# Machine Learning & Pattern Recognition

## Fingerprint Spoofing Detection

Franco Edoardo – S310228
Chiabodo Alessandro – S309234

## Table of Contents

## Abstract

The main goal of this report is to analyze a dataset composed of thousands of low-level fingerprint samples using machine learning algorithms to determine whether the sample is a real fingerprint or a spoofed one.
Initially we’ll perform an analysis of the dataset to determine its structure and trying to understand its distribution and which algorithm could perform better.
Then we’ll analyze the performance of each one of the main Machine Learning algorithms seen during the lectures to try to determine which is suited more to deliver the best classification system based on the minimum cost.

## Dataset exploration

First, we'll analyze the dataset by plotting the classes distributions and correlations, starting also to see how the PCA can impact out dataset.

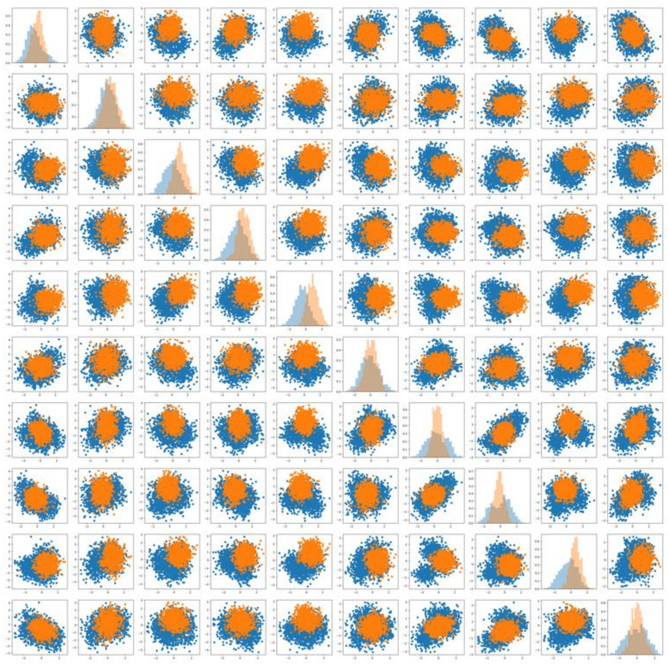


*Figure 1. Histograms and 2D scatter plots of the dataset*

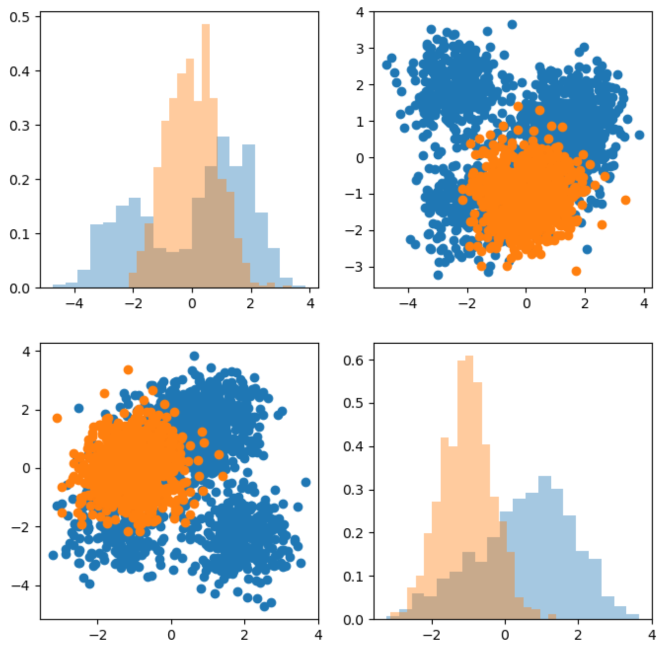


*Figure 2. Histograms and 2D scatter plots of the dataset – consider only the 2 principal directions*

We can observe that the non-target class is characterized by different classes. It's clearer in the PCA scatter plots that show different "groups" of non-target samples.

We can also suppose, by looking at the scatter plots, that linear decision boundaries will perform poorly, because of the overlapping of the samples.
Observing the histograms in the PCA plots, we can see that the target class follows a Gaussian-shaped distribution. This could lead our thoughts towards Gaussian classifiers.

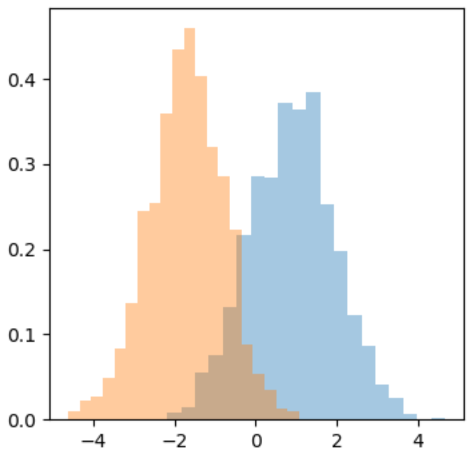


*Figure 3. Histogram of dataset features - LDA direction*

With LDA, we obtain only the main direction because it allows to compute at most C – 1 discriminant direction. By looking at the plot, we can see that data could be classified using linear decision boundaries, but scatter plots do not agree with this supposition.
We then expect that non-linear classifiers would perform better.

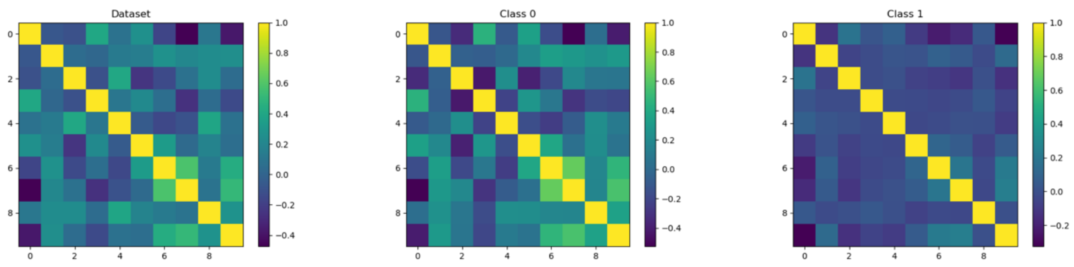


*Figure 4. Pearson correlation between features*

We can see that the target class has a very low correlation between feature, in contrast to what is seen in the correlation matrix of non-targets.

While PCA may remove some correlation, the number of dimensions is small, thus it may remove useful information that characterizes the target class. In contrast, it may remove some useless information in the non-target one. We will evaluate the performance of using and not using PCA.

We then analyze the number of PCA component that we could consider by plotting the relation between the number of components with PCA with the fraction of explained variance.
Possible values are those who retain about 0.95 of the dataset variance.

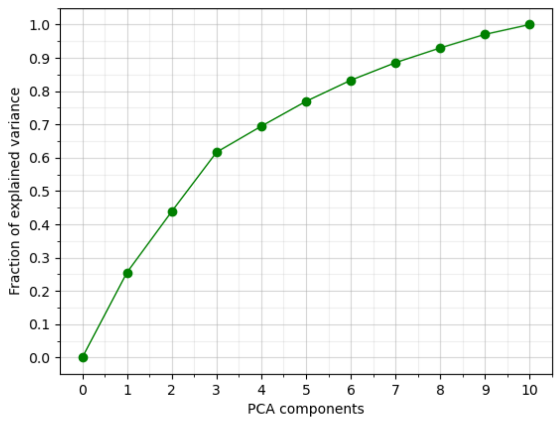


*Figure 5. PCA components over the fraction of explained variance*

Using PCA, most of the variance is taken out as soon as we start decreasing the number of features. Indeed, we could consider values between 7 and 9, because they retain around 90% to 97% of the total variance.

## Model evaluation

We now begin to analyze the various models. To extract evaluation and training datasets on which to perform our evaluations using only the training part of our data, we decide to use K-Folds, specifically using K = 5, which is a good trade-off in terms of time and computational cost.
We also analyze the effects of applying PCA with the values we selected in the previous stages.

For the moment we use minDCF as our objective metric.
Then, once we have chosen the best performing model, we will perform score calibration.

