Fingerprint Spoofing Detection

- Lorenzo Bellino 309413

Introduction

Abstract

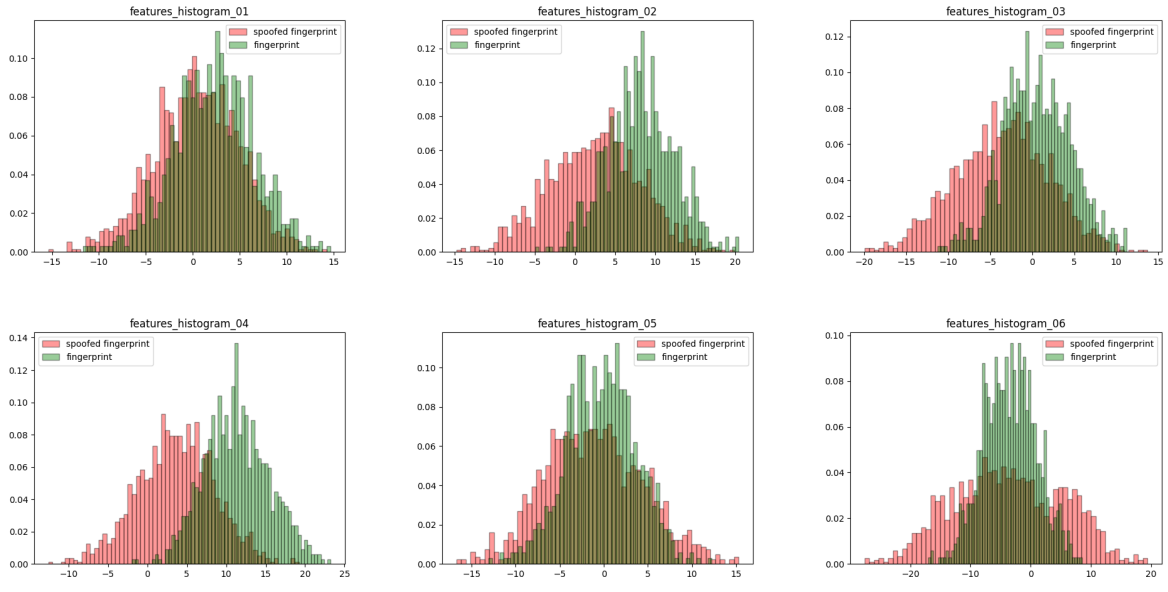
This report aims to examine a dataset composed of 2325 images of fingerprints using different Machine Learnin approaches. The dataset is composed of 2 classes: real and fake fingerprints. The goal is to classify the images in the correct class. The report is divided in 3 parts: the first one is about the preprocessing of the images and explore the dataset, the second one is about the training of the models and the third one is about the evaluation of them.

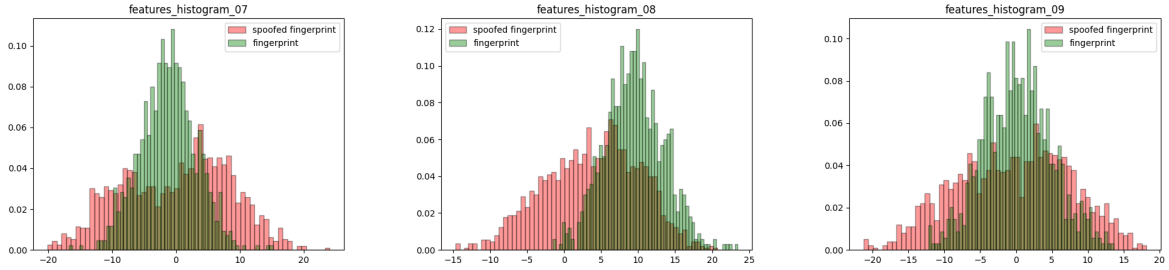
Considerations about the dataset

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

Feature Analysis

In the following images we can see the distribution of different features in the dataset diveded between authentic and spoofed fingerprint:

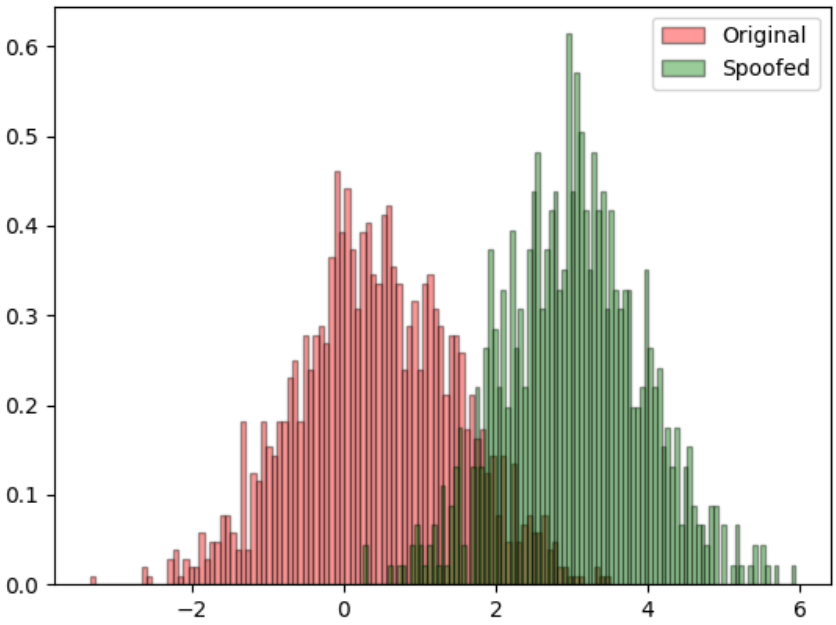




As we can see most of these features can be described by a Gaussian distribution, especially the first 4 and it is also evident that the distribution of the authentic fingerprint is much more similar to a normal distribution than the spoofed one, this can be explained by the fact that the spoofed fingerprint are actually a combination of different techniques and this could be the reason why the distribution is not so regular.

LDA

The first transformation that we can apply is the Linear Discriminant Analysis (LDA), this preprocessing step allow us to evaluate the features and decide if they can be linearly discriminble or if a gaussian model would be a better fit for this particular task. Looking at the graph we see how it is possible to use linear separation methods for the two classes but the separation would not be perfect and would lead to a number of errors.

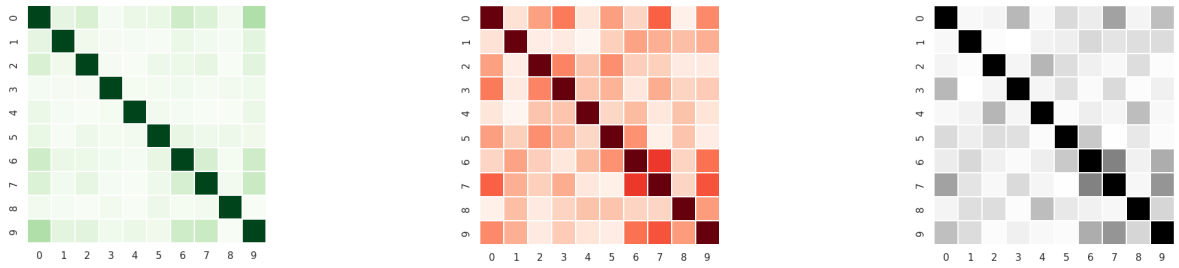


Correlation

Now we focus our attention of feature correlations using Pearson Correlations Plots

Authentic	Spoofed	All dataset

Authentic	Spoofed	All dataset



As we can see from the graphs, the correlation between the features is very low especially in the authentic fingerprint, this means that the features are not correlated. On the other hand in the plots for the spoofed fingerprint we can see that there are some features that are correlated, For example, features 6,7 and 9 seem to be quite correlated. this suggests that we may have benefit if we map data from 10-dimensional to 7-dimensional or 6-dimensional space in order to reduce the number of parameters to estimate for a model.

Explained Variance and Scree Plot

In order to decide how many features are needed in order to accurately represent the dataset we can look at the explained variance of the features and the scree plot.

