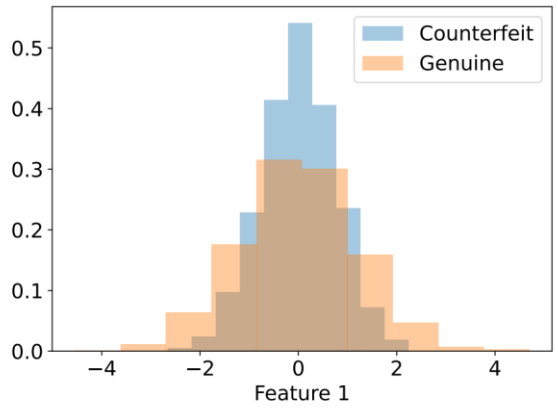
Stefano Fumero s323195 11/09/2024

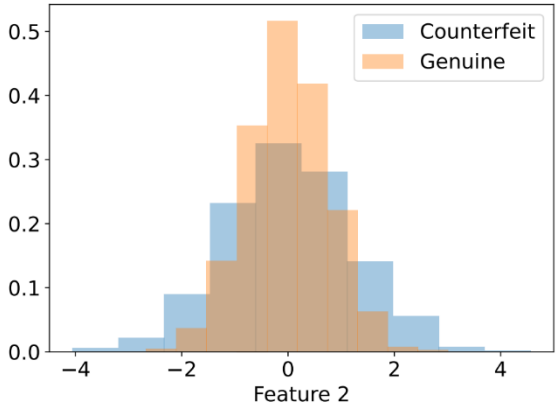
Machine learning and pattern recognition

Project report

The project involves solving a binary classification challenge, where the objective is to detect fingerprint spoofing. Specifically, the task is to distinguish between authentic and fake fingerprint images. The dataset provided contains labeled examples, with the labels indicating whether a fingerprint is genuine (True, label 1) or counterfeit (False, label 0). The data for each sample is represented using six features, which have been extracted to capture the high-level characteristics of the fingerprint images.

-DATA VISUALIZATION

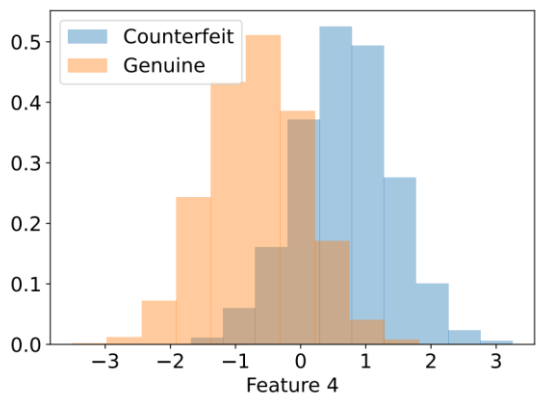


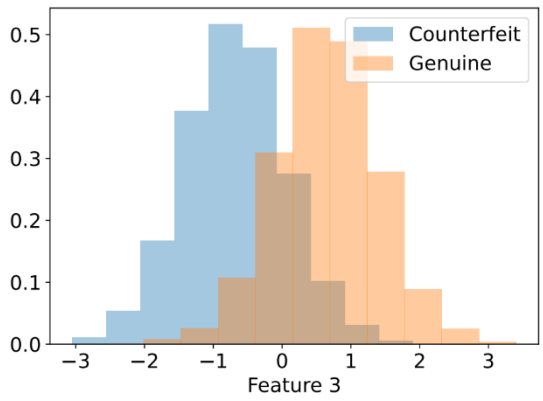


In the two histograms above, we can observe how data are distributed for the first two features, for the two classes. As we can see, there's a large overlap between the two classes for both features, in particular for values between -2 and 2. We can also observe that 9 modes (peaks) are evident from both histograms. Related to the large overlap mentioned before, this is strictly related to the similar mean of the two classes and the 'bell' shape evident from the pictures.

	Feature 1-class 0	Feature 1-class 1	Feature 2-class0	Feature 2-class 1
Mean	0.00287744	0.0005445478	0.01869316	-0.0085243739

	Feature 1-class 0	Feature 2-class 0	Feature 1-class1	Feature 2-class 1
Variance	0.56958105	1.42086571	1.43023345	0.57827792

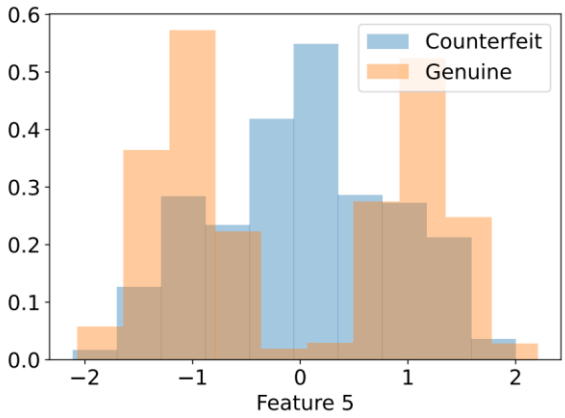


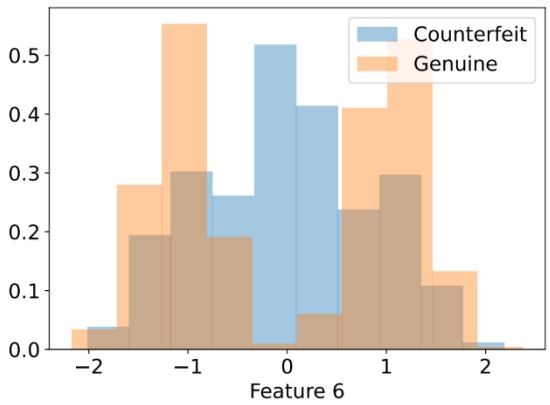


	Feature 3-class 0	Feature 3-class 1	Feature 4-class0	Feature 4-class 1
Mean	-0.68094016	0.665237846	0.01869316	0.6708362

	Feature 3-class 0	Feature 4-class 0	Feature 3-class1	Feature 4-class 1
Variance	0.54997702	0.53604266	0.5489026	0.55334275

If we repeat the same analysis for feature 3 and 4, we observe a minor overlap because the mean for the two classes is different in this case, while the variance is approximately the same. However, classes overlap in both cases between -1 and 1.

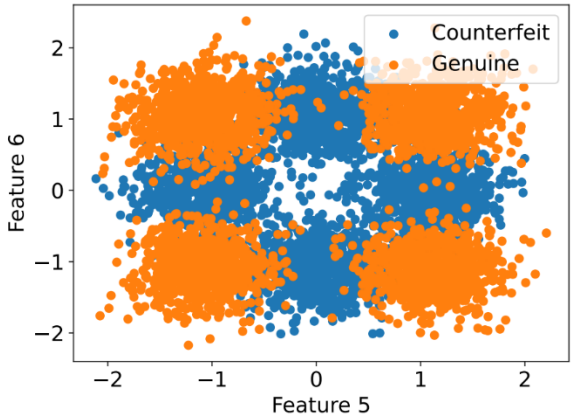




For what concerns features 5 and 6, the distribution of the genuine class changes in this case. In this case it has not a bell shape anymore like in the previous case, we will see what it will cause when trying to fit a uni-variate Gaussing model. The overlap in this case is visible from the histograms as in the two previous cases.