### 1. Multivariate Gaussian Classifier

The Multivariate Gaussian Classifier in its basic form is the first model we evaluate.

| PCA | minDCF |
|---|---|
| **No** | 0.329 |
| **9** | 0.349 |
| **8** | 0.337 |
| **7** | 0.32 |

We can see that the lowest value is obtained using PCA.
Furthermore, we choose the dimensions with the lowest value because they perform better.
We then try to also evaluate m equal to 6 and 5 to see if minDCF still decreases.

| PCA | minDCF |
|---|---|
| **6** | 0.352 |
| **5** | 0.36 |

Our thoughts are wrong, m = 6 and m = 5 gives the highest minDCF values, so we can discard them.

## 2. Naïve Bayes MVG

The Naïve Bayes version of the MVG assumes independence between the features of each class. This hypothesis is respected by the target class, but not for every feature in the non-target case, only few features are somehow correlated. However, we can try do apply this method and look at the results.
Since PCA diagonalizes the data covariance matrix, we include in the PCA dimension set the exact number of features of the dataset. This transformation does not reduce the number of dimensions, but it only diagonalizes the covariance matrix.

| PCA | minDCF |
|---|---|
| **No** | 0.464 |
| **10** | 0.366 |
| **9** | 0.37 |
| **8** | 0.376 |
| **7** | 0.375 |

As we expected, the Naïve assumptions are not exactly respected, this leads to worst results than the standard MVG.

## 3. Tied MVG

The tied version of the MVG instead, assumes that points of each class are equally spread, that practically means one covariance matrix for all classes.
The most important characteristic is that this version has linear decision boundaries.

| PCA | minDCF |
|---|---|
| **No** | 0.465 |
| **9** | 0.481 |
| **8** | 0.469 |
| **7** | 0.461 |

Model results are exactly what we expected, the tied MVG does not overperformed the quadratic versions.

## 4. Logistic Regression

As stated before and having in mind the results obtained in the last model evaluation, we expect linear classifiers to perform poorly compared to those non-linear.

In this case, we use cross-validation to select the best value of lambda according to minDCF. The first attempt is without any kind of normalization of the data but considering PCA.

| PCA | best λ | minDCF |
|---|---|---|
| **No** | 0.1 | 0.469 |
| **9** | 0.001 | 0.466 |
| **8** | 0.001 | 0.465 |
| **7** | 0.01 | 0.465 |

We then try z-score normalization to the input data.

| PCA | best λ | minDCF |
|---|---|---|
| **No** | 0.01 | 0.465 |
| **9** | 1e-5 | 0.474 |
| **8** | 0.0001 | 0.466 |
| **7** | 0.001 | 0.471 |

In this case, normalization does not improve performances, instead it seems to be harmful.
In both cases the results confirm our prior thoughts about linear decision boundaries.

We then try the quadratic feature expansion with Logistic Regression, and we expect to get better results.
As we did before, we first try without normalization.

| PCA | best λ | minDCF |
|---|---|---|
| **No** | 0.1 | 0.302 |
| **9** | 0.0001 | 0.322 |
| **8** | 0.01 | 0.302 |
| **7** | 1e-06 | 0.276 |

Observing the result obtained in the linear case, we could expect that z-score wouldn't be useful. Let's give it a try.

| PCA | best λ | minDCF |
|---|---|---|
| **No** | 0.001 | 0.287 |
| **9** | 0.1 | 0.297 |
| **8** | 0.001 | 0.287 |
| **7** | 0.1 | 0.287 |

Again, our prior assumptions were correct, this model achieves better performances without normalization.

For the moment, Q-Log-Reg with λ = 1e-6 and PCA with m = 7 and no normalization, is the best model we analyzed.

## 5. Support Vector Machine

We start to analyze the linear version of this type of models, but as in the previous cases, we expect a weak performance.

We tried different values of K (the normalization term) for the SVM, in particular we used 0, 1 and 10. Here we only report the results obtained with the last two, because using K=0 gave minDCF above 0.9 in almost all situations.

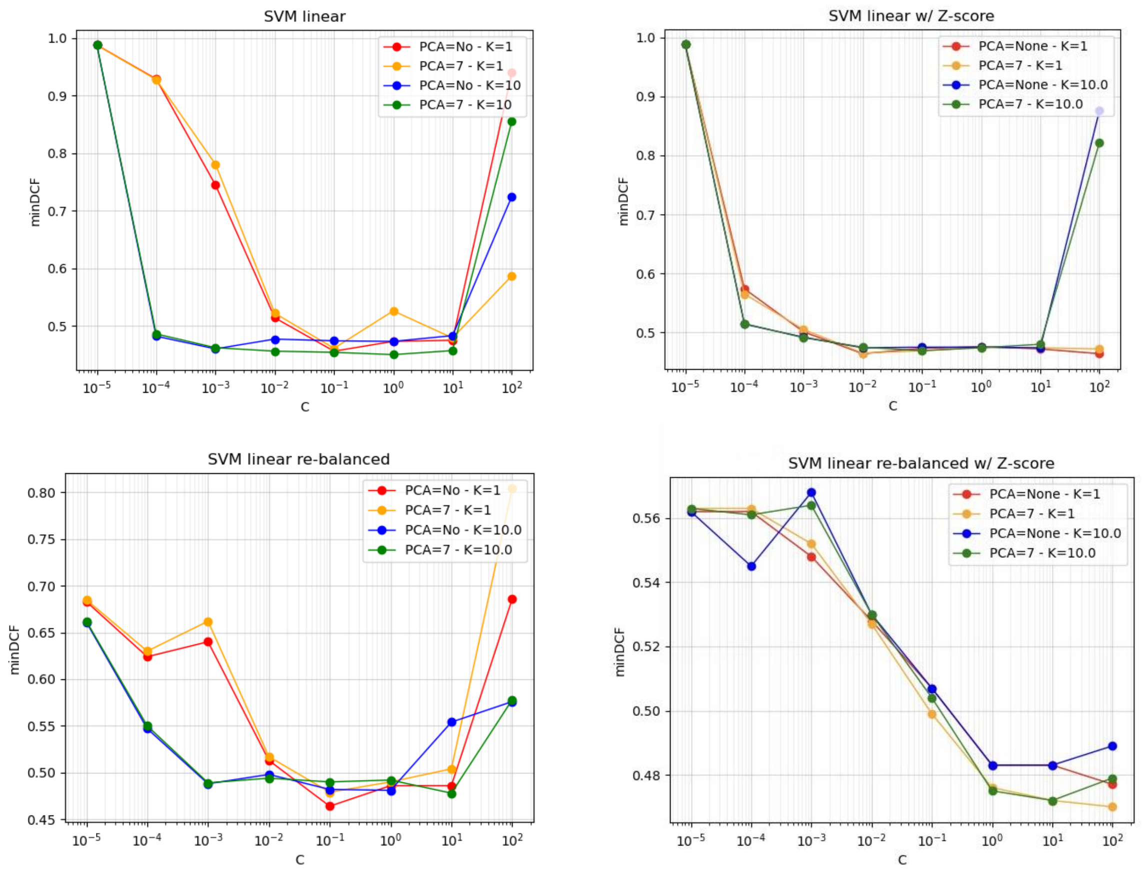


*Figure 6. SVM linear - hyper-parameter tuning*

Comparing these results, we can see that the z-score normalization has improved the performances of the model only in areas where the model was lacking, but in the best-case scenarios it didn’t change much.
Moreover, class-rebalancing did pretty much the same work as the normalization in term of performances, indeed the more “stable” results are given by the combination of rebalancing and normalization.
PCA instead, didn’t play a crucial role in the minDCFs as we can see comparing the same values of K.

To reduce the computation for the kernel stage, we can identify a reduced set of values of C where the model behaves best.
We then choose a range between 10-2 and 10.
We now evaluate the SVM with polynomial kernel, in particular with d = 2.

