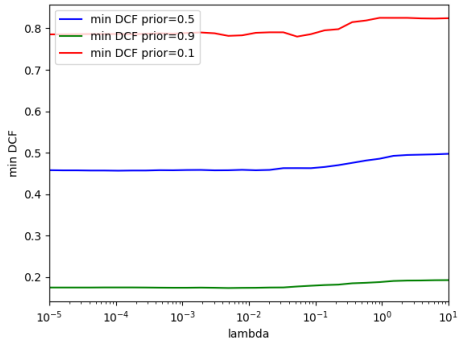

Figure 5.2: Z-Norm

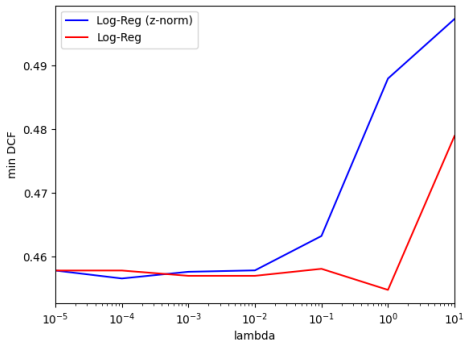

Figure 5.3: Raw vs Z-norm

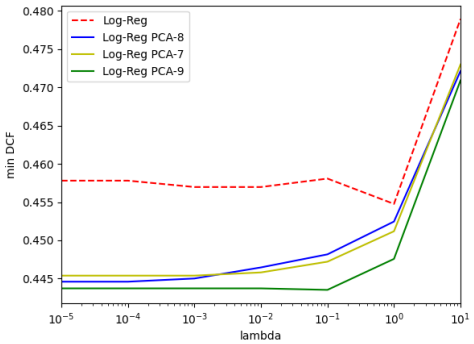

Figure 5.4: Z-Norm

Logistic Regression Classifier - $\lambda = 10^{-1}$						
	RAW			*Z-Normalizzation*		
PCA	minDCF ($\tilde{\pi} = 0.5$)	minDCF ($\tilde{\pi} = 0.9$)	minDCF ($\tilde{\pi} = 0.1$)	minDCF ($\tilde{\pi} = 0.5$)	minDCF ($\tilde{\pi} = 0.9$)	minDCF ($\tilde{\pi} = 0.1$)
-	0.458	0.173	0.782	0.463	0.179	0.794
9	0.444	0.173	0.791	0.460	0.172	0.790
8	0.448	0.173	0.786	0.460	0.173	0.782
7	0.447	0.172	0.792	0.459	0.174	0.782

Considerations

As expected Prior-Weighted Linear Logistic Regression performs poorly, showing again how linear decision boundaries aren't a good choice for this application. Despite that, results are in line with ones got from the validation part, PCA-9 seems to get the best out of the this Logistic model with $\lambda = 10^{-1}$

5.3.1 Quadratic Logistic Regression

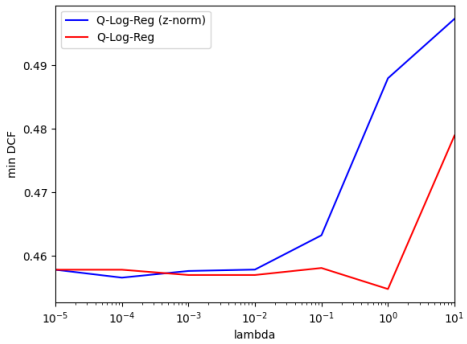

Figure 5.5: Raw vs Z-norm

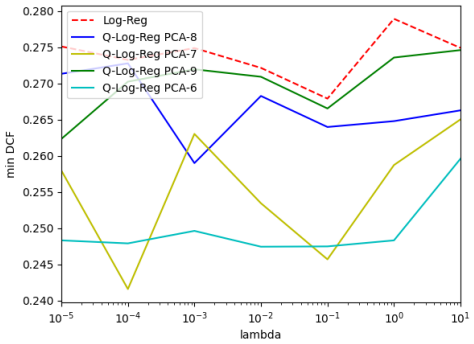

Figure 5.6: Z-Norm

Quadratic Logistic Regression Classifier - $\lambda = 10^{-4}$						
	RAW			*Z-Normalizzation*		
PCA	minDCF ($\tilde{\pi} = 0.5$)	minDCF ($\tilde{\pi} = 0.9$)	minDCF ($\tilde{\pi} = 0.1$)	minDCF ($\tilde{\pi} = 0.5$)	minDCF ($\tilde{\pi} = 0.9$)	minDCF ($\tilde{\pi} = 0.1$)
7	0.242	0.082	0.567	0.252	0.080	0.526

Considerations

Results for Quadratic-Logistic Regression are consistent with our expectations. In fact by providing a non-linear separation boundaries this model is able to outperform the Linear Logistic Regression and also to be the most promising one so far among the others. Both for validation and evaluation PCA 7 is effective, but the best hyperparameters for evaluation is $\lambda = 10^{-4}$ rather than $\lambda = 0.01$ for validation part, however this is the only small difference.