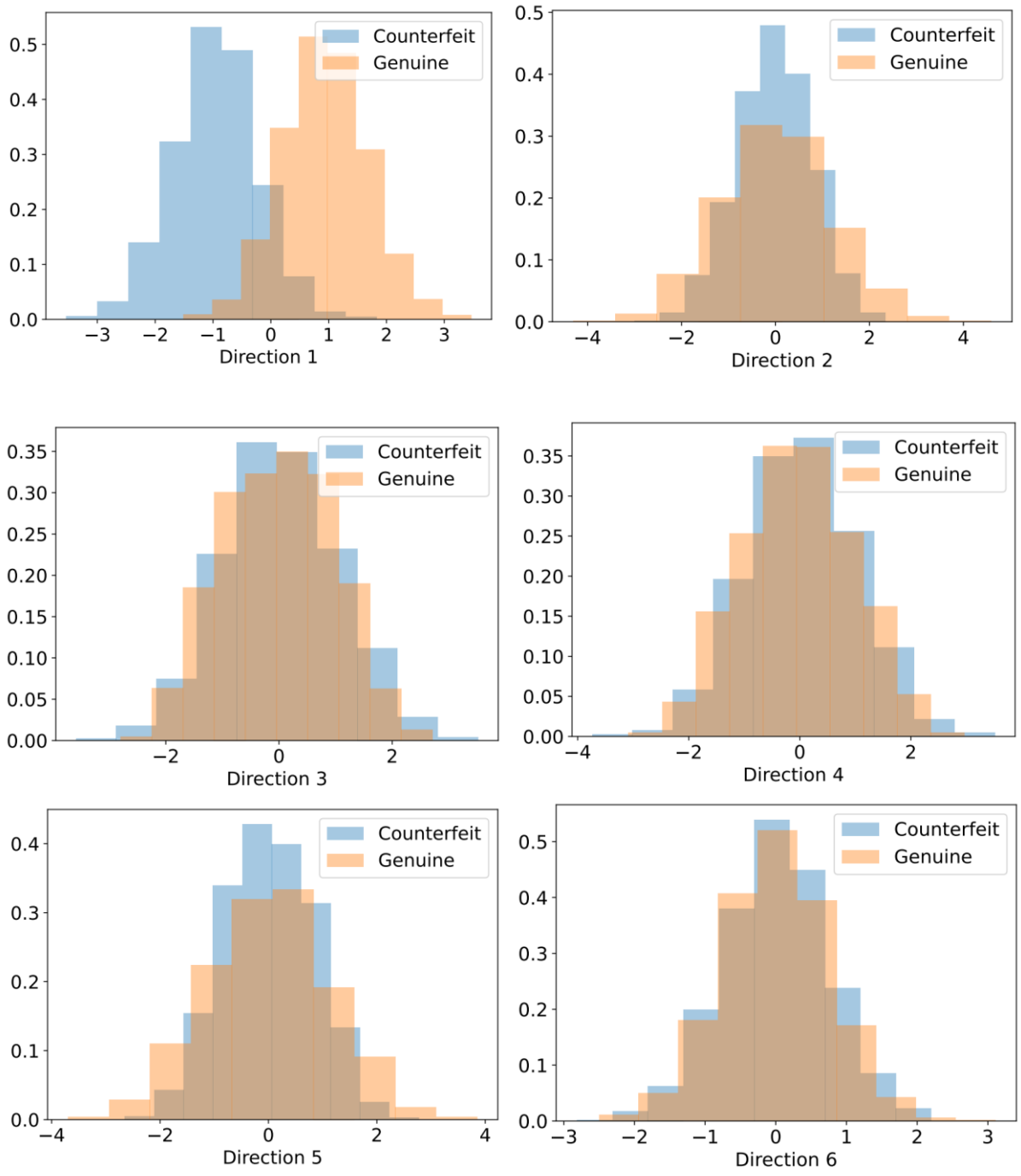
	Feature 5-class 0	Feature 5-class 1	Feature 6-class0	Feature 6-class 1
Mean	0.02795697	-0.0417251858	0.01869316	0.0239384879

	Feature 5-class 0	Feature 5-class 1	Feature 6-class 0	Feature 6-class 1
Variance	0.6800736	1.31776792	0.70503844	1.28702609



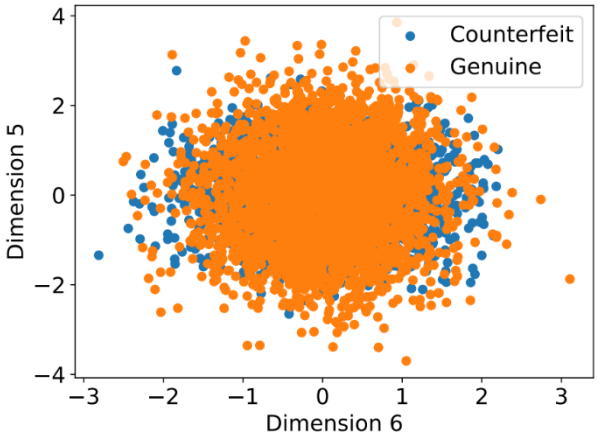
For completeness, also the scatter plot for the pair feature 5 and feature 6 has been reported. As we can spot from the image, there are 4 clusters for each class.

-PCA AND LDA (DIMENSIONALITY REDUCTION TECHNIQUES)

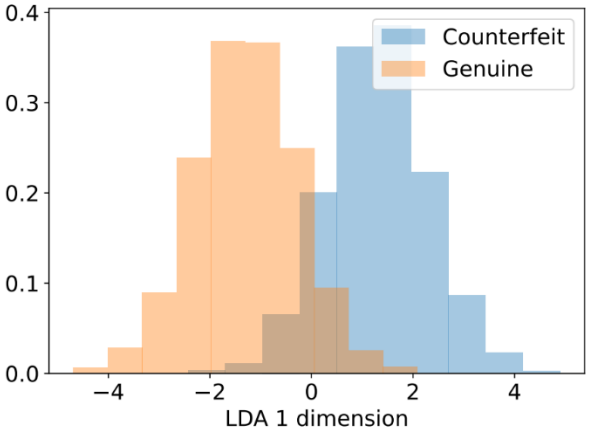


This is the result of PCA applied on the training dataset keeping 6 PCA directions. In this case, we are not effectively performing dimensionality reduction, the number of feature is the same as it was at the beginning. We are basically performing a linear transformation of the data through PCA.
For the first feature, we may observe that classes are much more separated than before. This happens because PCA reorganizes features in such a way that the maximum variance is captured by the first

principal components. As a result, the variations between classes might be amplified along these new directions, making the classes more easily separable in the space of principal components.



I report the scatter plot for the same features I have reported before as a comparison to show that, after PCA has been applied, clusters cannot be spotted easily as before because of the linear transformation performed by PCA (they have been projected in another direction).