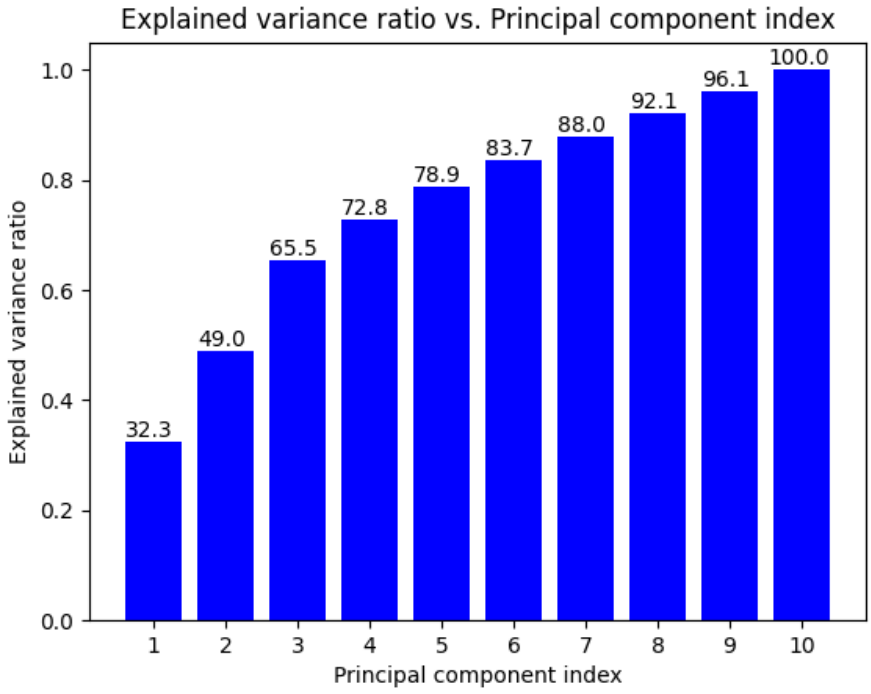
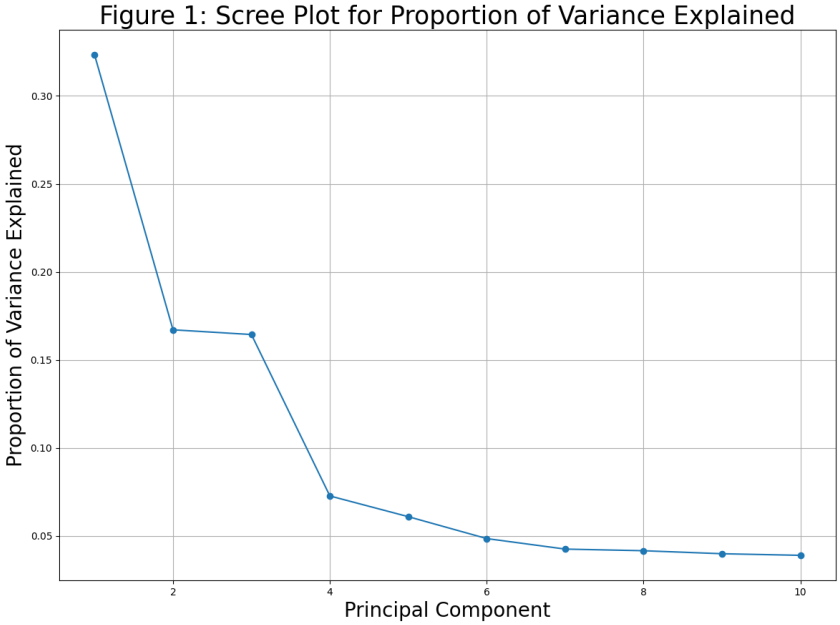


Figure 1: Scree Plot for Proportion of Variance Explained



As we can see the first 6 components explain almos 90% of the variance of the dataset, this means that we can reduce the dimensionality of the dataset from 10 to 6 without losing too much information. Furthermore, looking at the scree plot we can see that the components after the 6th have a value of explained variance below 0.05 this means that they are not very useful for the classification task.

Classification

For all of the classification tasks we will use the following metrics:

- K-Fold Cross Validation with $K = 5$
- working point at $(π, Cfn, Cfp) = (0.5,1,10)$

Multivariate Gaussian Classifier

We are now ready to begin our examination of machine learning models by looking at Gaussian classifiers such as the full covariance (MVG), the diagonal covariance (Naive Bayes, which assumes that assumes that covariance matrices Σ are diagonal matrices), as well as their tied assumption, which means that each class has its own mean μc but the covariance matrix Σ is computed over the whole dataset.

Since the dataset distribution can be approximated by a Gaussian distribution we can assume that MVG will perform with good result on our dataset.

	MVG	**MVG + diagonal covariance**	**MVG + tied covariance**	**MVG + diagonal Covariance + tied covariance**
minDCF PCA = 6	0.330	0.385	0.483	0.555
minDCF PCA = 7	0.337	0.383	0.481	0.559

	MVG	MVG + diagonal covariance	MVG + tied covariance	MVG + diagonal Covariance + tied covariance
minDCF PCA = 8	0.339	0.385	0.484	0.552
minDCF PCA = 9	0.341	0.380	0.479	0.551
minDCF PCA = None	0.337	0.475	0.472	0.548

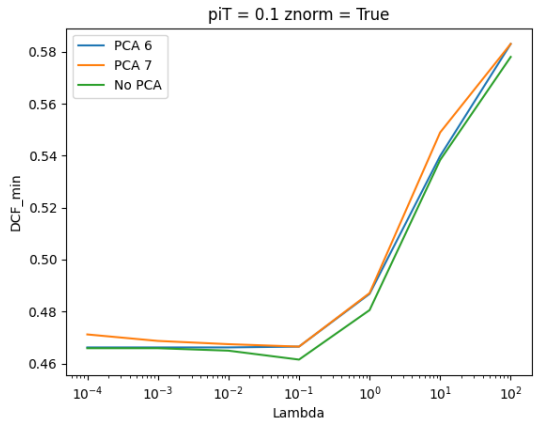
As expected The MVG model yelded the best results indicating that quadratic models are the best fit for this dataset. while the Naive approach is not much worst the model with MVG + tied covariance + diagonal covariance is the worst one. Different values of PCA did not affect the results so we need to conduct more test in order to understand if the PCA is actually useful for this task.

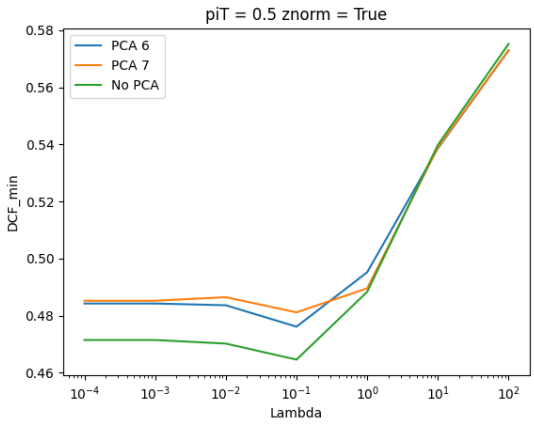
Logistic Regression

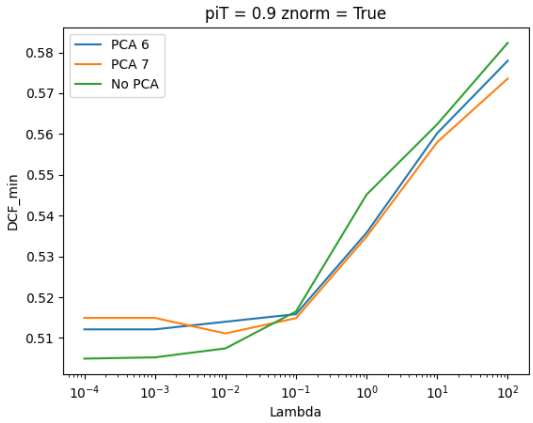
Now we can start to test for some Logistic Regression Models. At first the balanced version of logistic regression will be tested, but the results are expected to bee rather poor as a consequence of the dataset's classes being unbalanced. For this reason and because of the results obtained with the quadratic MVG model we can expect better results from the quadratic logistic regression model.