5.4 Support Vector Machine

5.4.1 SVM Linear

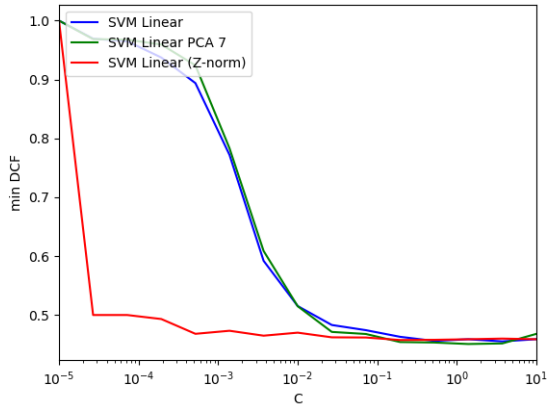

Figure 5.7: SVM Linear

As we can see we obtain a similar result also in the evaluation part, in the table below we will see in detail using C=10 and K=1. The best result is in this case with PCA 8 and not anymore without PCA

SVM Linear			
RAW			
PCA	minDCF ($\tilde{\pi} = 0.5$)	minDCF ($\tilde{\pi} = 0.9$)	minDCF ($\tilde{\pi} = 0.1$)
-	0.459	0.176	0.794
9	0.470	0.183	0.789
8	0.456	0.178	0.780
7	0.468	0.178	0.769

5.4.2 SVM Polynomial

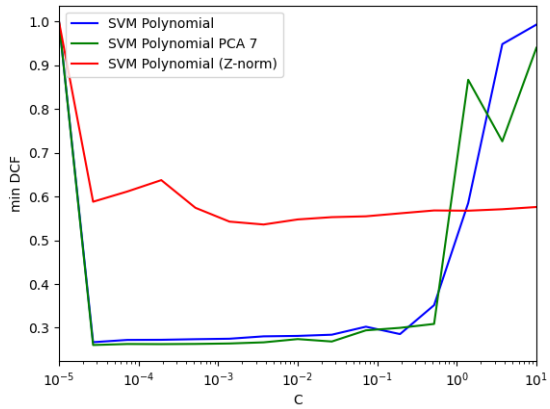

Figure 5.8: SVM Polinomial

Also for the SVM Polynomial classifier we obtain a similar behaviour than in the training set, even if PCA 7 is now slightly better.

SVM Polynomial			
RAW			
PCA	minDCF ($\tilde{\pi} = 0.5$)	minDCF ($\tilde{\pi} = 0.9$)	minDCF ($\tilde{\pi} = 0.1$)
-	0.288	0.102	0.567
9	0.289	0.093	0.610
8	0.289	0.104	0.607
7	0.263	0.110	0.550

5.4.3 SVM RBF

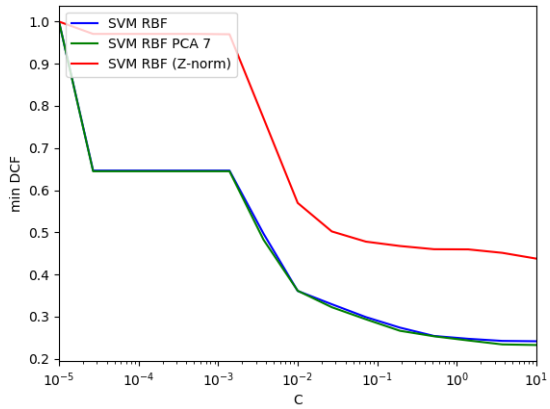

Figure 5.9: SVM RBF

Finally we can see the results for SVM RBF and also here we don't have unexpected results. Indeed, as for SVM Polynomial, minDCF with PCA 7 perform better, we confirm the result we obtained in the validation part:

SVM RBF			
RAW			
PCA	minDCF ($\tilde{\pi} = 0.5$)	minDCF ($\tilde{\pi} = 0.9$)	minDCF ($\tilde{\pi} = 0.1$)
-	0.241	0.086	0.519
9	0.242	0.085	0.490
8	0.242	0.085	0.492
7	0.234	0.085	0.477

Considerations

Overall also in testing set the best model is SVM RBF as we first hypothesized.