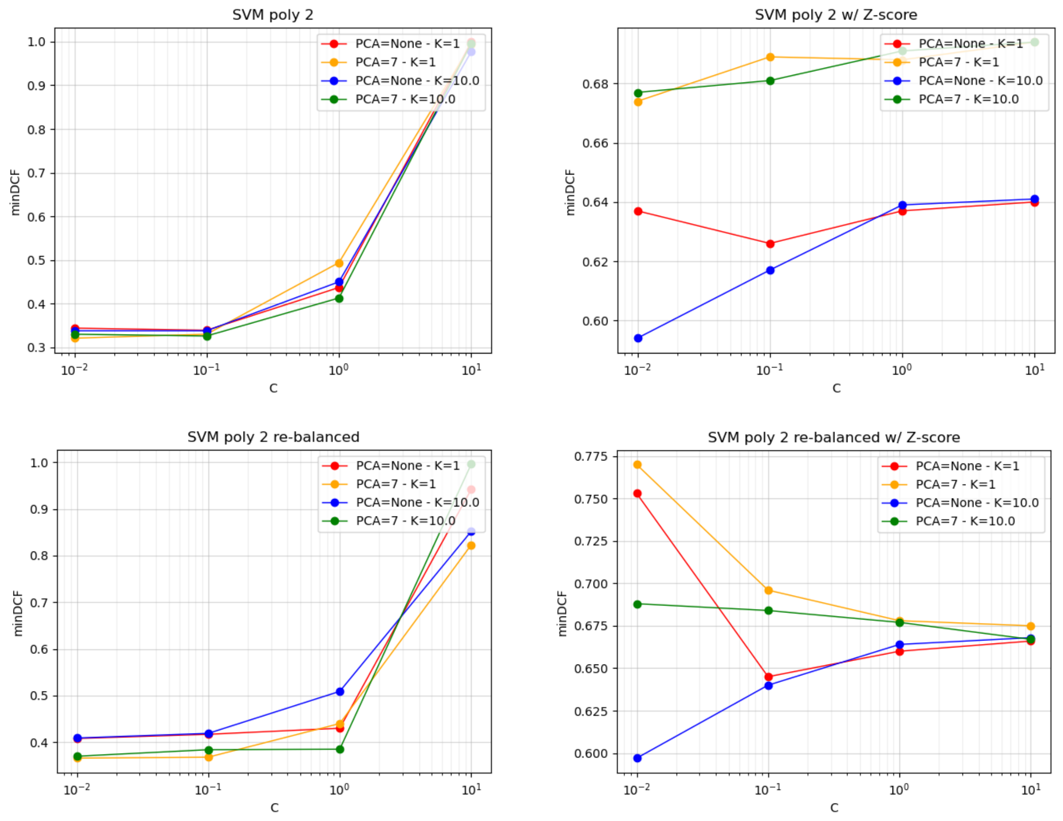
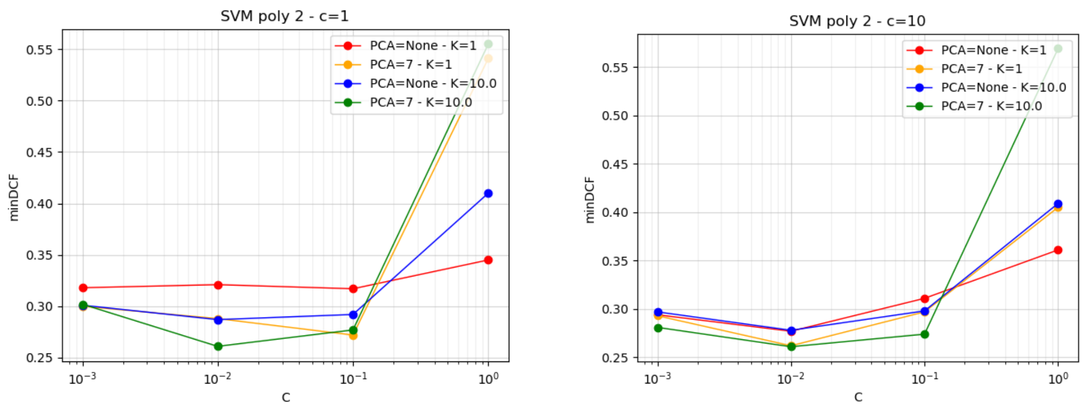


*Figure 7. SVM kernel polynomial degree 2 - hyper-parameter tuning*

We can immediately notice that normalization had a negative effect, indeed it provided 2 times worse results in lower values of C.
Rebalancing classes instead, didn't change much.

We now try different values of c, the constant term added to the dot product in the polynomial kernel function. By looking at the previous graphs, we can modify the values of C by removing 10 and adding $10^{-3}$ to better identify the minimum.



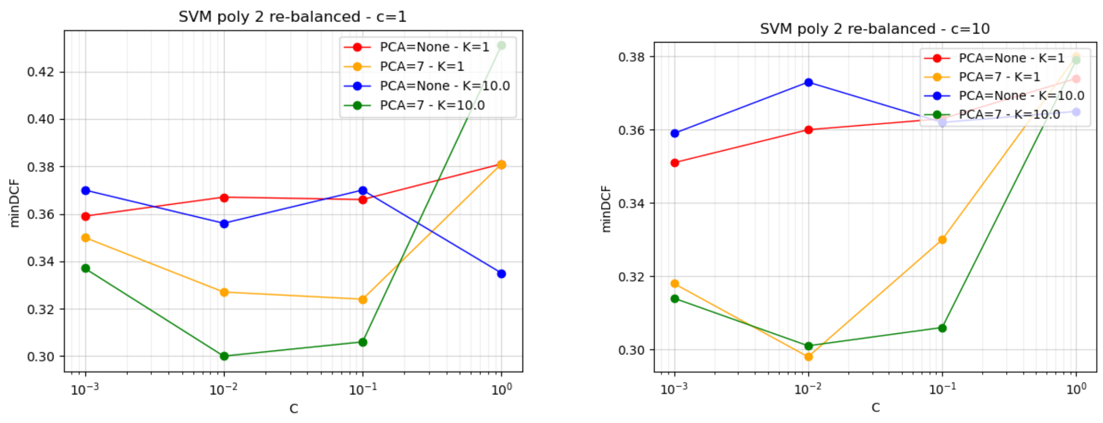


*Figure 8. SVM kernel polynomial degree 2- hyper-parameter tuning - "c" variations*

Values of c different than zero produce slightly better results, in particular with c = 10. Moving the window of values of C confirmed that the minimum (in almost all the cases) was on C = $10^{-2}$.

Class rebalancing seems to be ineffective also in these cases, for the moment we stop applying it and we will evaluate standard models.
Instead, in the standard version of the polynomial model, we found a value that overcomes the one in the q-log-reg.
With the configuration: c = 10, C = $10^{-2}$, d = 2, K = 10 and PCA with 7 dimensions we found a minDCF value of 0.261.
We now evaluate the polynomial kernel in 3 dimensions.