Linear Logistic Regression

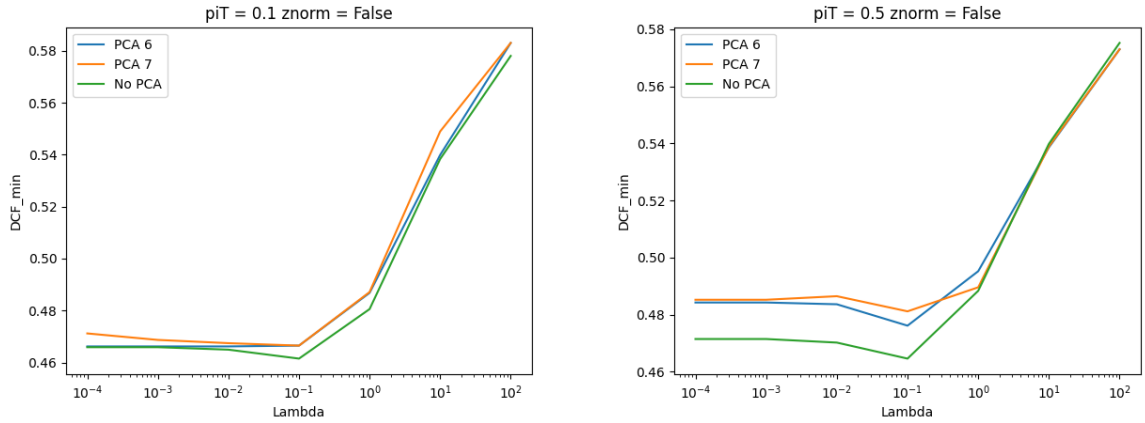
In this first graphs we can see the values of minDCF for a range of valuse of λ ,using both Z-Normalization and no Z-Normalization versions. As we said before, we tried different PCA and even different value of πT to see how results changed.

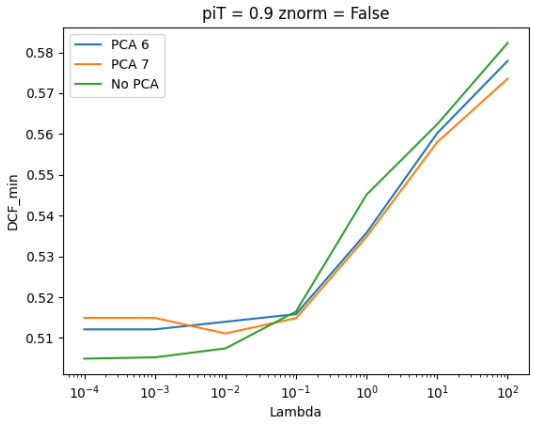






In the 3 images above we can see valuse of minDCF for different values of piT and PCA with Znormalization applied.





In the 3 images above we can see valuse of minDCF for different values of piT and PCA without Znormalization applied.

As we see from these plots, the minimum is reached by setting λ=0.1, so we will consider it as the best value for hyperparameter λ. We can notice that PCA with 6 components helped us reduce the minDCF value, while Z-normalization did not improve the performance. In fact, even when we combined PCA and Z-normalization, we got worse results than with PCA alone. In the table below are the results with and without Z-normalization for differnt values of λ and PCA=6 and as we can see Z-normalization did not improve the results.

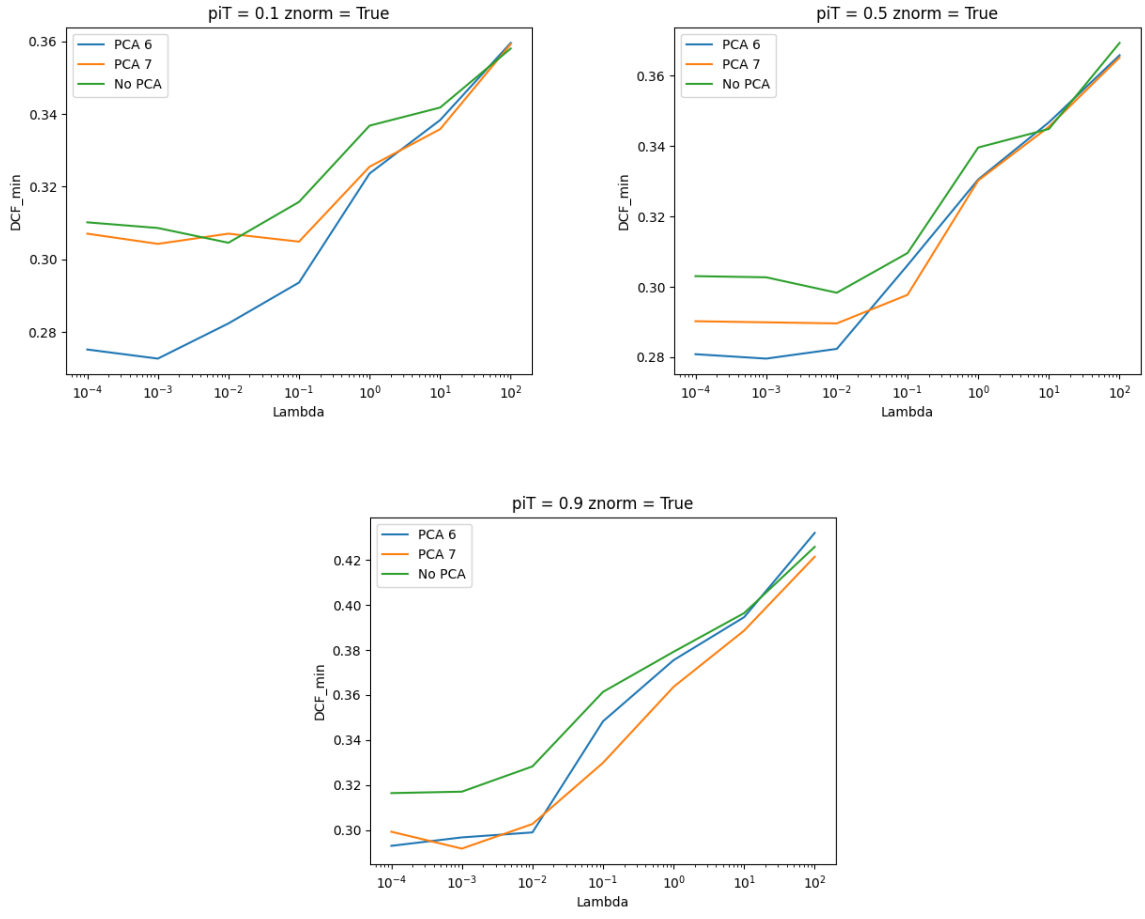
λ	Z-Normalization	no Z-Normalization

λ	Z-Normalization	no Z-Normalization
10^{-4}	0.469	0.471
10^{-3}	0.467	0.468
10^{-2}	0.463	0.465
10^{-1}	0.466	0.466
10^{0}	0.486	0.486
10^{1}	0.539	0.541
10^{2}	0.583	0.582