Also LDA has been applied, but in this case to reduce the dimensionality from 6 to 1 (since we only have two classes). If we compare this histogram with the ones of the first section, we observe that LDA is finding a direction that allows us to effectively separate classes defining a threshold that is approximately close to zero.As we can see from the histogram, classes overlap betweem -2 and 2, but the overlapping region is much less significant than the one obtained plotting original data

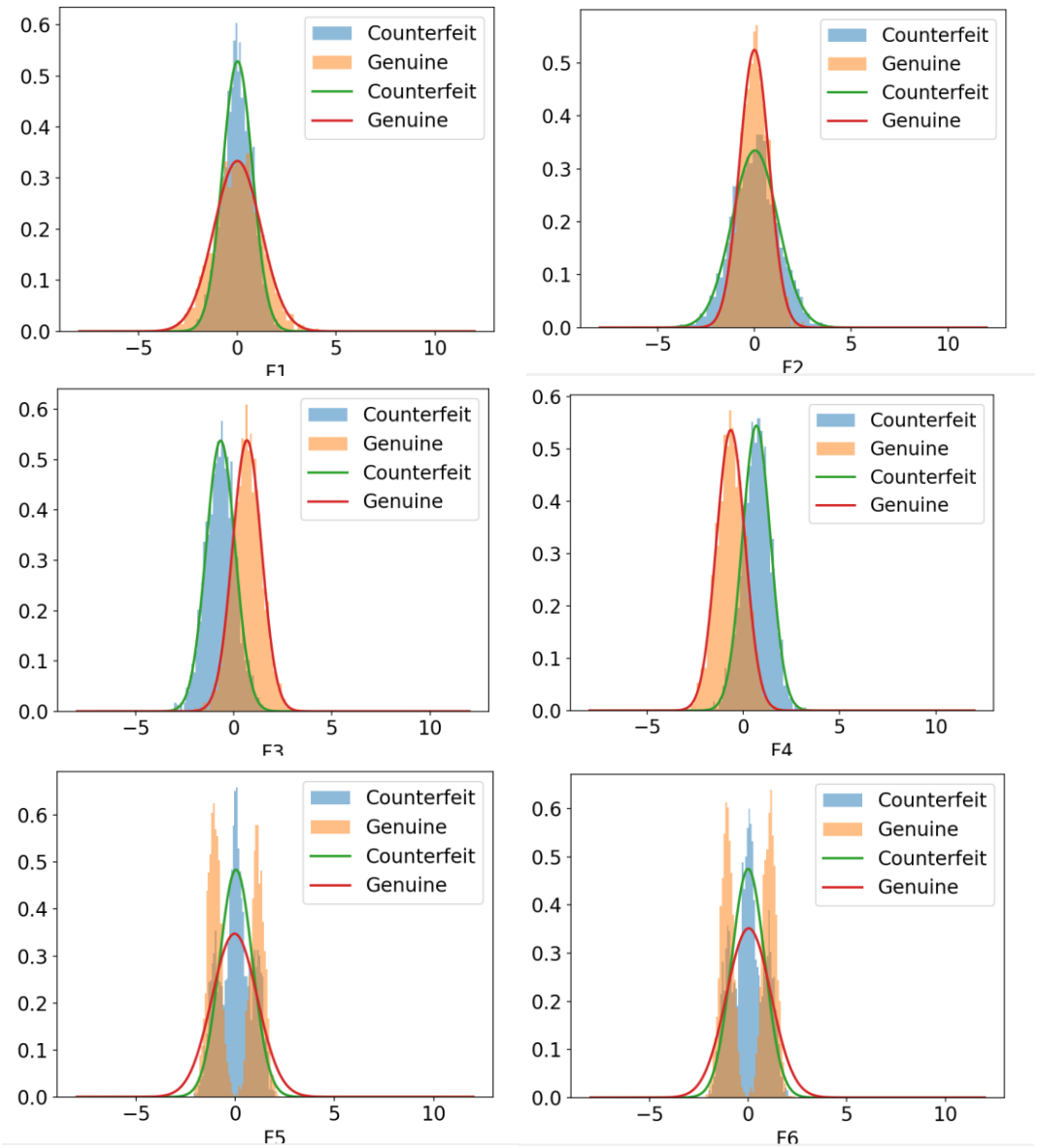
If we apply LDA for classification, setting the threshold as $\frac{\mu_1+\mu_2}{2}$ and choosing the orientation that results in the projected mean of the second class being larger than the projected mean of the first class, we obtain an error rate of 9.3%, with 186 errors out of 2000 samples.
This threshold has been introduced for simplicity and, since we are not yet get to the definition of the optimial threshold, I tried to vary it between -2 and 2 (taking 100000 points in the previous range) to see if we could improve the error rate. I obtained, as best result, an error rate of approximately 9.05% with a threshold equal to -0.037068. This confirms that the threshold we're using is not the best one yet.
LDA assumes Gaussian-distributed noise, when the number of directions is large the within class covariance matrix is singular or close to singular. It could thus be beneficial to pre-process our data with PCA before applying LDA. This is the result we obtain:

	m=5	**m=4**	**m=3**	**m=2**
Error rate (%)	9.3	9.2	9.2	9.2

As we can observe, we get an improvement for m=2,3,4. Since the dimensionality of our data is not that high, it changes only of 0.1%, but it's better anyway.

-GAUSSIAN DENSITY ESTIMATION

In this part, we aim to fit uni-variate Gaussian models to the various features of each class in the dataset. For each class, and for each feature within that class, we'll calculate the Maximum Likelihood (ML) estimates of the parameters for a 1-dimensional Gaussian distribution.



As we can observe in the histograms above, the Gaussian distribution fits for the first 4 features, while it does not for the last two. Indeed, when modeling continuous values, Gaussian distributions arises naturally in a wide variety of contexts, in many cases we observe that data histograms present Gaussian-like shapes. As we have already stated in the data visualization section, the last two features didn't present this kind of shape, thus we obtained the result we expected.

-GENERATIVE MODELS (MVG,NAIVE BAYES AND TIED GAUSSIAN CLASSIFIERS)

Error rate for Gaussian classifiers on the binary classification problem:

Classifier	No PCA	PCA (m=6)	PCA (m=5)	PCA (m=4)	PCA (m=3)	PCA (m=2)	PCA (m=1)
MVG	7.0%	7.0%	7.1%	8.1%	8.8%	8.8%	9.2%
Tied Gaussian	9.3%	9.3%	9.3%	9.2%	9.2%	9.2%	9.3%
Naive Bayes	7.2%	8.9%	8.8%	8.8%	9.0%	8.8%	9.2%

As we can observe, MVG is the model that performs better overall, while the Tied Gaussian classifier reports an error rate that is equal to the one we obtained using LDA for classification.

Correlation for class 0:

1.00	0.00055	0.0327	0.0337	0.0198	-0.021
0.00055	1.000	-0.0178	-0.0179	-0.0264	0.023
0.0327	-0.0178	1.000	-0.00333	-0.0110	0.0271
0.0337	-0.0179	-0.00333	1.000	0.00855	0.0223
0.0198	-0.0264	-0.0110	0.00855	1.000	0.0229
-0.021	0.023	0.0271	0.0223	0.0229	1.000

Correlation for class 1:

1.000	-0.0164	0.00620	0.0173	0.0139	-0.000128
-0.0164	1.000	-0.0202	-0.0161	-0.0170	0.0192
0.00620	-0.0202	1.000	0.0490	-0.00436	-0.0171
0.0173	-0.0161	0.0490	1.000	-0.0134	0.0406
0.0139	-0.0170	-0.00436	-0.0134	1.000	0.0128
-0.000128	0.0192	-0.0171	0.0406	0.0128	1.000

Looking at the correlation matrixes for the two classes can help us to better understand results obtained with the three different classifiers. As we expected, unconstrained MVG is the model that performs better since we have a large number of samples with respect to the number of features, which means that we are able to properly estimate the mean and the covariance matrix for each class with the ML approach. The same goes for the Naive Bayes Gaussian classifier, which has a performance only slightly worse with respect to MVG. We expect this result if the Naive Bayes assumption is verified, which means that features can be considered approximately indipendent. The correlation matrix has diagonal elements equal to 1, whereas out-of-diagonal elements correspond to the correlation coefficients for all feature pairs. In this case we have small values, which means that the correlation is really low.
On the other hand, the Tied Gaussian classifier is the one which provides worst results, and this happens when the distribution of the two classes is different (the two covariance matrixes are different).
We have also tried to apply PCA to reduce feature dimensionality to simplify the estimate (removing dimensions with low variance), but, as we can observe, it didn't improve classification performances. Indeed, in this case, the number of dimensions of raw data is not really high, they are only 6. As we can observe, in the case of the Naive Bayes classifier, the error rate increases when we apply PCA, this happens because, since PCA rotates data and Naive Bayes classifier compute a covariance matrix parallel to axis, it is possible that the transformation makes things worst.