5.5 Gaussian Mixture Models

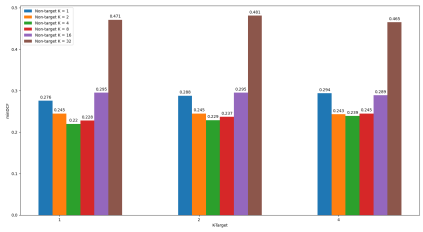

Figure 5.10: GMM Full

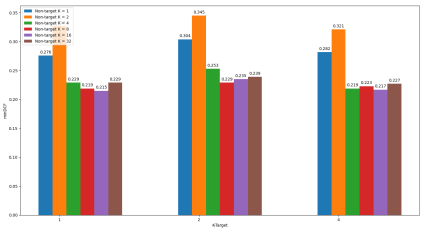

Figure 5.11: GMM Naive

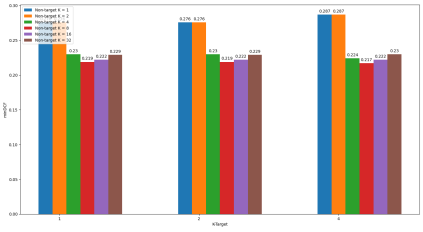

Figure 5.12: GMM Tied

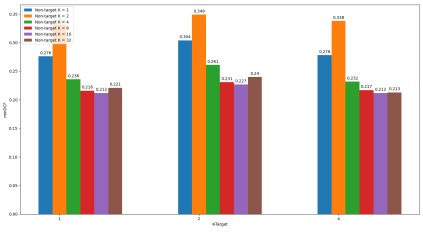

Figure 5.13: GMM Naive Tied

GMM FULL - 1 T 4 NT			
RAW			
PCA	minDCF ($\tilde{\pi} = 0.5$)	minDCF ($\tilde{\pi} = 0.9$)	minDCF ($\tilde{\pi} = 0.1$)
-	0.220	0.076	0.473
9	0.219	0.075	0.460
8	0.218	0.074	0.460
7	0.217	0.075	0.451

GMM NAIVE - 1 T 16 NT			
RAW			
PCA	minDCF ($\tilde{\pi} = 0.5$)	minDCF ($\tilde{\pi} = 0.9$)	minDCF ($\tilde{\pi} = 0.1$)
-	0.215	0.072	0.459
9	0.217	0.075	0.515
8	0.206	0.071	0.475
7	0.201	0.071	0.415

GMM TIED - 4 T 8 NT			
RAW			
PCA	minDCF ($\tilde{\pi} = 0.5$)	minDCF ($\tilde{\pi} = 0.9$)	minDCF ($\tilde{\pi} = 0.1$)
-	0.217	0.073	0.475
9	0213	0.074	0.442
8	0.216	0.075	0.438
7	0.211	0.073	0.444

GMM TIED NAIVE - 1 T 8 NT			
RAW			
PCA	minDCF ($\tilde{\pi} = 0.5$)	minDCF ($\tilde{\pi} = 0.9$)	minDCF ($\tilde{\pi} = 0.1$)
-	0.212	0.071	0.519
9	0.219	0.071	0.539
8	0.218	0.071	0.537
7	0.220	0.072	0.544

Considerations

The GMM results seems pretty consistent with the expected ones. The GMM Naive model outperforms the others, although in this case we applied a different number of components from the one used in the validation, for target class. This changes proves to be efficient since in this way Naive is still the best model among the others.

Chapter 6

Scores calibration

6.1 Scores calibration

As we did for validation some models may produce uncalibrated scores, calibration of the scores is needed in order to choose the best performing model in a fair way.