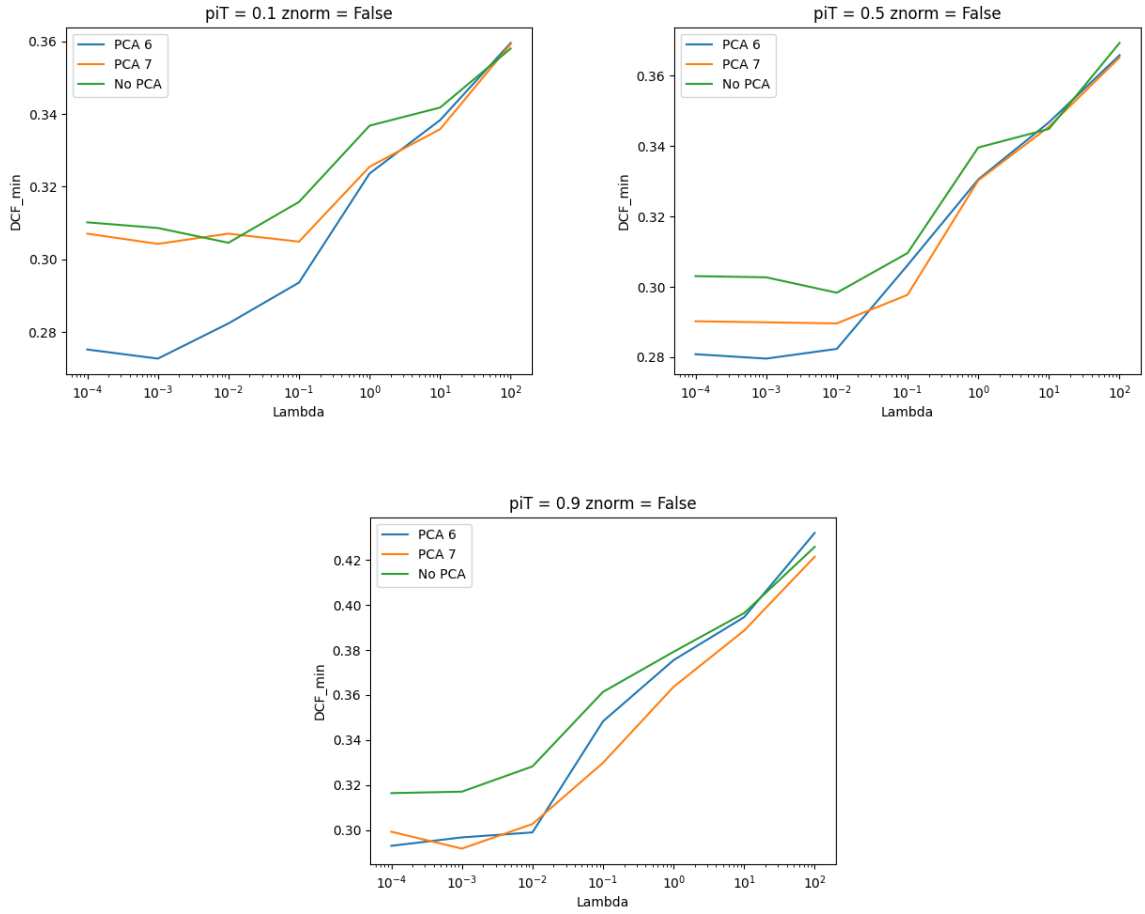
from the graphs above we can also deduce that the value of πT that fits best our dataset is 0.1 while 0.5 and 0.9 are not good values for this task. An other consideration regards the value for λ and we can see that the best results are obtained with $10^{-2} < λ < 10^{-1}$.

Quadratic Logistic Regression

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



In the 3 images above we can see valuse of minDCF for different values of piT and PCA without Znormalization applied.



In this case we can see a notable improvement in the results, in fact the minimum is reached by setting $λ=10^{-3}$, so we will consider it as the best value for hyperparameter λ. We can notice that PCA with 6 components helped us reduce the minDCF value, while Z-normalization did not improve the performance. In fact, even when we combined PCA and Z-normalization, we got worse results than with PCA alone.

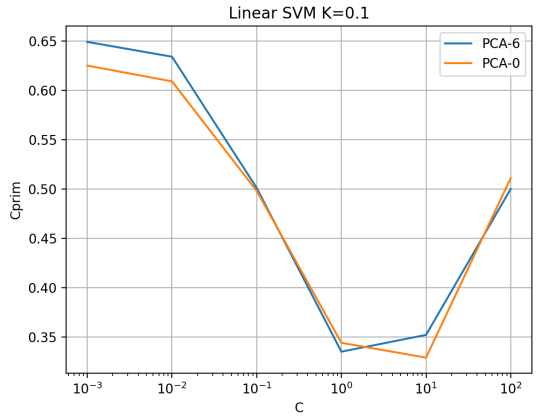
In the table below we can see values ov minDCF for different values of λ and PCA=6.

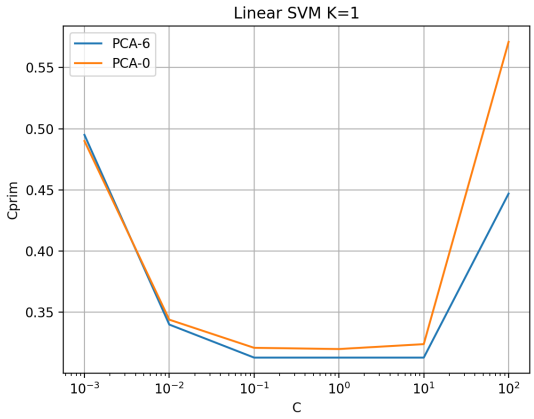
λ	Z-Normalization	no Z-Normalization
10^{-4}	0.275	0.275
10^{-3}	0.272	0.272
10^{-2}	0.282	0.285
10^{-1}	0.293	0.294
10^{0}	0.323	0.323
10^{1}	0.338	0.338
10^{2}	0.359	0.358

Support Vector Machines

Linear SVM

Given the poor result with linear models we expect linear SVM to perform suboptimally. We tested anyways various values of C and K, in particular $C = 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{2}$ and $K = 0.1, 1$. Below we can see the rusults for this tests.





In the table below are reported the results for all valuse of C and K.

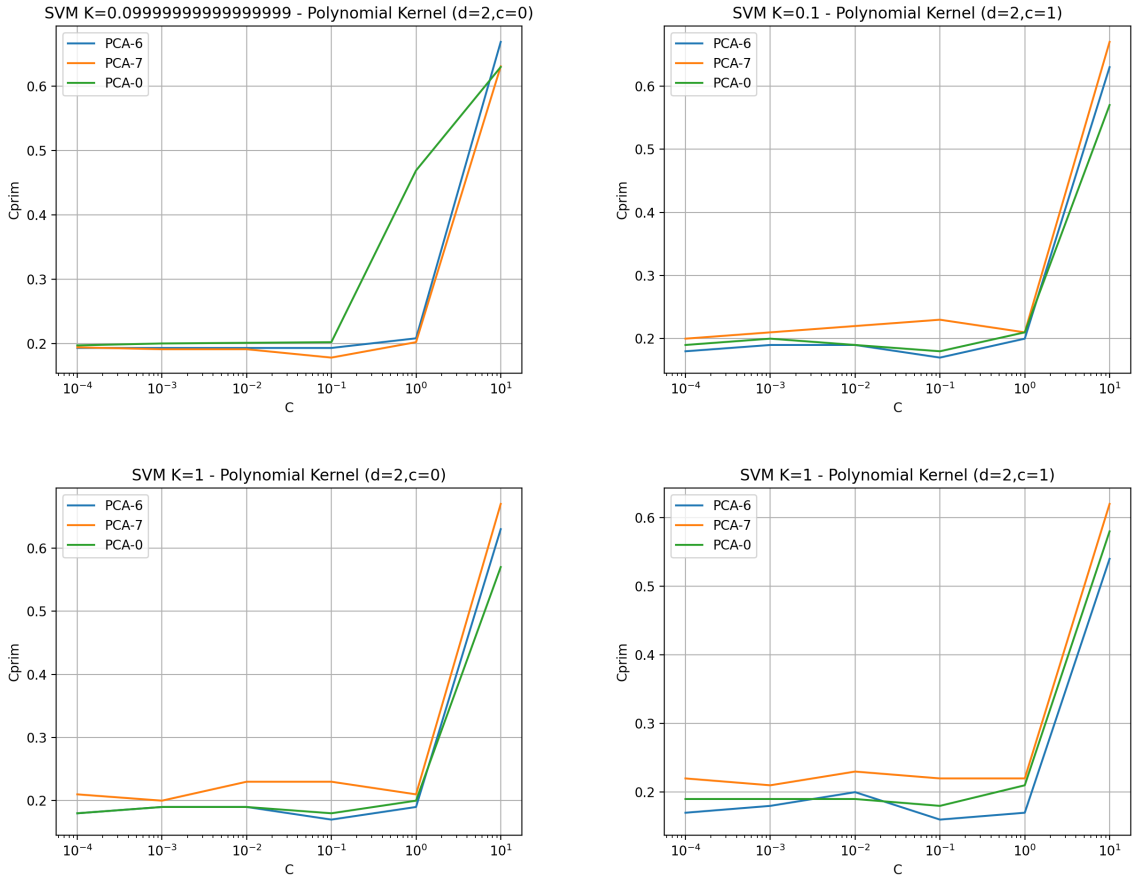
K	C	PCA	minDCF (pi = 0.1)	minDCF (pi = 0.5)	Cprim
0.1	0.001	6	0.935	0.364	0.649
0.1	0.01	6	0.912	0.356	0.634
0.1	0.1	6	0.723	0.279	0.501
0.1	1.0	6	0.478	0.192	0.335
0.1	10.0	6	0.523	0.17	0.352
0.1	100.0	6	0.711	0.288	0.5
0.1	0.001	0	0.498	0.183	0.341
0.1	0.01	0	0.471	0.174	0.322
0.1	0.1	0	0.471	0.173	0.322
0.1	1.0	0	0.459	0.178	0.314
0.1	10.0	0	0.463	0.174	0.318
0.1	100.0	0	0.466	0.174	0.32
1	0.001	6	0.711	0.278	0.495
1	0.01	6	0.488	0.191	0.34
1	0.1	6	0.459	**0.167**	0.313
1	1.0	6	0.456	0.173	0.313
1	10.0	6	0.455	0.171	0.313
1	100.0	6	0.639	0.254	0.447
1	0.001	0	0.483	0.179	0.331