Error rate for the classification keeping only the first 4 features:
Error rate for MVG:
Number of erros: 159 (out of 2000 samples)
Error rate: 8.0%
Error rate for tied Gaussian Model:
Number of erros: 190 (out of 2000 samples)
Error rate: 9.5%
Error rate for Naive Bayes classifier:
Number of erros: 153 (out of 2000 samples)
Error rate: 7.6%

Error rate for the classification keeping only feature 1 and 2:
Error rate for MVG:
Number of erros: 730 (out of 2000 samples)
Error rate: 36.5%
Error rate for tied Gaussian Model:
Number of erros: 989 (out of 2000 samples)
Error rate: 49.5%

Error rate for the classification keeping only features 3 and 4:
Error rate for MVG:
Number of erros: 189 (out of 2000 samples)
Error rate: 9.4%
Error rate for tied Gaussian Model:
Number of erros: 188 (out of 2000 samples)
Error rate: 9.4%

First of all, we observe that performances get worse when we reduce the number of features, suggesting that the model performs better when all the six features are available.
When we limit the analysis to only features 1 and 2, the MVG model records a significantly lower error rate compared to the Tied Gaussian model. This is consistent with the characteristics of features 1 and 2: the variances differ between the classes, while their means are similar. The Tied Gaussian model, which assumes equal variances between the classes, is clearly penalized in this scenario. In contrast, the MVG model, which allows for different variances, is better suited to capture these differences, although the error rate remains high, suggesting that these two features alone are not sufficient for accurate classification. For the third and fourth features, we notice that both models achieve almost the same error rate. This is consistent with the characteristics of features 3 and 4, where the class means differ but the variances are similar. In this case, the Tied Gaussian model is not penalized because its assumption of equal variances is valid, while the MVG model does not gain any advantage from having greater flexibility. This demonstrates how the Tied Gaussian model can be effective when the data characteristics align with its assumptions.

-BAYES DECISIONS AND MODEL EVALUATION

We consider 5 different applications in terms of prior,Cfn and Cfp (pi,Cfn,Cfp):

1. (0.5,1.0,1.0)
2. (0.9,1.0,1.0)
3. (0.1,1.0,1.0)
4. (0.5,1.0,9.0)
5. (0.5,9.0,1.0)

Representing the applications in terms of effective priors:

$$\pi *= \frac{\pi C_{fn}}{\pi C_{fn} + (1-\pi) C_{fp}}$$

I evaluated, for each application and for each classifier, the normalized DCF and the minimum DCF (two couples among the previous five applications lead to the same effective prior, I reported all the results for completeness, even if It is clear that they are the same):

MVG classifier:

Effective prior	Normalized DCF	Min DCF
0.5	0.139929	0.130168
0.9	0.400058	0.342310
0.1	0.305140	0.262913
0.1	0.305140	0.262913
0.9	0.400058	0.342310

Tied Gaussian classifier:

Effective prior	Normalized DCF	Min DCF
0.5	0.186028	0.181244
0.9	0.462558	0.442108
0.1	0.406058	0.362839
0.1	0.406058	0.362839
0.9	0.462558	0.442108

Naive Bayes classifier:

Effective prior	Normalized DCF	Min DCF
0.5	0.143929	0.131128
0.9	0.389257	0.350950
0.1	0.302163	0.256960
0.1	0.302163	0.256960
0.9	0.389257	0.350950

We can observe that, when considering a uniform prior and equal costs for false negatives and false positives, the MVG classifier demonstrates its best performance, as reflected in the lowest normalized and minimum Detection Cost Function values. This suggests that the MVG classifier is well-suited for applications where the likelihood of encountering legitimate and impostor users is balanced, and the consequences of misclassification are equally weighted.

As the prior probability of encountering legitimate users increases (π = 0.9), the performance of the MVG classifier degrades, with a noticeable increase in both normalized and minimum DCF. This indicates that

when the model is biased towards expecting more legitimate users, it becomes less effective at distinguishing between legitimate and impostor cases, likely due to an increased tendency to accept impostors. Conversely, when the prior probability favors impostors ($\pi = 0.1$), the classifier's performance is somewhat better than in the previous case but still worse than under a uniform prior. This reflects the model's improved ability to reject impostors.
The scenarios where the costs of misclassification are asymmetric—either with a higher cost for false positives (stronger security) or for false negatives (ease of use)—further illustrate the sensitivity of the MVG classifier to these parameters. When the cost of accepting a fake image is higher, the classifier's performance resembles that of a scenario with a lower prior probability for legitimate users, as the model becomes more cautious in granting access. Conversely, when the cost of rejecting a legitimate image is higher, the model behaves similarly to when the prior for legitimate users is increased, showing a higher tendency to accept users, even at the risk of allowing impostors.
The Tied Gaussian classifier generally performs worse than the MVG, with higher normalized and minimum DCF values across all scenarios, indicating that tying the covariance matrices in this model results in less flexibility and reduced effectiveness in adapting to varying prior probabilities and costs.
The Naive Bayes classifier, on the other hand, shows performance closer to the MVG, particularly under a uniform prior, but it struggles as the prior probabilities become more extreme or as the costs become asymmetric.
In summary, the MVG classifier proves to be the most robust across the different applications, especially when the prior probabilities and costs are balanced. However, its performance declines as the scenarios deviate from this balance.

Then, I applied PCA reducing the feature subspace from 6 to 1 evaluating three applications with effective priors: 0.1,0.5,0.9. Here are the results:

Application with effective prior 0.1, actual DCF:

Classifier	No PCA	m=6	m=5	m=4	m=3	m=2	m=1
MVG	0.305140	0.305140	0.304147	0.352903	0.387913	0.387913	0.396985
Tied Gaussian	0.406058	0.406058	0.405066	0.403082	0.408186	0.395993	0.402090
Naive Bayes	0.302163	0.392025	0.393017	0.396985	0.395001	0.386921	0.396985

Application with effective prior 0.1, min DCF:

Classifier	No PCA	m=6	m=5	m=4	m=3	m=2	m=1
MVG	0.262913	0.262913	0.273826	0.301187	0.356327	0.352647	0.368648
Tied Gaussia	0.362839	0.362839	0.364823	0.360999	0.368088	0.362983	0.368648
Naive Bayes	0.256960	0.353479	0.354471	0.361415	0.364535	0.356183	0.368648

Miscalibration rate for the application with effective prior 0.1 ($\frac{(DCF-minDCF)100}{minDCF}$) in %