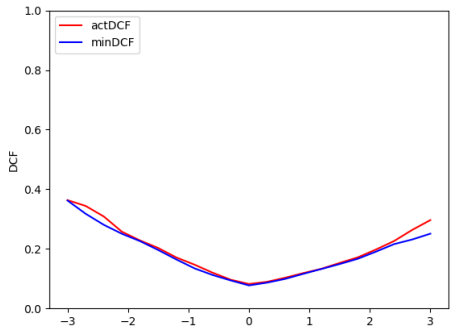

Figure 6.1: min vs act MVG

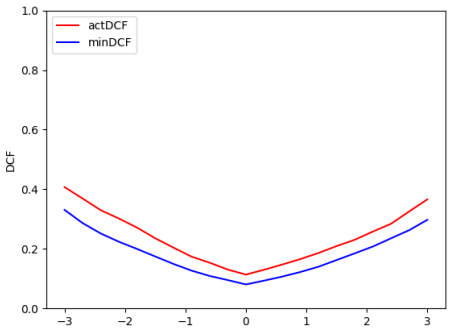

Figure 6.2: min vs act Quadratic LogReg

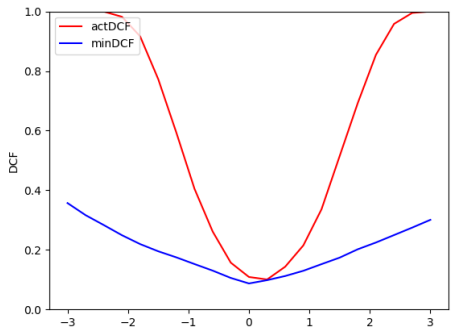

Figure 6.3: min vs act SVM RBF

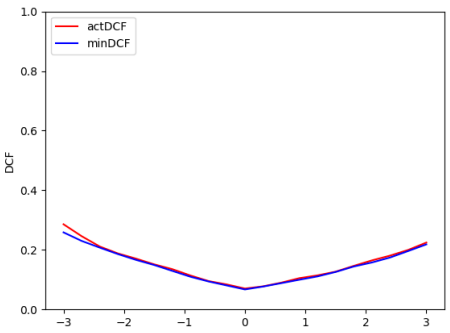

Figure 6.4: min vs act GMM

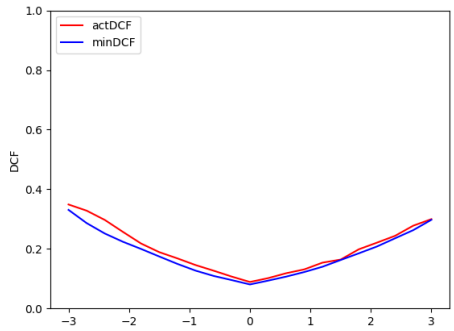

Figure 6.5: Calibration Quadratic LogReg

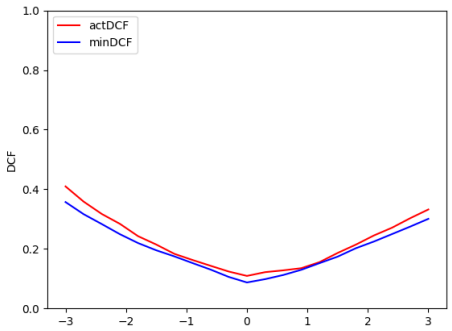

Figure 6.6: Calibration SVM RBF

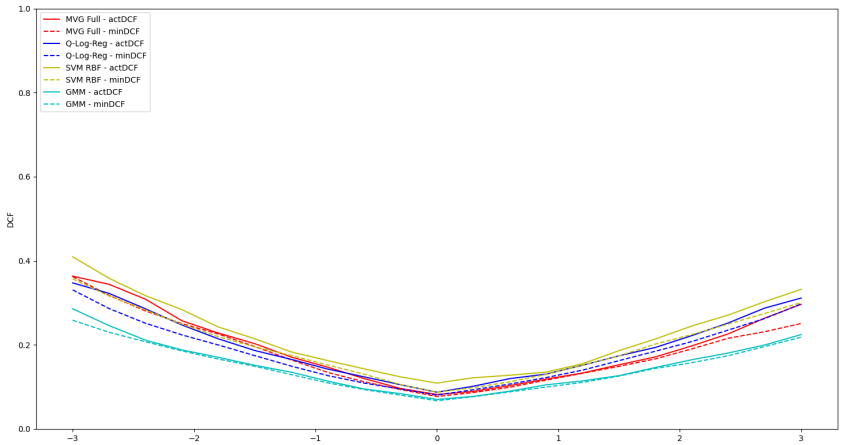

Figure 6.7: Model comaparis by actual and min DCF

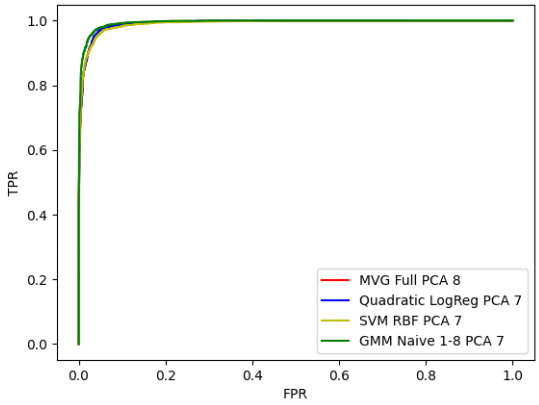

Figure 6.8: ROC curve plot

Considerations

The score calibrations results comply with the expected ones, GMM Naive is the models that performs better than all the other as shown in the images above.

Chapter 7

Conclusion

7.1 Conclusion

In conclusion, our approach, started with a particular attention to non-linear model and ended with the choice of **GMM NAIVE 1C - 8C** is actually accurate with the exception that in the evaluation part the number of components of non target class is 16. Overall with this choice we are able to get a cost of 0.250 for the given working point ($\pi = 0.5, C_{fn} = 1, C_{fp} = 10$) in the validation part.