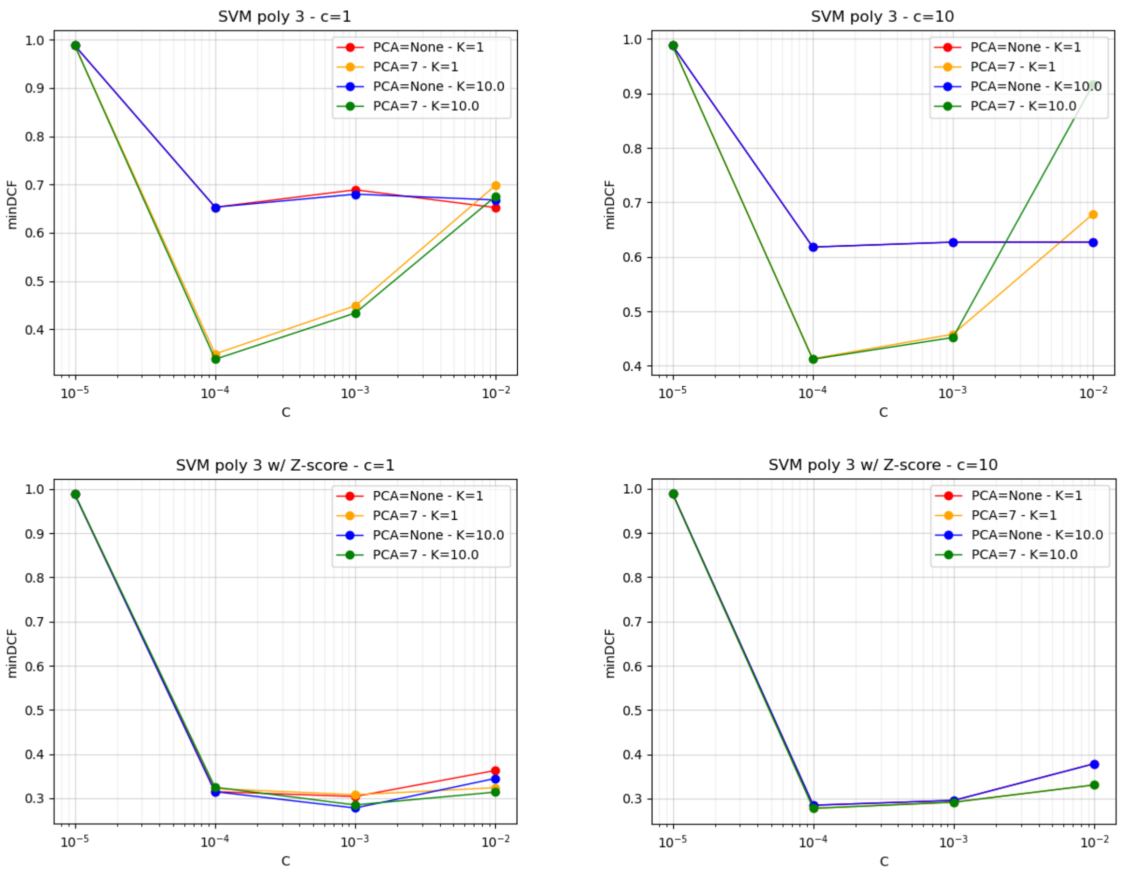


*Figure 9. SVM kernel polynomial degree 2 - hyper-parameter tuning*

In this case, normalization had a positive effect on the model, but neither the original or the normalized version of the degree 3 polynomial kernel beat the result obtained for the kernel of degree 2.
We finally evaluate the RBF with different values of gamma.

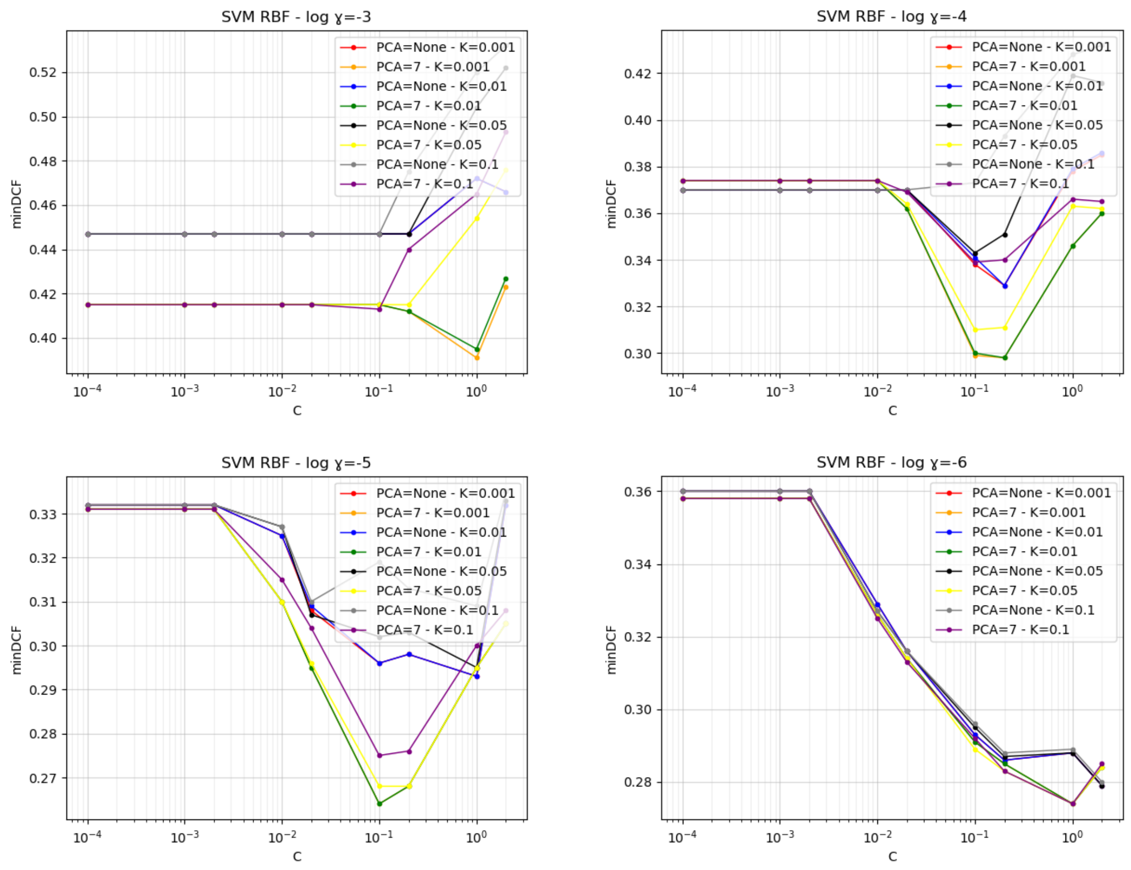


*Figure 10. SVM kernel RBF - hyper-parameter tuning*

In these cases we evaluated different values of K (= sqrt(ε)) while trying to find a minimum value. The minimum value of minDCF is reached by the configuration:
log ɣ = -5, K = 0.01, PCA dimensions = 7, C = 0.1 with a value of minDCF = 0.264, not small enough to outperform the polynomial of degree 2.

At this step we didn't find the best new model, but we had a confirmation that PCA is effective.

## 6. Gaussian Mixture Model (GMM)

It's now the moment for the last model to be evaluated, as the first step we evaluate the optimal number of dimensions for each GMM (for each class).

We have adopted a grid search approach by evaluating components from 1 to 32 for both the target and non-target classes, as we expect, from the analysis carried out in the data exploration phase, that the target class will consist of a few components and the non-target class of a larger number than the target class.

For visualization purposes, we split the results in two different plots.

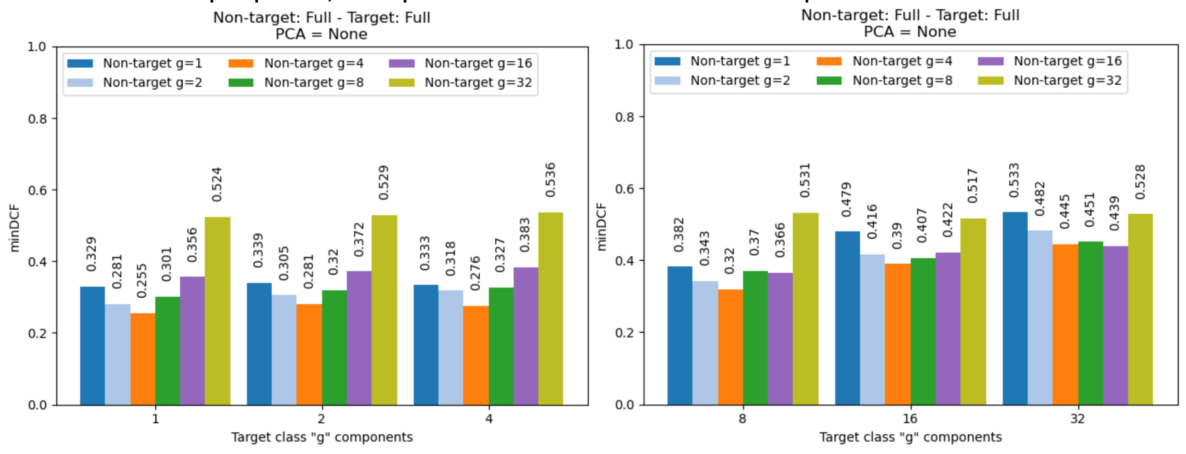


*Figure 11. GMM per-class number of clusters "g" - grid search*

The first thing we notice is that more components for the target class are the worst-case scenarios, exactly as we thought.
Then, we can look at the best combination of the "g" components in the first graph: g = 1 for the target class and g = 4 for the non-target one, at least for the full-covariance case.

We'll now try reducing dimensionality with different values of PCA.

| PCA | Target g | Non-target g | minDCF |
|---|---|---|---|
| **No** | 1 | 4 | 0.255 |
| **10** | 1 | 4 | 0.255 |
| **9** | 1 | 4 | 0.259 |
| **8** | 1 | 4 | 0.26 |
| **7** | 1 | 4 | 0.246 |
| **6** | 1 | 4 | 0.254 |

PCA with 7 dimensions improved minDCF by few points.