Classifier	No PCA	m=6	m=5	m=4	m=3	m=2	m=1
MVG	16.06	16.06	11.08	17.18	8.87	10.00	7.67
Tied Gaussia	11.87	11.87	11.01	11.58	10.89	9.10	9.02
Naive Bayes	17.58	10.91	10.89	9.85	8.35	8.66	7.72

Application with effective prior 0.5, actual DCF:

Classifier	No PCA	m=6	m=5	m=4	m=3	m=2	m=1
MVG	0.139929	0.139929	0.141913	0.160938	0.175899	0.175995	0.185004
Tied Gaussian	0.186028	0.186028	0.186044	0.185020	0.185004	0.185036	0.187020
Naive Bayes	0.143929	0.177995	0.174987	0.176971	0.179932	0.176971	0.185004

Application with effective prior 0.5, min DCF:

Classifier	No PCA	m=6	m=5	m=4	m=3	m=2	m=1
MVG	0.130168	0.130168	0.133145	0.153722	0.173435	0.173083	0.176907
Tied Gaussia	0.181244	0.181244	0.181164	0.182108	0.183020	0.178859	0.176907
Naive Bayes	0.131128	0.172699	0.173691	0.171675	0.174619	0.171019	0.176907

Miscalibration rate for the application with effective prior 0.5 ($\frac{(DCF-minDCF)100}{minDCF}$) in %

Classifier	No PCA	m=6	m=5	m=4	m=3	m=2	m=1
MVG	7.50	7.5	6.59	4.70	1.42	1.68	4.58
Tied Gaussia	2.7	2.64	2.69	1.60	1.08	3.46	5.62
Naive Bayes	9.76	10.91	10.89	9.85	8.35	8.66	7.72

Application with effective prior 0.9, actual DCF:

Classifier	No PCA	m=6	m=5	m=4	m=3	m=2	m=1
MVG	0.400058	0.400058	0.398041	0.459821	0.468030	0.443404	0.477535
Tied Gaussian	0.462558	0.462558	0.462558	0.461550	0.456509	0.478543	0.480559
Naive Bayes	0.389257	0.451181	0.466014	0.462990	0.459101	0.442396	0.477535

Application with effective prior 0.9, min DCF:

Classifier	No PCA	m=6	m=5	m=4	m=3	m=2	m=1
MVG	0.342310	0.342310	0.351238	0.415035	0.439228	0.438364	0.434188
Tied Gaussia	0.442108	0.442108	0.445132	0.444124	0.434188	0.435196	0.434188
Naive Bayes	0.350950	0.435916	0.434044	0.431308	0.434332	0.432316	0.434188

Miscalibration rate for the application with effective prior 0.9 ($\frac{(DCF-minDCF)100}{minDCF}$) in %

Classifier	No PCA	m=6	m=5	m=4	m=3	m=2	m=1
MVG	6.00	16.85	13.32	10.83	6.57	1.15	9.97
Tied Gaussia	4.6	4.62	3.92	3.93	5.14	9.95	10.70
Naive Bayes	10.9	3.49	7.37	7.33	5.71	2.31	9.97

If we consider score well calibrated if the loss rate is under 15/20%, then we can conclude that all the models and applications produce a score that is well calibrated. Obviously, looking at the tables above, we can understand that there are models that are better calibrated than others, in particular there are cases in which the mis-calibration rate is really low when PCA is applied reducing the feature size to 2 or 3. The configuration that gives the best result overall is the one with effective priore qual to 0.5, which means the case in which samples ara balanced between the two classes and costs are uniform. Indeed, the result is coherent with what we observed in the first section, since data are roughly evenly distributed between the two classes (and thus the same goes for the training dataset used to train the models).

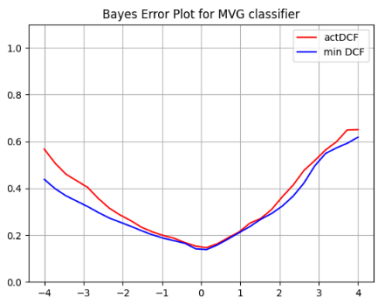
For each application (for each prior) let's analyze what is the best classifier and the best configuration in terms of minDCF:

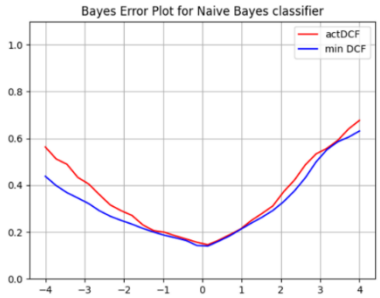
application with prior 0.1: Naive Bayes without PCA (min DCF = 0.256960);

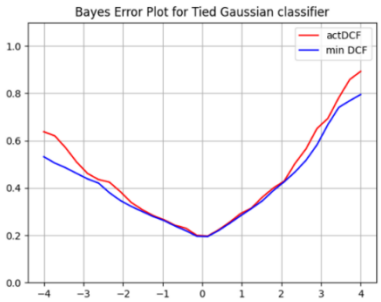
application with prior 0.5: MVG without PCA (min DCF = 0.130168);

application with prior 0.9: MVG without PCA (min DCF = 0.342310).

Then, considering the application with effective prior equal to 0.1 (our main application), I chose the setup that gave the best results in terms of minimum DCF and then I highlighted their relation with the actual DCF through Bayes error plot. Overall, the three best configurations shown to be well calibrated in the considered range (-4,+4) with quite a few differences among them.

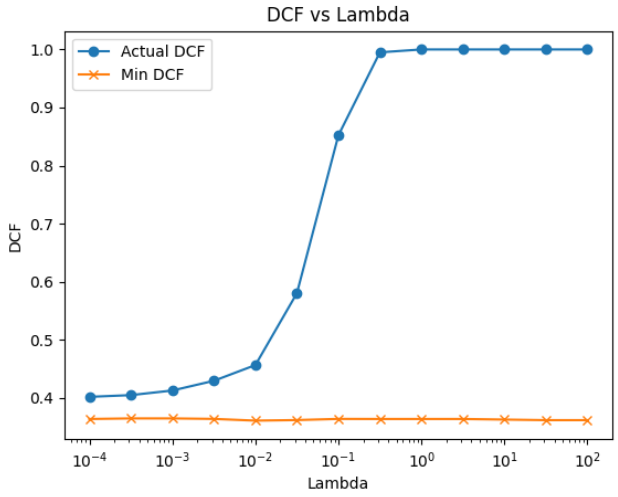




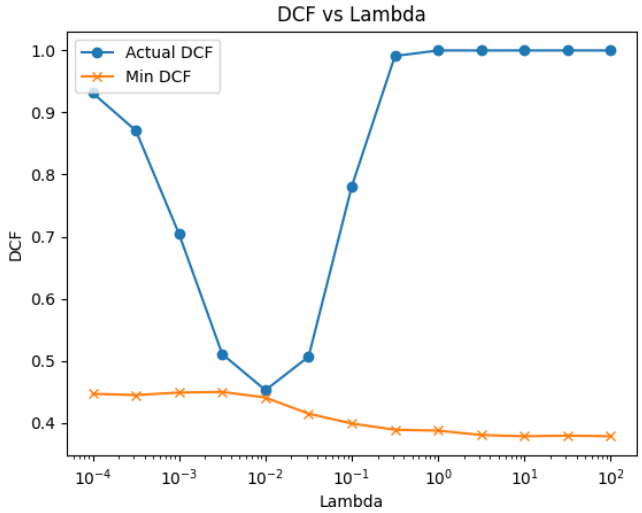


-LOGISTIC REGRESSION

We are analyzing the binary logistic regression model on the project data (TrainData) and we start considering the non-weighted version of the model. The first objective is to evaluate how the regularization coefficient affects min DCF and actual DCF by plotting how it changes for different values of lambda. To obtaina good coverage, I used numpy.logspace(-4, 2, 13) to generate different values of lambda. As we can see from the image below, since we have a large number of samples, regularization seems ineffective, and actually degrades actual DCF since the regularized models tend to lose the probabilistic interpretation of the scores (it tends to 1 and it always increases as lambda increases).