K	C	PCA	minDCF (pi = 0.1)	minDCF (pi = 0.5)	Cprim
1	0.01	0	0.462	0.172	0.317
1	0.1	0	0.461	0.175	0.318
1	1.0	0	0.467	0.174	0.321
1	10.0	0	0.467	0.172	0.32
1	100.0	0	0.465	0.172	0.318

As we can see the best results are obtained with $C = 1$ and $K=1$

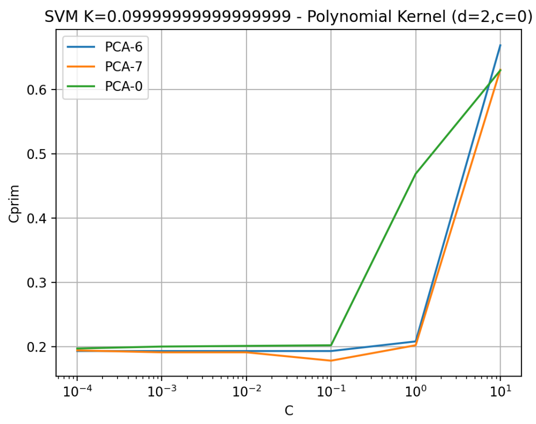
Quadratic SVM - Polynomial Kernel

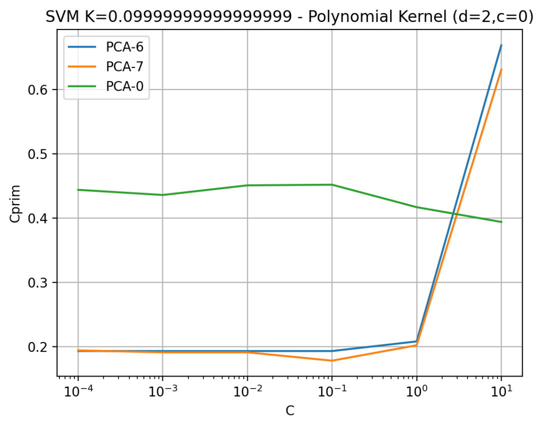
As we said before, we expect quadratic SVM to perform better than linear SVM, because our data seems to be better separated by quadratic rules. We tested various values of C and K, in particular $C = 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{2}$ and $K = 0.1, 1$. Below we can see the rusults for this tests.



As we can see the models without PCA and with $PCA = 6$ perform better than the other models, in particular the best results are obtained with values ov $C < 1$. The value of K does not seem to affect the results a lot, only some marginal imporvements are noticeable with $K = 1$.

An other options would be to apply some Z-normalization to the dataset, but as we can see from the graphs below, the results are not improved by this operation so this option is discarded.





The image on the left is the result of the model without Z-normalization, while the image on the right is the result of the model with Z-normalization.

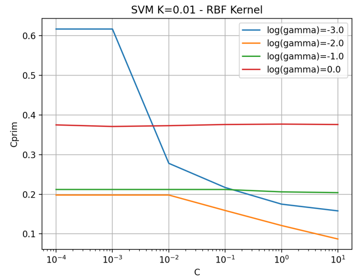
In conclusion the best results for each combination of C and K are reported in the table below.

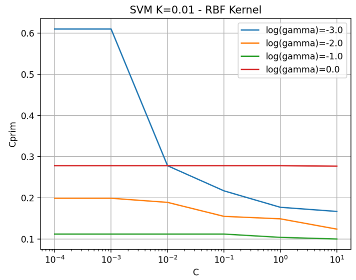
K	c	PCA	minDCF
0.1	0	6	0.191
0.1	1	6	0.175
1	0	6	0.179
1	1	6	0.172

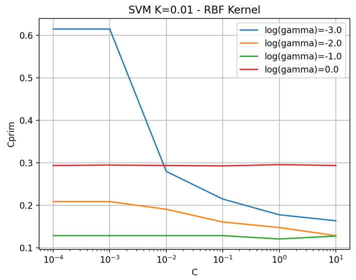
Quadratic SVM - RBF Kernel

As we said before, we expect quadratic SVM to perform better than linear SVM, because our data seems to be better separated by quadratic rules. We tested different values of $gamma$ for the RBF kernel, in particular $gamma = 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{2}$ and $C = 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{2}$ and $K = 0.1$ and $K = 1$.

K = 0.1 and PCA = 0,6,7

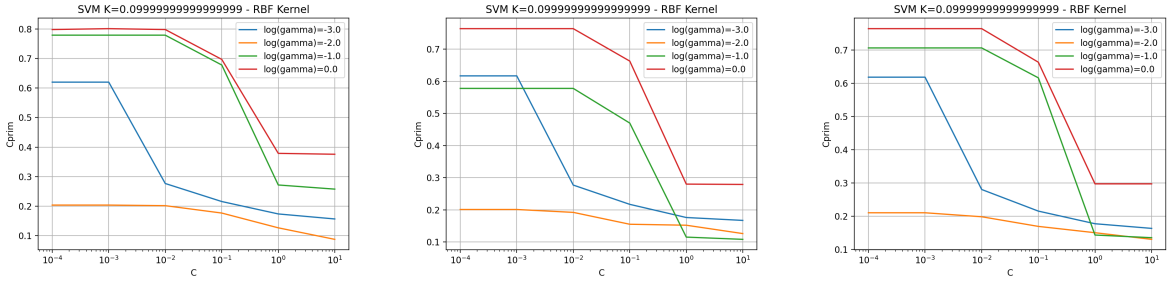






we can see some small improvement using PCA. The best results are obtained with $PCA = 6$ and $gamma = 10^{-1}$.

K = 1 and PCA = 0,6,7



Again PCA seems to improve the results, in particular the best results are obtained with $PCA = 6$ and $gamma = 10^{-1}$.

The Best results are reported below.