Previously, the diagonal Gaussian model didn't provide good results mostly due to the fact that the non-target class has quite high correlation between features, but in this case, that constraint is no more valid, so we will try different combinations of diagonal (D) and full covariance (F).

We also evaluate applying the tied full covariances version (FT) and the tied-diagonal version (TD) on both classes, but we expect more influence on the non-target one because seems that the clusters have similar distributions.

To reduce the time needed to train the models we will focus only on PCA = 7.
For simplicity we only report best values for each combination.

| PCA | Target g | Non-target g | minDCF |
|---|---|---|---|
| **7** | 1 - F | 4 - F | 0.246 |
| **7** | 2 - F | 8 - D | 0.243 |
| **7** | 2 - F | 4 - FT | 0.245 |

| | | | |
|---|---|---|---|
| **7** | 1 - F | 16 - TD | 0.235 |
| **7** | 1 - D | 4 - F | 0.246 |
| **7** | 2 - D | 16 - D | 0.237 |
| **7** | 2 - D | 4 - FT | 0.237 |
| **7** | 1 - D | 16 - TD | 0.235 |
| **7** | 4 - FT | 4 - F | 0.245 |
| **7** | 2 - FT | 16 - D | 0.242 |
| **7** | 4 - FT | 8 - FT | 0.244 |
| **7** | 1 - FT | 16 - TD | 0.235 |
| **7** | 8 - TD | 4 - F | 0.235 |
| **7** | 2 - TD | 16 - D | 0.241 |
| **7** | 4 - TD | 8 - FT | 0.241 |
| **7** | 16 - TD | 8 - TD | 0.229 |

Now that we have established that the best configuration for the GMM classifiers is the last one, we're going to try to find the combination of values for the hyperparameter alpha that will use this configuration.

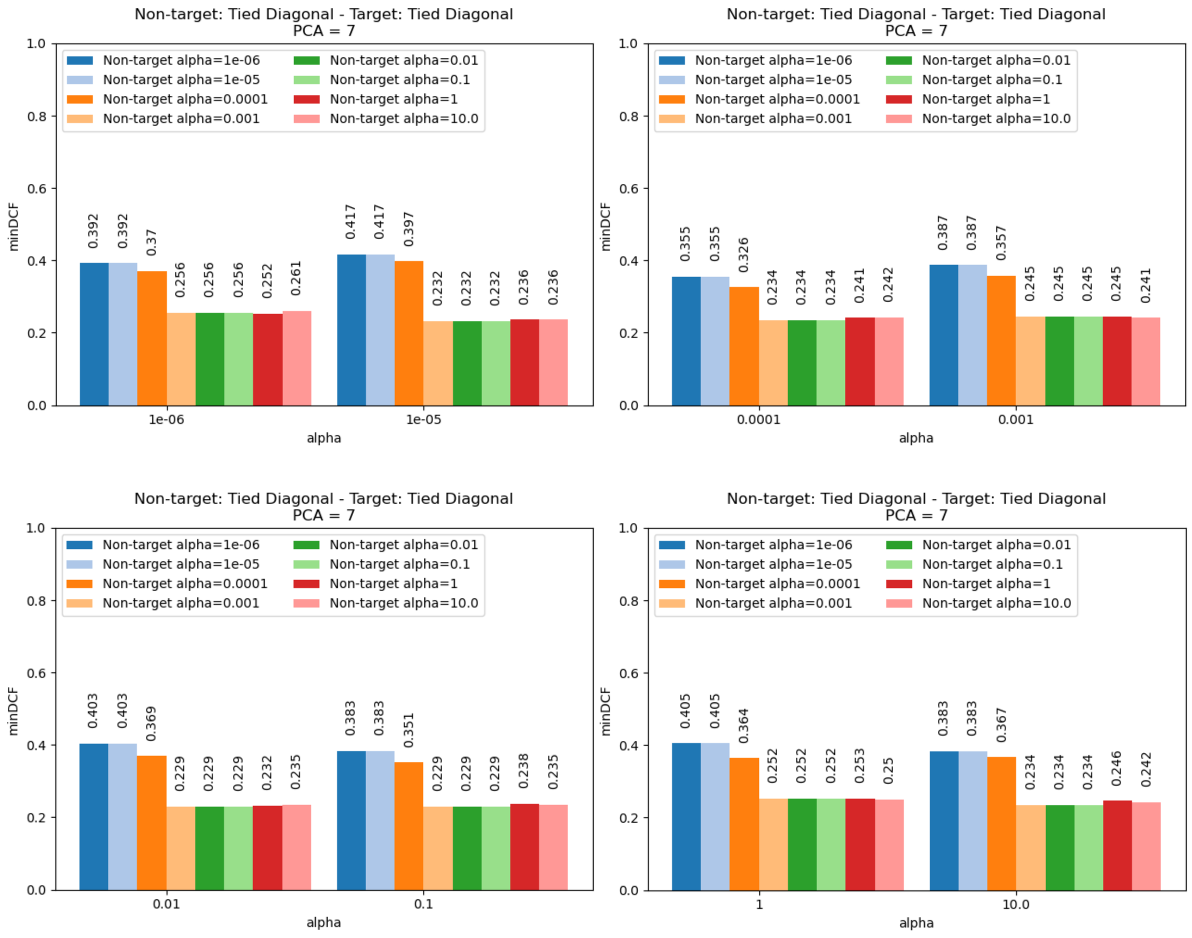


*Figure 12. GMM "alpha" hyper-parameter - grid search*

According to these plots, the best combination of values is between 0.01 and 0.1 for the target class and between 0.001 and 0.1 for the non-target class. The minimum value of minDCF is equal to that obtained in the previous step, which means that we have already used one of the best pairs of alphas (i.e., 0.1 for both classes).

The application of diagonalization and the tied covariances step gave good results, especially when the number of components was increased compared to the standard case.

We can then consider the configuration (16 - TD), (8 - TD) as the best evaluated for GMMs and also the best model so far.

## Summary

The best results we obtained are summarized in the following table.

| Model | minDCF |
|---|---|
| **Poly SVM**<br>d = 2, c = 10, C = $10^{-2}$, K = 10, PCA=7 | 0.261 |
| **RBF SVM**<br>log γ = -5, K = 0.01, C = 0.1, PCA = 7 | 0.264 |
| **GMM**<br>Target: 16 comp - Tied diagonal<br>Non-target: 8 comp - Tied diagonal<br>PCA = 7 | **0.229** |

We don’t consider Q-Log-Reg because it is very similar to the polynomial version of the SVM, and it provided worse results.

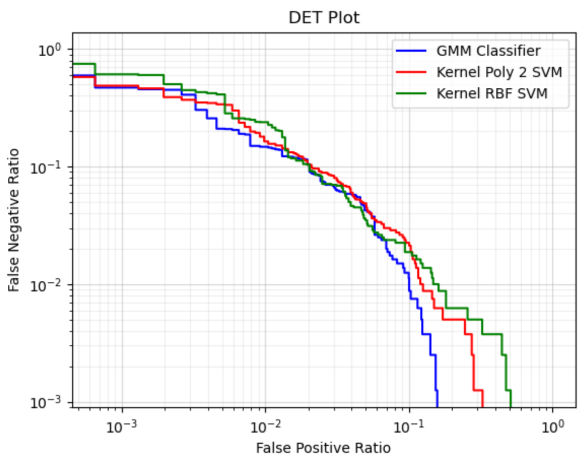


*Figure 13. DET plot - comparison between selected models*

From the DET plot we can see that, in almost every situation, the GMM is the model that gives better results.

We now consider calibration of these models and analyze whether score-level fusion may provide additional improvement.

## Score calibration

So far, we've only used the minDCF as a metric to evaluate the different classifiers, but the cost we pay depends on the threshold used in the classifier, which can be very different from the best "theoretical" one that would allow a better classification.
We'll now introduce the actual DCF as a metric that also takes into account the calibrated score of the classifier.

| Model | PCA | minDCF | actDCF |
|---|---|---|---|
| **GMM** | 7 | 0.229 | 0.235 |
| **Poly 2 SVM** | 7 | 0.261 | 0.396 |
| **RBF SVM** | 7 | 0.264 | 1.0 |

This table shows that the GMM model is fairly well calibrated, while the two SVMs are not, with the emphasis on the RBF model, which gives useless predictions.