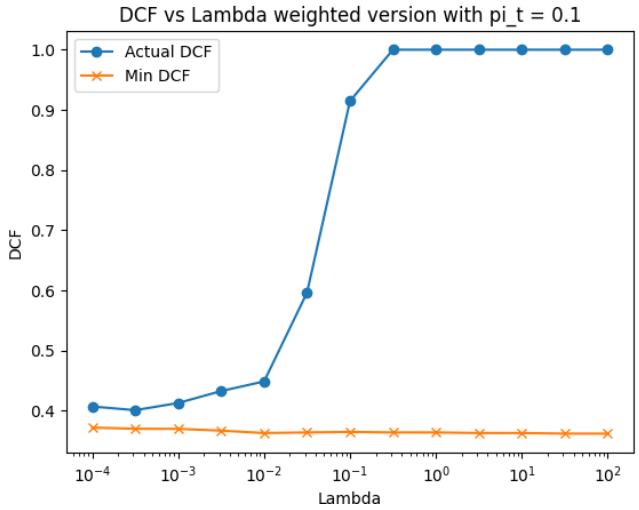


To better understand the role of regularization, we analyze the results that we would obtain if we had fewer training samples (1 out of 50). The result we obtain is the following:



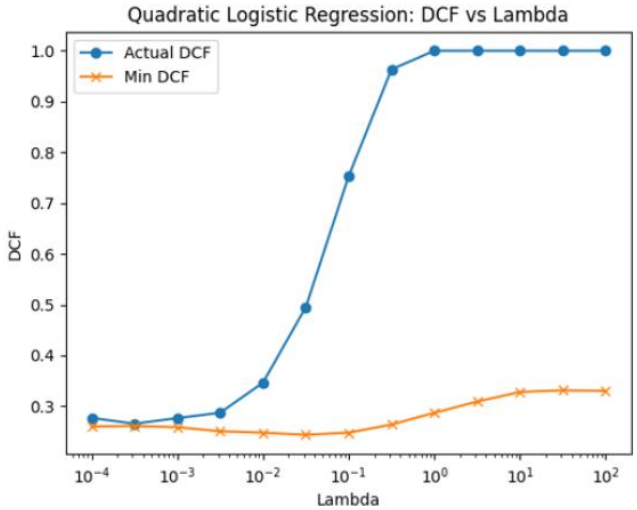
When the number of samples is lower, then the risk of overfitting is higher. In this case the regularization term has a more important role. The risk of generalization when lambda is high is a problem, that's why we observe that the actual DCF is above 0.9. Indeed, the model performs well on training samples, but may have poor classification accuracy for unseen data. The same happens when lambda grows too much, because the solution has small norm but it's not able to properly separate classes. Scores tend to lose their probabilistic interpretation, which means that the theoretical threshold is not the correct one anymore.

In the following we will again consider only the full dataset and we'll repeat the analysis with the prior-weighted version of the model.



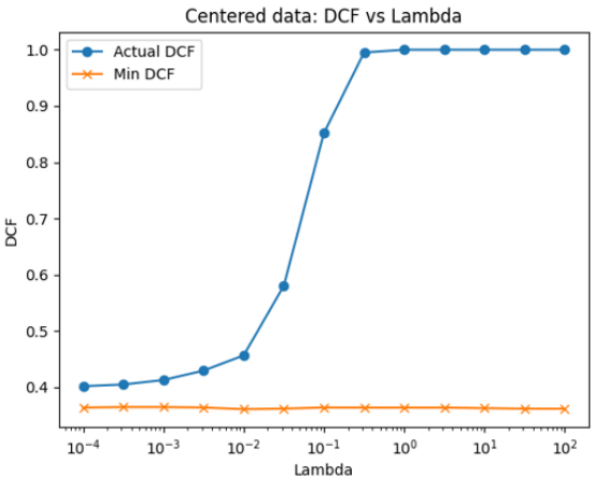
As we can observe, the model is not providing any advantage with respect to the standard one (the one that considers the empirical prior of the training set). As we have already observed in the previous section, data in our dataset are roughly balanced between the two classes, and we expect a similar distribution for our evaluation set, thus fixing a pi_t=0.1 doesn't improve the performances of the classifier.

We now repeat the analysis with the quadratic logistic regression model (again, full dataset only). By expanding the features, we are able to obtain quadratic separation surfaces in the original feature space as we had in the case of non-tied Gaussian classifiers. This is the result we obtained:



This is the model that gives us the best performances in terms of minimum DCF, reaching 0.2436315 in the lowest point of the previous graph. In this case regularization seems to be less effective because, as we can observe, the minimum DCF tends to increase as lambda becomes greater, but it tends to decrease it when it's not too high. Indeed, Quadratic logistic regression introduces greater complexity compared to linear logistic regression, increasing the number of parameters and interactions between variables. Excessive regularization can therefore have a more pronounced effect on the model's ability to fit the data, leading to an increase in the DCF if λ is too high.

On the other hand, the actual DCF decreases as lambda increases only in one point, then it gets always worse, confirming that, also in this case, the best models in terms of minimum DCF are not necessarily those that provide the best actual DCFs (because of score mis-calibration).



For the sake of completeness, we also analyze the effects of centering our data with respect to the dataset mean. Since the original features were already almost standardized, we observe only minor variations in this case.

If we compare the models that we have trained and tested up to this point, as anticipated before, for our main application with pi_t=0.1, the quadratic logistic regression is the one that provides the best result in terms of minimum DCF. The model that performed best up to this point was MVG (refer to the previous sections for details), but, as we know, the model assumes that points of each class can be modeled by a Multivariate Gaussian Distribution. This assumption must be later verified on the data but, as we have already seen in the third section when fitting univariate Guassian distributions, features 5 and 6 do not present a Guassian distribution (a bell shape), which means that also MVG doesn't properly fit for those two features, while it does for the other 4, obtaining overall a good result, but not as good as the quadratic logistic regression. What we can conclude is that the quadratic logistic regression model may better adapt to situations where the classes have different distributions or are unevenly distributed in the feature space.