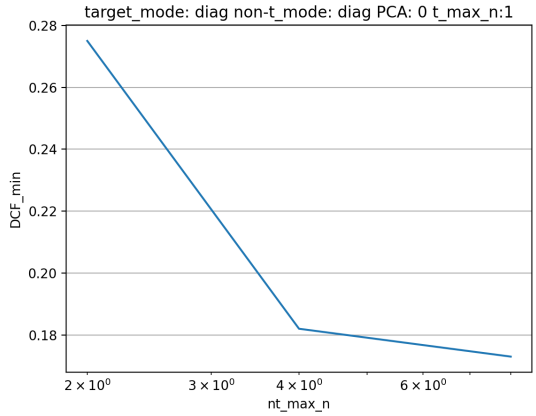
K	c	gamma	PCA	minDCF
0.1	1	0.1	6	0.172

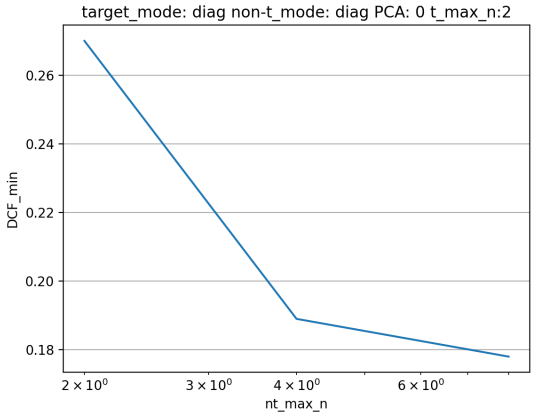
Gaussian Mixture Model - GMM

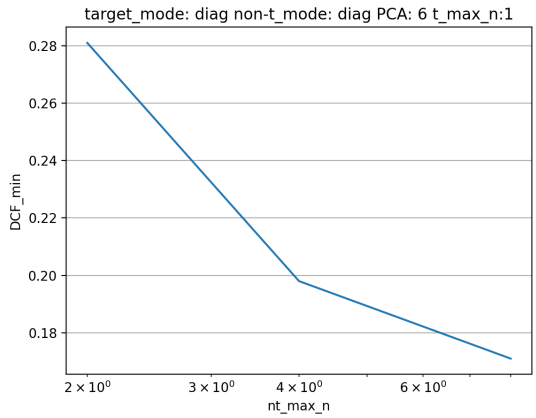
The last type of classifier is the Gaussian Mixture Model. This assumes that it is more convinient to represent the data with gaussian distributions with multiple components. Considering previous results with MVG and Naive Bayes we expect some correlations between classes but since MVG and NB results were comparable GMM can be expected to perfor good as well. The problem is approaced by trying different combinations of model and component for non-target class and target class. In particular:

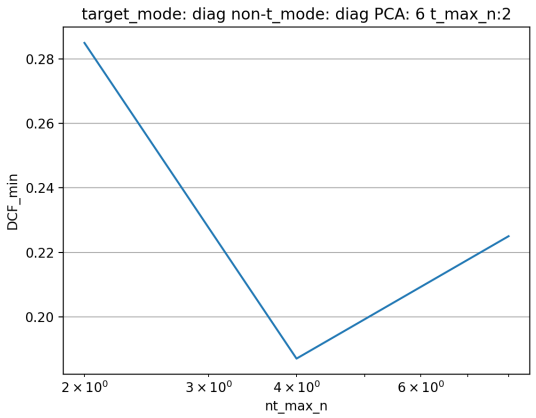
- $1,2$ components for target class Authentic
- $2,4,8$ components for non-target Spoofed

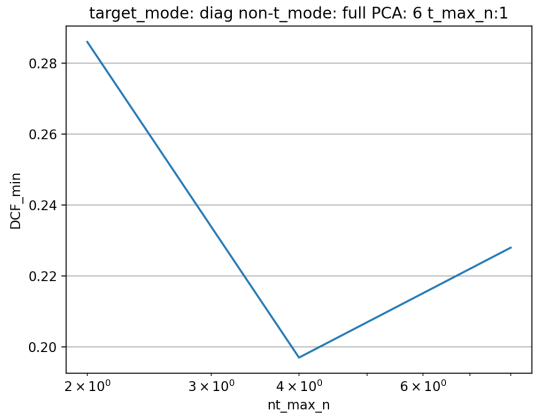
We know that Spoofed samples are distributed into 6 different sub-classes, so we expected more opti- mistic results when representing Spoofed fingerprints with a higher number of components. We then tried to configure our model trying with all the combinations of the Full-covariance model, Diagonal-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

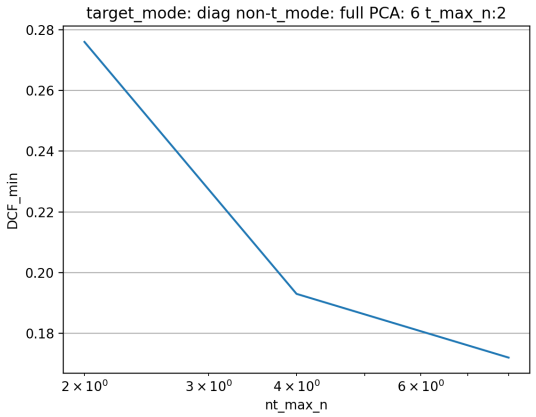

target_mode: diag non-t_mode: diag PCA: 0 t_max_n:1
DCF_min
0.28
0.26
0.24
0.22
0.20
0.18
2×10^0
3×10^0
4×10^0
6×10^0
nt_max_n

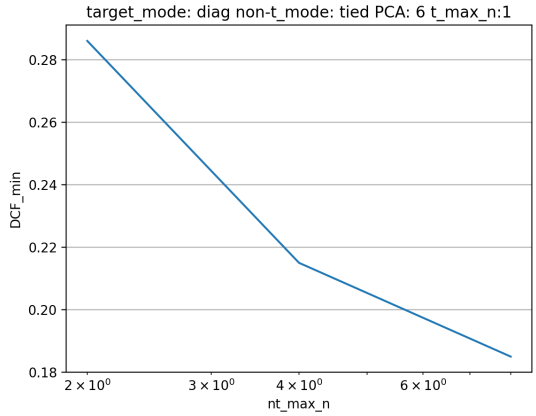

target_mode: diag non-t_mode: diag PCA: 0 t_max_n:2
DCF_min
0.26
0.24
0.22
0.20
0.18
2×10^0
3×10^0
4×10^0
6×10^0
nt_max_n

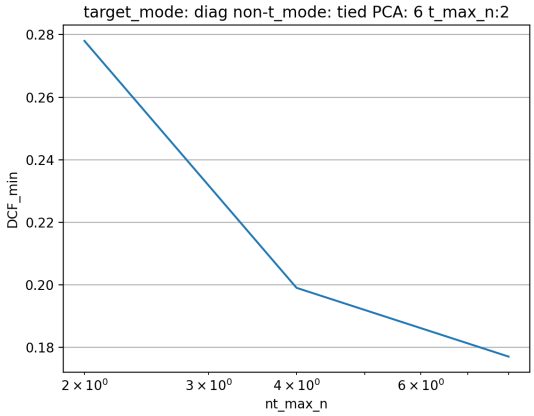

target_mode: diag non-t_mode: diag PCA: 6 t_max_n:1
DCF_min
0.28
0.26
0.24
0.22
0.20
0.18
2×10^0
3×10^0
4×10^0
6×10^0
nt_max_n

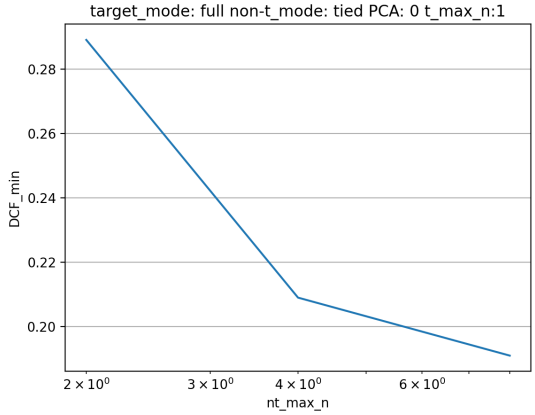

target_mode: diag non-t_mode: diag PCA: 6 t_max_n:2
DCF_min
0.28
0.26
0.24
0.22
0.20
2×10^0
3×10^0
4×10^0
6×10^0
nt_max_n


target_mode: diag non-t_mode: full PCA: 6 t_max_n:1
DCF_min
0.28
0.26
0.24
0.22
0.20
2×10^0
3×10^0
4×10^0
6×10^0
nt_max_n


target_mode: diag non-t_mode: full PCA: 6 t_max_n:2
DCF_min
0.28
0.26
0.24
0.22
0.20
0.18
2×10^0
3×10^0
4×10^0
6×10^0
nt_max_n







So we reported some of the most iconic plots. Among some of them, you can find some of the lowest DCF values. It is interesting to note that the best results are obtained by applying 8 components to the non-target class, which confirms what was previously assumed about the Spoofed Fingerprint distribution over several sub-classes. In addition, performance is better for the number of components equal to 2 for the target class. A value however low enough to confirm the analysis at the beginning of the report about the Gaussian distribution of Authentic samples. The same can be said when you notice that the best performance is obtained for the configuration with Diag Cov for both classes. Only the absence of PCA for the best configuration deviates from what has been stated so far. We decided to report just the results for the best configurations, so the cases with Target class samples represented by a Diagonal Covariance Gaussian

In the graphs above it can be seen that the best results are obtained with with the configuration of Diagonal Covariance applied to both **target** and **non-target** classes and with 8 components for the non-target class and 2 components for the target class. This is consistent with the results that we have seen so far.