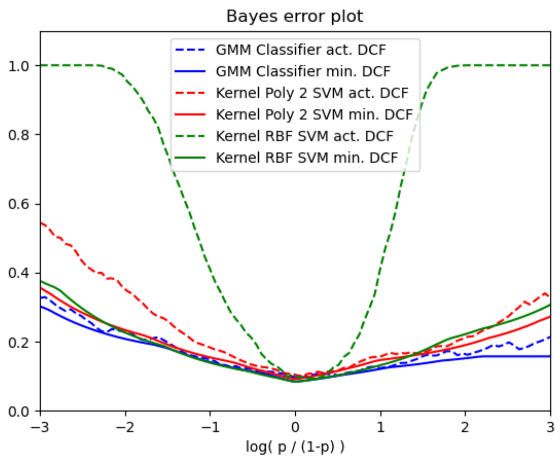


*Figure 14. Bayes error plot for the three selected models*

In figure 14 we have a wider view of how well calibrated the models are.
The graph remarks what we found out in the previous table.
We will now apply score calibration.

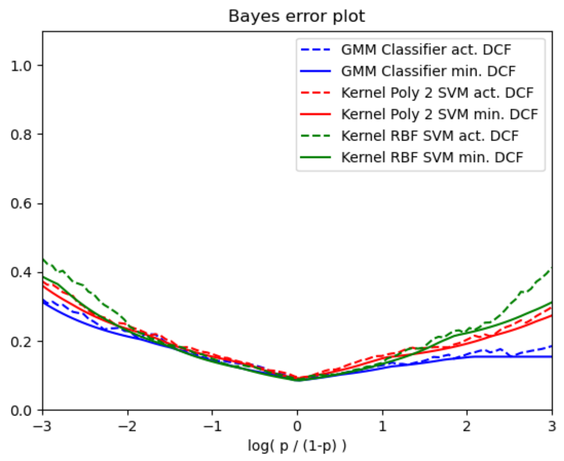


*Figure 15. Bayes error plot for the three selected models - calibrated*

The following results have been obtained by applying k-folds approach.

| Model | PCA | minDCF | actDCF |
|---|---|---|---|
| **GMM** | 7 | 0.230 | 0.237 |
| **Poly 2 SVM** | 7 | 0.260 | 0.266 |
| **RBF SVM** | 7 | 0.265 | 0.313 |

## Model fusion

We tried all the possible combination of the three selected models and here are the results:

| Model | PCA | minDCF | actDCF |
|---|---|---|---|
| **GMM + Poly 2 SVM** | 7 | 0.255 | 0.269 |
| **GMM + RBF SVM** | 7 | 0.249 | 0.270 |
| **Poly 2 SVM + RBF SVM** | 7 | 0.273 | 0.281 |
| **All three** | 7 | 0.252 | 0.272 |

The fusion seems to be ineffective, having each possible combination to perform worse than the single models.
However, we select the fusion of GMM and Poly 2 SVM (the one that shown best performances) for further analysis.

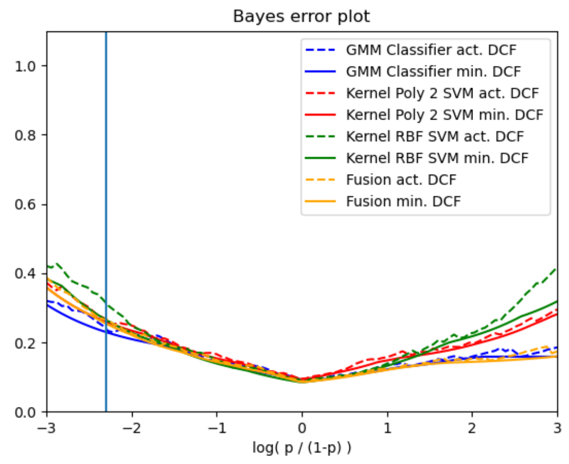


*Figure 16. Bayes error plot comparing selected models and their fusion - calibrated*

The Bayes error plot confirms the results obtained in the previous point, especially in the area near our working point (the vertical blue line).

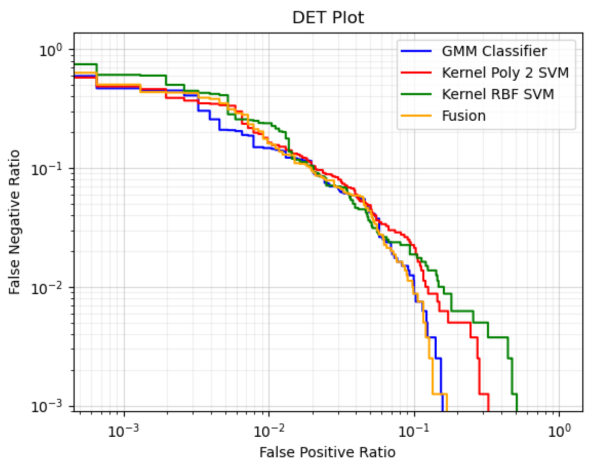


*Figure 17. DET plot comparing the chosen models and the fusion model*

Taking a further look to our four models (including the chosen fusion) by means of DET plot, we can see that the best models are the GMM and the fusion or the poly SVM alone depending on the area we are referring to.

However, our final choice is the Gaussian Mixture Model alone.

## Evaluation

We finally analyze the performances of our chosen models on the test dataset, training all of them on the entire train dataset.
The following table compares the results obtained with the evaluation set and the ones obtained (using cross validation) during the previous analysis using only the training set.

| | | Validation set (with K-Folds) | | Evaluation set | |
|---|---|---|---|---|---|
| **Model** | **PCA** | **minDCF** | **actDCF** | **minDCF** | **actDCF** |
| **GMM** | 7 | 0.230 | 0.237 | 0.226 | 0.229 |
| **Poly 2 SVM** | 7 | 0.260 | 0.266 | 0.247 | 0.248 |
| **RBF SVM** | 7 | 0.265 | 0.313 | 0.231 | 0.244 |
| **Fusion (GMM + Poly 2 SVM)** | 7 | 0.255 | 0.269 | 0.220 | 0.229 |

Then, we want to see if our chosen models are well calibrated for our target application.
To do this we make use of the Bayes error plot.

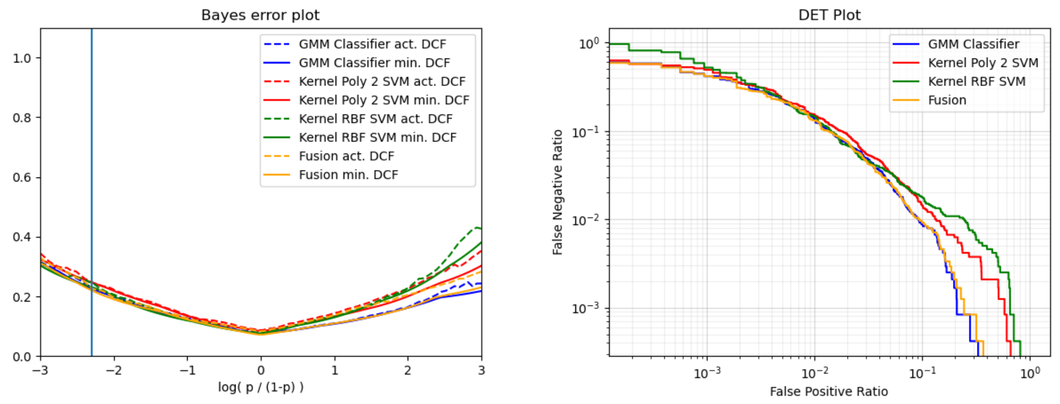


*Figure 18. Bayes error plot and DET plot on evaluation set comparing selected models and their fusion - calibrated*

Again, as in the previous DET plot analysis, we see that the best models are GMM and Fusion, with GMM being slightly better.
The bayes error plot shows good performances for almost all models in left side of the graph, while in the right side only GMM and Fusion still behave well, having worst results for the two kernel SVMs.