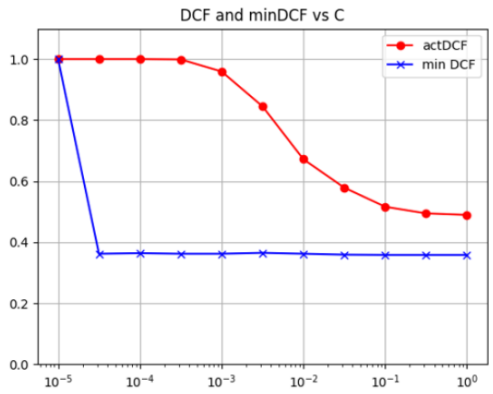
-SUPPORT VECTOR MACHINES

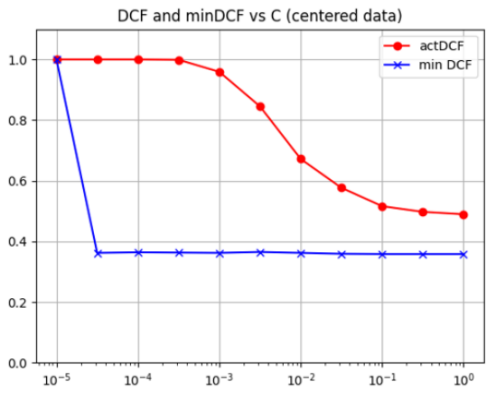
In the following, we apply SVM to the project data. We start from the linear SVM setting K=1.0 with varying values of the regularization parameter C (using numpy.logspace(-5, 0, 11)). As illustrated in the plot of minDCF and actDCF against C, we observe that the regularization coefficient indeed affects the results, although its impact on the two metrics is different.

- **MinDCF**: For small values of C (strong regularization), the minDCF is higher, indicating that the model struggles to find a decision boundary that minimizes the detection cost. As C increases (weaker regularization), minDCF decreases, reaching a plateau, which suggests that the model is better at fitting the data when allowed more flexibility.
- **ActDCF**: Similarly, the actual DCF initially decreases with increasing C, indicating improved performance. However, at a certain point, further increases in C lead to a slight increase in actDCF. This could be attributed to the model starting to overfit the training data, leading to poorer generalization to unseen data.

The scores appear well-calibrated for the target application with a priore qual to 0.1, showing consistency between minDCF and actDCF trends. Overall, the linear SVM demonstrates a performance comparable to other linear models such as logistic regression but is generally outperformed by models that account for non-linear interactions, like the quadratic logistic regression.

Centering the data before applying SVM does not significantly alter the results, confirming that the linear SVM's performance is relatively stable with respect to this preprocessing step.

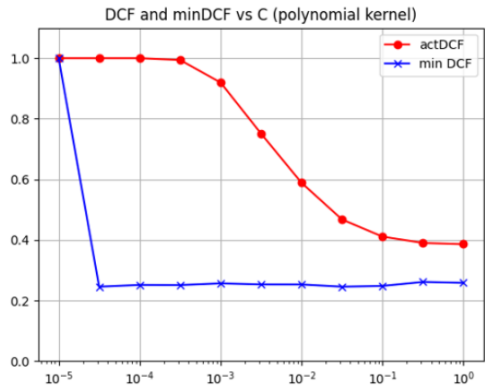




When using a polynomial kernel of degree 2 (with d=2, c=1), the SVM model can capture non-linear relationships between features. As observed in the results, the polynomial SVM generally performs better than the linear SVM in terms of minDCF, reflecting the ability of the quadratic model to better separate classes that are not separable in the original feature space.

However, the actDCF shows more variability with different values of C. This suggests that while the quadratic SVM may achieve better separation in theory (as reflected in minDCF), it can be prone to overfitting, especially when C is too large, leading to less reliable predictions on unseen data (hence the fluctuating actDCF).

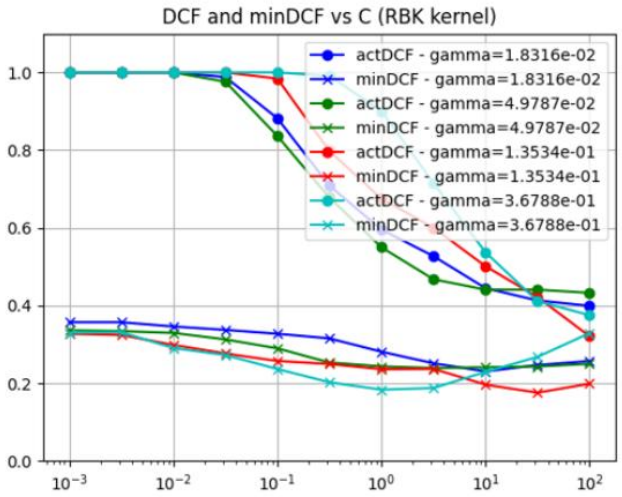
Compared to previous models like logistic regression and MVG, the polynomial SVM is more sensitive to the choice of hyperparameters, but when appropriately tuned, it can offer superior performance in capturing complex patterns within the data.



The RBF kernel SVM introduces an additional hyperparameter γ, which controls the influence of individual training examples. A grid search over different combinations of γ and C was performed, with the results plotted for both minDCF and actDCF.

- **MinDCF**: Certain combinations of γ and C lead to significantly lower minDCF values, indicating that the RBF kernel can capture intricate non-linear relationships within the data, leading to a more optimal decision boundary. This is especially evident for moderate values of γ and C, where the model finds a balance between underfitting and overfitting.
- **ActDCF**: The variability in actDCF across different γ and C values underscores the need for careful tuning. Models with large C and small γ tend to overfit, as reflected in increased actDCF. Conversely, small C or large γ can lead to underfitting, with actDCF also rising. The best-performing models in terms of minDCF may not always correspond to the best actDCF, indicating some degree of miscalibration.

The RBF SVM demonstrates strong performance compared to previous models, particularly in scenarios where the data exhibits complex, non-linear structures. The ability to fine-tune both γ and C provides flexibility, but it also requires extensive experimentation to identify the optimal settings.



The superior performance of the quadratic SVM model can be attributed to the underlying data distribution, as observed in the 'Data Visualization' section. In this earlier analysis, we noted that while some features exhibited overlapping distributions between the two classes, other features had more distinct distributions with different means but similar variances. Specifically, the scatter plots revealed that the data for some feature pairs formed distinct clusters that were not linearly separable.

Given this non-linear distribution, a linear decision boundary, as employed by the linear SVM, is inadequate for effectively separating the classes. The quadratic SVM model, on the other hand, introduces non-linear decision boundaries by transforming the input space into a higher-dimensional space where these clusters become more easily separable.

The effectiveness of the quadratic SVM is especially pronounced in scenarios where the classes overlap but have distinct non-linear boundaries. This matches well with the feature distribution noted during data visualization, where simple linear separations would struggle to capture the complex boundaries between the genuine and fake fingerprints. The quadratic kernel allows the SVM to better capture these non-linear relationships, leading to improved classification performance, as reflected in lower minDCF values compared to the linear model.

This outcome highlights the importance of considering the underlying data distribution when selecting a model. The presence of non-linearities in the feature space, as seen in the data visualization, makes the quadratic model a more appropriate choice, enabling it to outperform the linear model by a considerable margin.

-GMM

In this section, we train full covariance and diagonal models with different number of components for each class (up to 32), computing minimum and actual DCF. Results are reported in the tables below:

Components for the class 0:1, components for class 1 varies from 1 up to 32.

Components Class 1	minDCF / actDCF (Full)	minDCF / actDCF (Diagonal)
1	0.2629 / 0.3051	0.2570 / 0.3022
2	0.2649 / 0.3051	0.2605 / 0.3051
4	0.2139 / 0.2368	0.2098 / 0.2258
8	0.1850 / 0.1957	0.2045 / 0.2207
16	0.1495 / 0.2055	0.1433 / 0.1807
32	0.1847 / 0.2273	0.1373 / 0.1967

Components for the class 0:2, components for class 1 varies from 1 up to 32.