Calibration

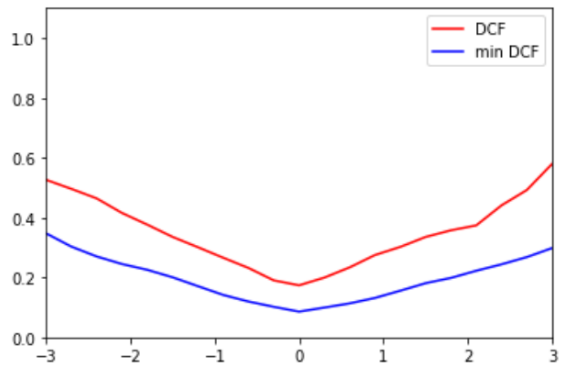
To evaluate the different models, so far we have used only the minimum cost of detection (minDCF), which however depends on a threshold. To understand if the threshold is the theoretical one, we will use a metric called actual DCF (actDCF). The method that we will adopt is based on Logistic Regression,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

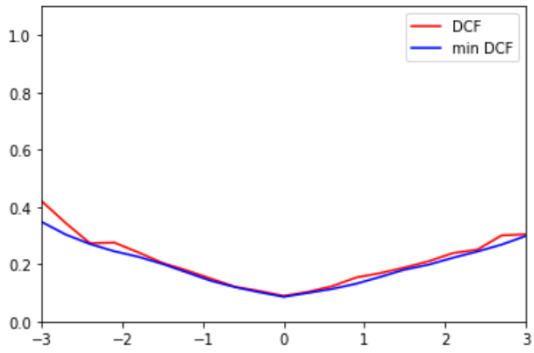
We take just the best 3 models to calibrate :

- GMM Target Mode=DiagCov e Non-Target Mode=DiagCov (2,8), PCA=6

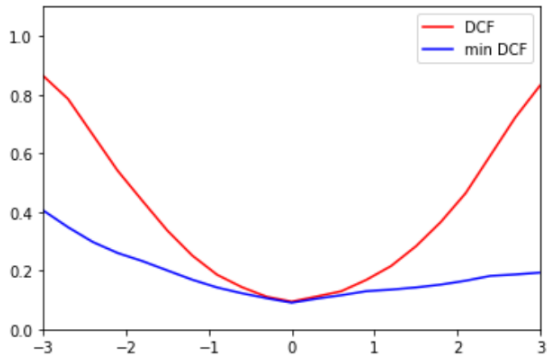
- Quadratic Logistic Rregression lambda = 10-2, PCA = 6, piT = 0.1
- SVM with kernel RBF, KSVM = 1, gamma = 10-3, C = 10, PCA = 6, piT = 0.1

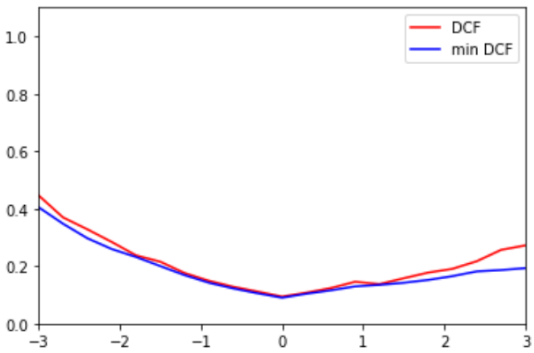
Quadratic Logistic Regression Non-Calibrated vs Calibrated



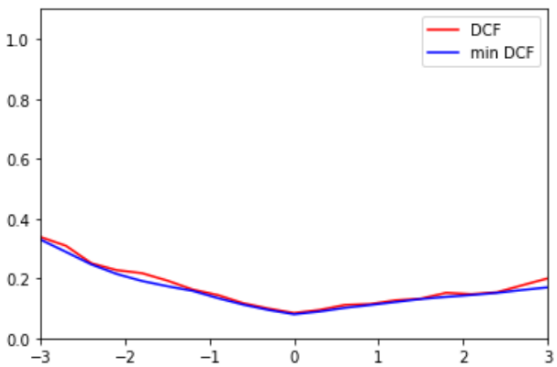


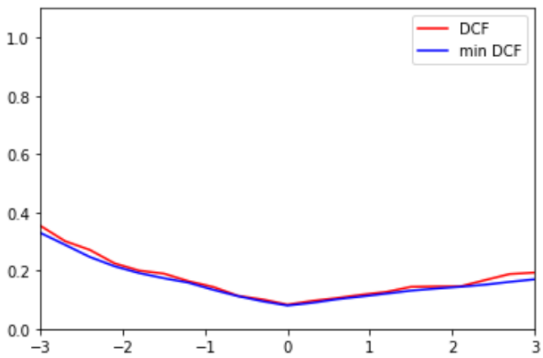
SVM with RBF Kernel Non-Calibrated vs Calibrated





GMM Non-Calibrated vs Calibrated

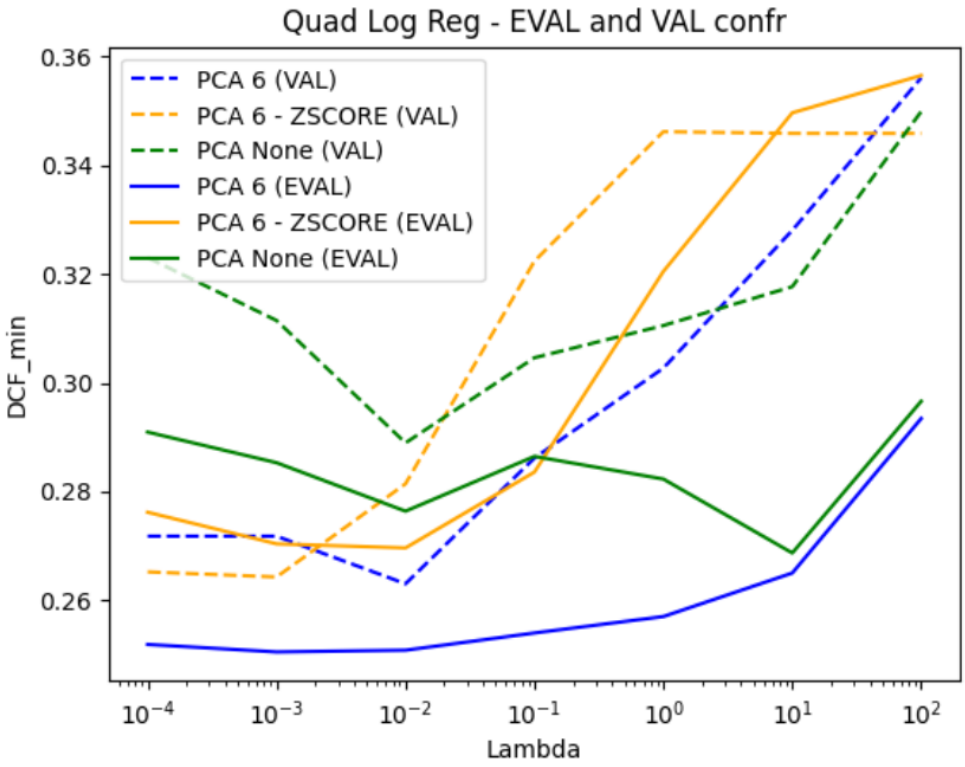




Evaluation

Now it is time for the avaluation phase of this experiment. the tests will be performed on the 3 best models previously described. The training and the test will be performed not on K-fold but directly on the Test set and Training set.