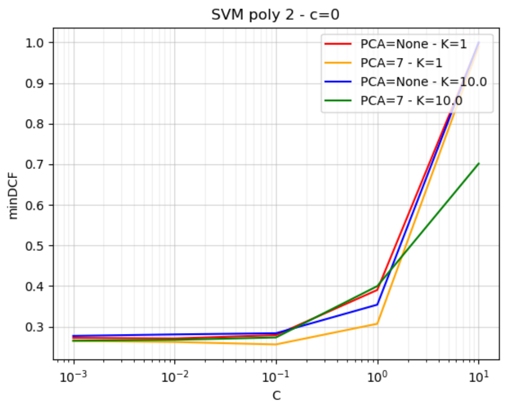
## Previous choices analysis

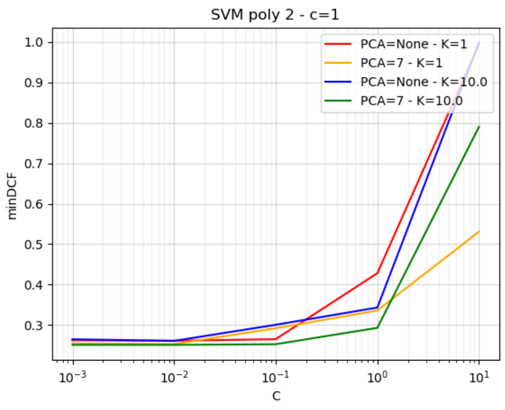
We now evaluate our prior decisions (chosen models, hyper-parameters) to see if we got the best or nearly best results.
We do not explicitly include these models but instead evaluate them within the context of GMMs since Full Covariance and Diagonal (Naive Bayes) Gaussian models (1-F - 1-F and 1-D - 1-D) are both particular instances of our GMMs (but not the Tied Gaussian models).

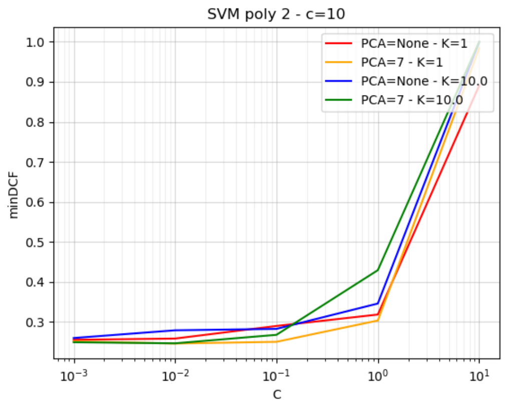
As evaluation metric we use minimum detection cost function (minDCF).

### Polynomial SVM

We first start considering the polynomial SVM of degree 2.







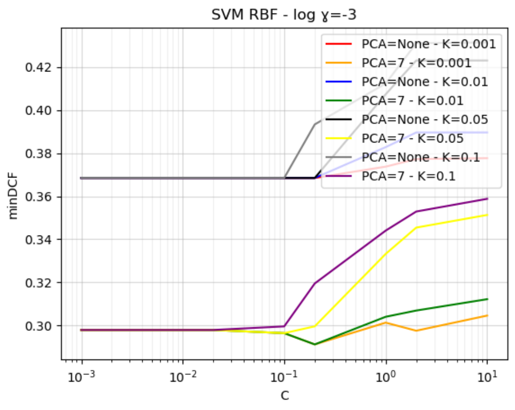
**Model configuration**

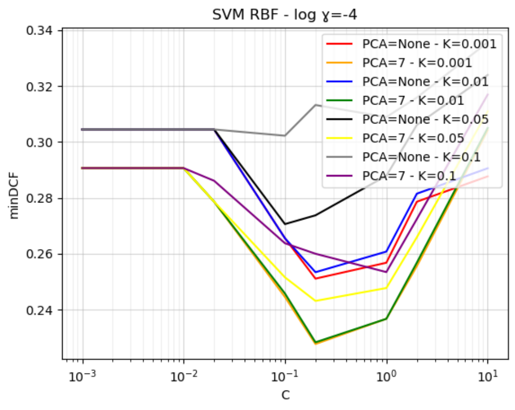
| **Step** | c | C | K | **PCA** | **minDCF** |
|---|---|---|---|---|---|
| **Validation** | 10 | $10^{-2}$ | 10 | 7 | 0.261 |
| **Evaluation** | 0 | $10^{-1}$ | 1 | 7 | 0.256 |
| **Evaluation** | 1 | $10^{-2}$ | 10 | 7 | 0.251 |
| **Evaluation** | 10 | $10^{-2}$ | 10 | 7 | 0.247 |

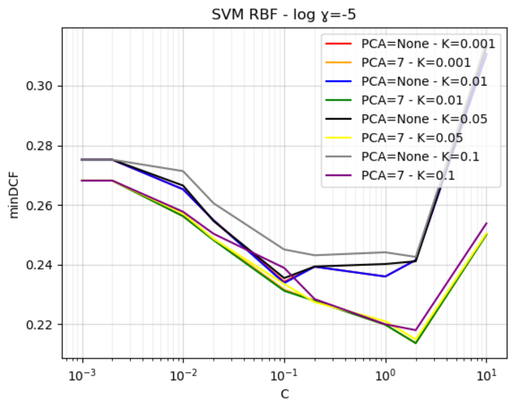
The best configuration found during the training step coincides with the one found here using the evaluation set. In this case we chose the optimal configuration.

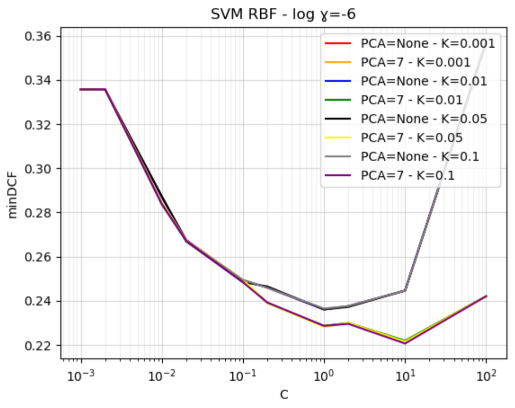
**RBF SVM**

Now, it’s the turn of the RBF kernel SVM.









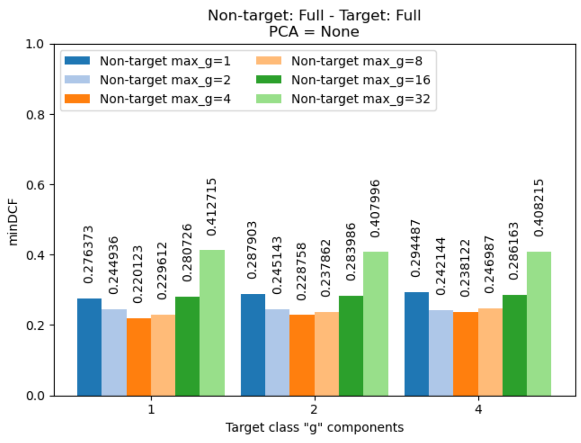
**Model configuration**

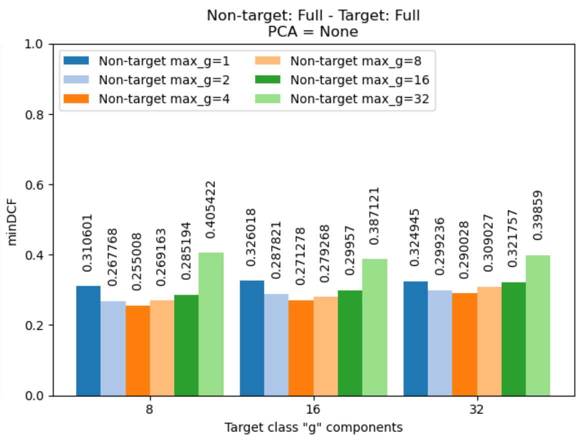
| Step | log γ | C | K | PCA | minDCF |
|---|---|---|---|---|---|
| **Validation** | -5 | $10^{-1}$ | $10^{-2}$ | 7 | 0.264 |
| **Evaluation** | -3 | 0.2 | $10^{-2}$ | 7 | 0.291 |
| **Evaluation** | -4 | 0.2 | $10^{-3}$ | 7 | 0.227 |
| **Evaluation** | -5 | 2 | $10^{-2}$ | 7 | 0.214 |
| **Evaluation** | -6 | 10 | $10^{-1}$ | 7 | 0.221 |

Here we can see that we found a sub-optimal decision, the best configuration differs only for the value of C, moving from 0.1 to 2 taken with the evaluation set.

## GMM

Continuing, we evaluate GMM models. Based on what we've seen before they could deliver a good approximation of our dataset and thus perform really well.





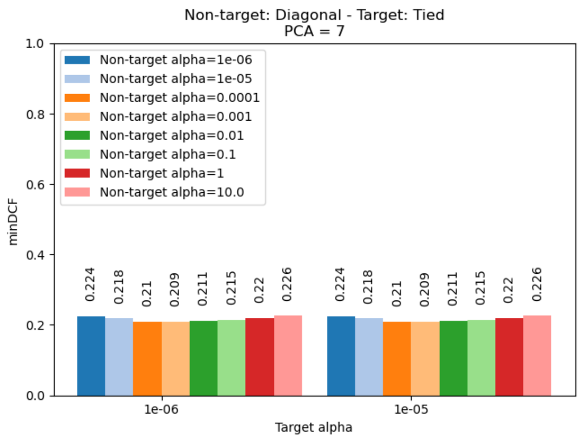
As we did during the grid search in the training part, we evaluate one step at a time. Here we compare different values of g for both target and non-target classes using FullCov and no pre-processing. The PCA evaluation is the next phase.

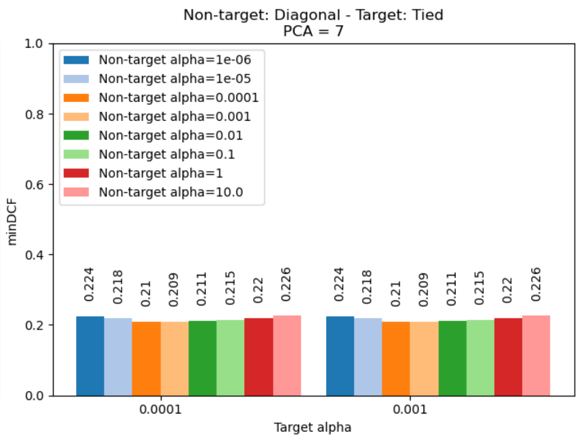
| PCA | Target g | Non-target g | minDCF |
|---|---|---|---|
| **No** | 1 | 4 | 0.220 |
| **10** | 1 | 4 | 0.220 |
| **9** | 1 | 4 | 0.219 |
| **8** | 1 | 4 | 0.218 |
| **7** | 1 | 4 | 0.217 |
| **6** | 1 | 4 | 0.221 |

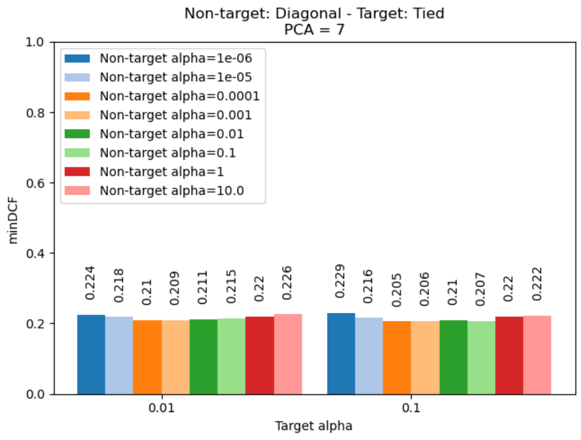
At the moment, each step seems to confirm our previous choices. Moving forward we'll find out more and we'll make the final evaluation.

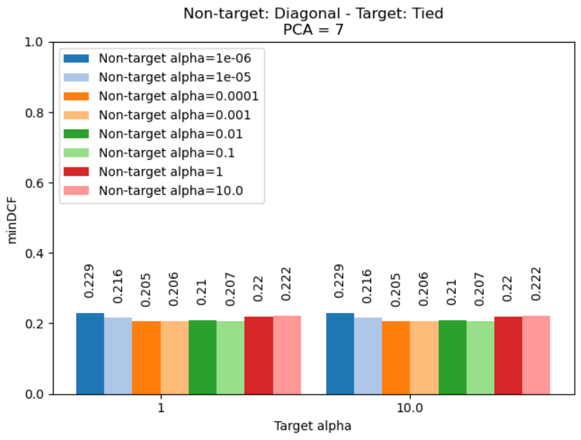
| Model configuration | | | Validation (with K-Folds) | | | Evaluation | | |
|---|---|---|---|---|---|---|---|---|
| **PCA** | **Target sigma** | **Non-Target sigma** | **Target g** | **Non-target g** | **minDCF** | **Target g** | **Non-target g** | **minDCF** |
| **7** | F | F | 1 | 4 | 0.246 | 2 | 4 | 0.211 |
| **7** | F | D | 2 | 8 | 0.243 | 2 | 8 | 0.211 |
| **7** | F | FT | 2 | 4 | 0.245 | 2 | 8 | 0.215 |
| **7** | F | TD | 1 | 16 | 0.235 | 2 | 16 | 0.216 |
| **7** | D | F | 1 | 4 | 0.246 | 16 | 8 | 0.214 |
| **7** | D | D | 2 | 16 | 0.237 | 2 | 32 | 0.210 |
| **7** | D | FT | 2 | 4 | 0.237 | 2 | 16 | 0.217 |
| **7** | D | TD | 1 | 16 | 0.235 | 2 | 16 | 0.216 |
| **7** | FT | F | 4 | 4 | 0.245 | 2 | 4 | 0.208 |
| **7** | FT | D | 2 | 16 | 0.242 | 2 | 32 | 0.207 |
| **7** | FT | FT | 4 | 8 | 0.244 | 2 | 32 | 0.211 |
| **7** | FT | TD | 1 | 16 | 0.235 | 4 | 16 | 0.215 |
| **7** | TD | F | 8 | 4 | 0.235 | 8 | 8 | 0.215 |
| **7** | TD | D | 2 | 16 | 0.241 | 8 | 16 | 0.210 |
| **7** | TD | FT | 4 | 8 | 0.241 | 2 | 16 | 0.213 |
| **7** | TD | TD | 16 | 8 | 0.229 | 8 | 16 | 0.215 |

Here we can see that the best configuration is not the one we previously chose, but we go deeper and try to find the best value of alpha of the current best configuration.









The combination alphas that give us the best minDCF is 0.1 and $10^{-4}$ for the target and non-target classes, respectively, with a value of minDCF of 0.205.

Summarizing the configuration that resulted to be the best for this problem is:

- Target class: Full-Tied, alpha = 0.1, g = 2
- Non-Target class: Diagonal, alpha = $10^{-4}$, g = 32

The PCA dimensions used are 7 and the final minDCF is 0.205.

The configuration we chose, instead, is:

- Target class: Tied-Diagonal, alpha = 0.1, g = 16
- Non-Target class: Tied-Diagonal, alpha = 0.1, g = 8

Also, in this case we used 7 dimensions in PCA, but with a minDCF value of 0.220 on the evaluation set.

Again, we found a sub-optimal model configuration.

**Fusion**

Finally, we evaluate the fusion by comparing our choices with the fusion of the optimal solutions.

| Model | PCA | minDCF | actDCF |
|---|---|---|---|
| **Fusion (GMM + Poly 2 SVM)** Sub-optimal config. | 7 | 0.220 | 0.229 |
| **Fusion (GMM + Poly 2 SVM)** Optimal config. | 7 | 0.211 | 0.213 |

Even for the fusion, we have a sub-optimal configuration.

**Conclusion**

| Model | PCA | minDCF sub-optimal | minDCF optimal |
|---|---|---|---|
| **GMM** | 7 | 0.226 | 0.205 |
| **Poly 2 SVM** | 7 | 0.247 | 0.247 |
| **RBF SVM** | 7 | 0.231 | 0.214 |
| **Fusion (GMM + Poly 2 SVM)** | 7 | 0.220 | 0.211 |

In the end, the two models that performed significantly better were the GMM and the fusion of GMM and Poly SVM. For the GMM, it's easy to see how the particular nature of the problem, with multiple subclasses, favours this model.
However, we found sub-optimal solutions for both models, with a particular difference for the GMM (0.220 vs. 0.205).
The evaluation and validation populations could really be similar, as our results were almost identical between the two (although slightly better on the evaluation one).