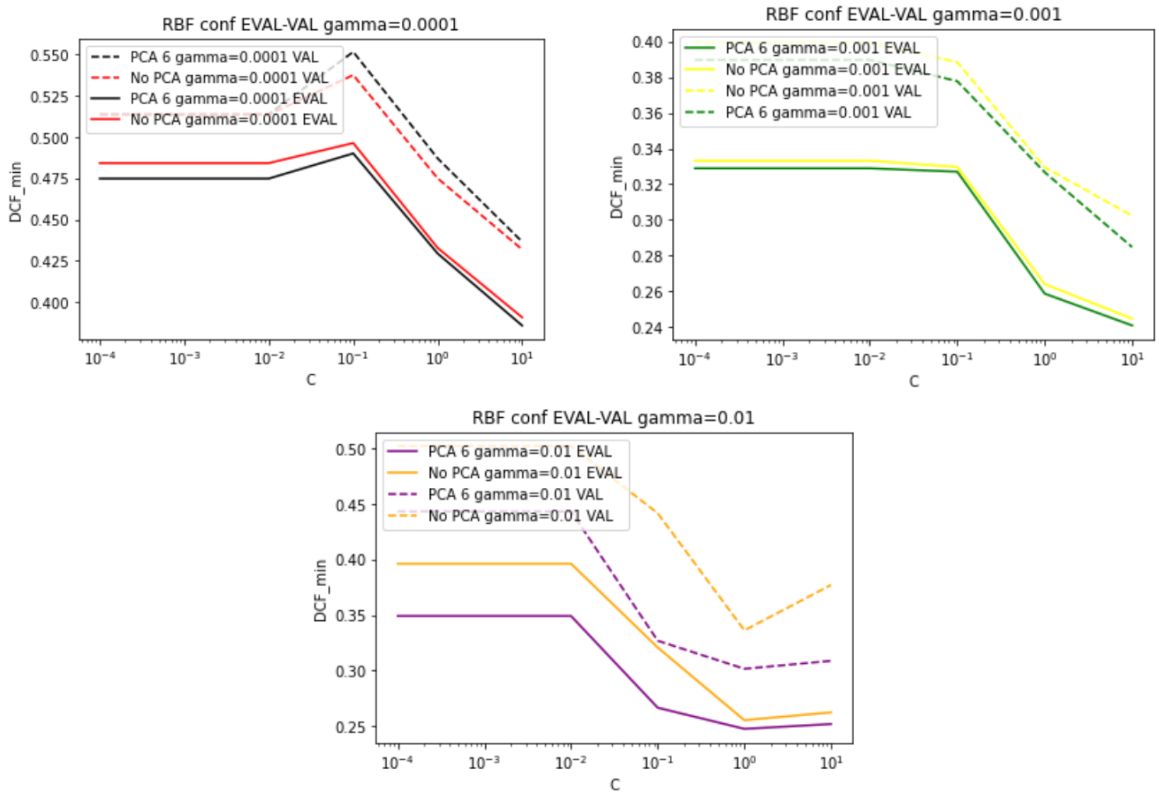
Logistic Regression



From the plot above we can see that the values of DCF are sligthly different but the trends of the models are very similar.

SVM with RBF Kernel

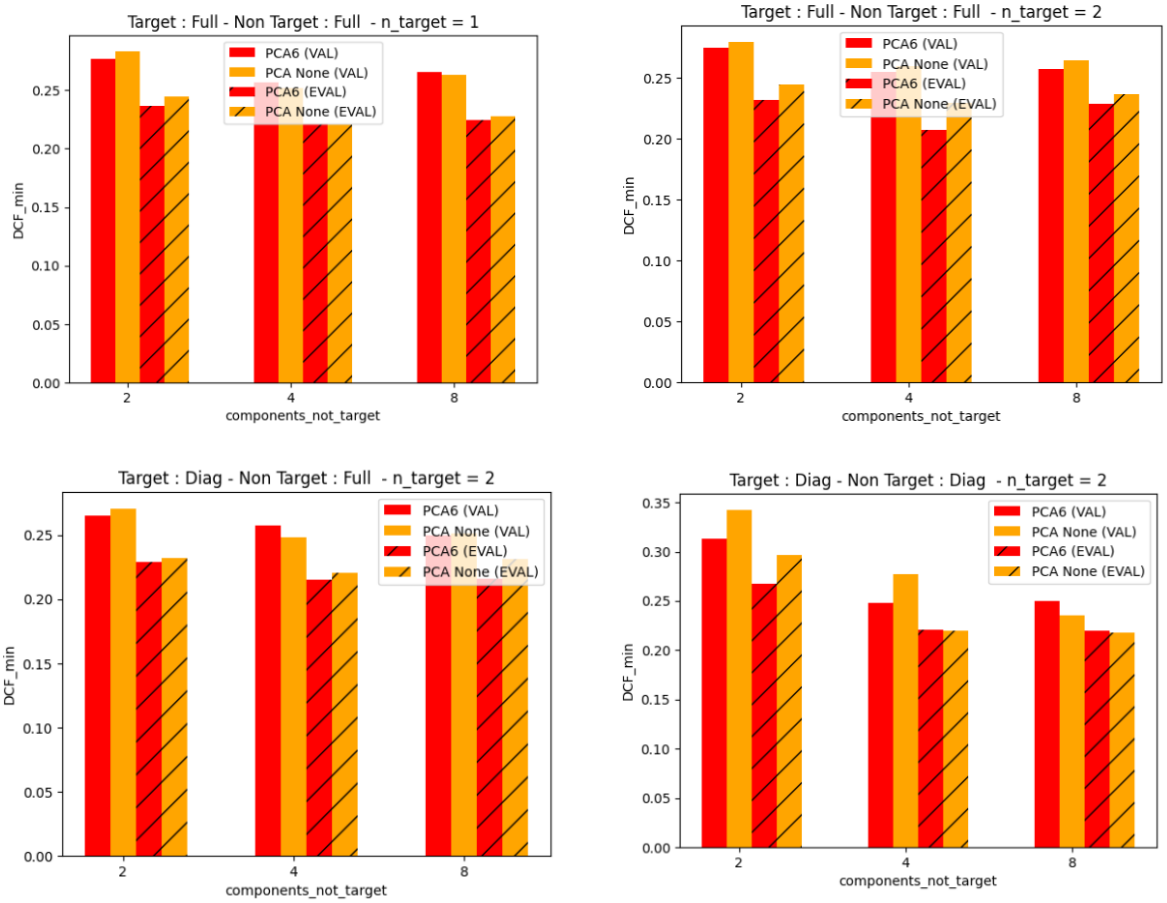
In this phase, we check that with RBF kernel we have the same results that in the validation's phase. The value of K is fixed to 1, changing $γ$ values in order to choose the best hyperpa-rameter.



Again the results seems to be consistent and the best results are obtained with $\gamma = 10^{-3}$ and $C = 10$ and with $PCA = 6$.

Gaussian Mixture Model

The last model to be considered is the Gaussian Mixture Model. We will use the same configuration as in the validation phase, so the best configuration is with Target Mode = Diag (2/8).



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

Conclusions

In general the results obtained during the Evaluation phase are similar to the ones obtained during the Validation phase. This means that the models are consistent and that the results are reliable. The best results are obtained with the GMM model, in particular with the configuration of Diagonal Covariance applied to both **target** and **non-target** classes and with 8 components for the non-target class and 2 components for the target class. This is consistent with the results that we have seen so far. The other models also performed well, in particular the SVM with RBF kernel and the Quadratic Logistic Regression. The calibration phase also confirmed the reliability of the models and the consistency of the results.