Components Class 1	minDCF / actDCF (Full)	minDCF / actDCF (Diagonal)
1	0.2181 / 0.2317	0.2449 / 0.2716
2	0.2160 / 0.2337	0.2489 / 0.2674
4	0.2234 / 0.2255	0.1990 / 0.2099
8	0.1860 / 0.2133	0.2039 / 0.2089
16	0.1701 / 0.1980	0.1536 / 0.1731
32	0.1857 / 0.2269	0.1436 / 0.1810

Components for the class 0:4, components for class 1 varies from 1 up to 32.

Components Class 1	minDCF / actDCF (Full)	minDCF / actDCF (Diagonal)
1	0.2330 / 0.2458	0.1451 / 0.1501
2	0.2317 / 0.2357	0.1542 / 0.1611
4	0.2161 / 0.2395	0.1481 / 0.1687
8	0.1887 / 0.2064	0.1403 / 0.1515
16	0.1745 / 0.1871	0.1372 / 0.1687
32	0.1867 / 0.2380	0.1412 / 0.1677

Components for the class 0:8, components for class 1 varies from 1 up to 32.

Components Class 1	minDCF / actDCF (Full)	minDCF / actDCF (Diagonal)
1	0.1759 / 0.2004	0.1731 / 0.1763
2	0.1808 / 0.1913	0.1762 / 0.1874
4	0.1959 / 0.1989	0.1585 / 0.1838
8	0.1786 / 0.1928	0.1463 / 0.1809
16	0.1526 / 0.1725	0.1324 / 0.1487
32	0.1755 / 0.1903	0.1312 / 0.1517

Components for the class 0:16, components for class 1 varies from 1 up to 32.

Components Class 1	minDCF / actDCF (Full)	minDCF / actDCF (Diagonal)
1	0.1670 / 0.1783	0.2069 / 0.2217
2	0.1657 / 0.1753	0.2039 / 0.2217
4	0.1918 / 0.1918	0.1963 / 0.2050
8	0.1755 / 0.2050	0.1979 / 0.2140
16	0.1631 / 0.1766	0.1622 / 0.1769
32	0.1752 / 0.1955	0.1644 / 0.1799

Components for the class 0:32, components for class 1 varies from 1 up to 32.

Components Class 1	minDCF / actDCF (Full)	minDCF / actDCF (Diagonal)
1	0.2570 / 0.2580	0.1748 / 0.1964
2	0.2560 / 0.2600	0.1748 / 0.1974
4	0.2460 / 0.2825	0.1929 / 0.1979
8	0.2191 / 0.2251	0.1929 / 0.1989
16	0.1938 / 0.2180	0.1850 / 0.2069
32	0.2337 / 0.2499	0.1766 / 0.1989

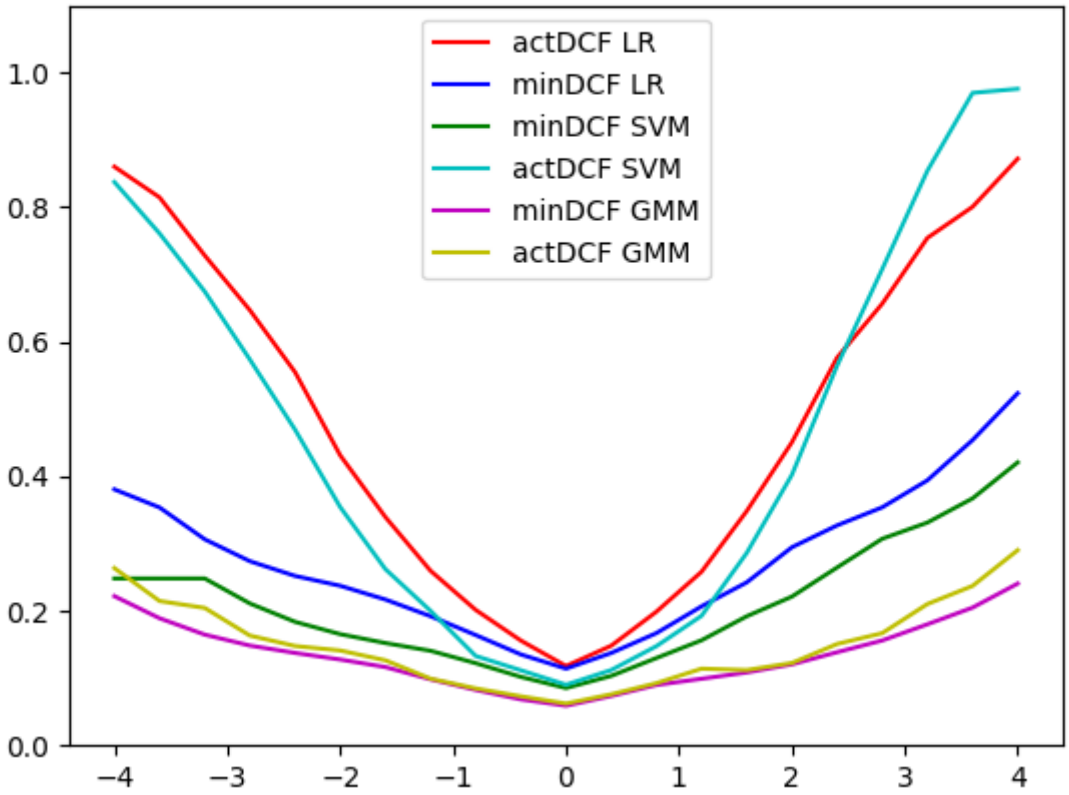
Based on the results reported for GMM models with different numbers of components and both full and diagonal covariance matrices, we can observe some clear trends. Generally, diagonal covariance models offer comparable, if not better, results in certain configurations than full covariance models. For instance, with 32 components for class 1 and 1 component for class 0, the GMM with diagonal covariance reported a minDCF/actDCF of 0.1748 / 0.1964, compared to 0.2570 / 0.2580 for the full covariance model. This surprising result might be attributed to the reduced complexity of diagonal covariance models, which fit better when sample sizes are limited or features are weakly correlated, as observed in the correlation matrix.

Regarding combinations that work best, the strongest performances were observed with higher numbers of components for class 1 compared to class 0. For example, GMMs with 16 components for class 1 and 1 component for class 0 showed significantly better results, with diagonal covariance achieving a minDCF of 0.1433 compared to 0.1495 for full covariance. This supports the idea that more components for the more complex class (fake or genuine) allow the model to better capture the variability within the data.

One surprising result was that in some configurations, such as 16 components for both classes, diagonal covariance outperformed full covariance. This is contrary to expectations, as full covariance, being more flexible, was expected to perform better. However, the data reveal that many of the features exhibit low correlation, making diagonal covariance models less prone to overfitting and better at managing reduced complexity.

When comparing GMMs to other classifiers such as SVM and logistic regression, we see distinct advantages. The best performing model among the GMMs was the one with 16 components for class 1 and 8 components for class 0, using diagonal covariance, which reached a minDCF of 0.1324 and an actual DCF equal to 0.1487. In comparison, the best model for SVM was the quadratic SVM with a RBF kernel, with gamma=0.1353352832366127 and C=32.622776601683793, which achieved superior classification by effectively capturing non-linear patterns (actual DCF=0.4276 and minDCF=0.1773). Meanwhile, for logistic regression, the best performance was obtained using quadratic logistic regression, with the regularization parameter λ=3.16227766e-02, reaching a minimum DCF of 0.2436 and an actual DCF = 0.4972. Although the quadratic models (both for SVM and logistic regression) provided good performances, GMM with diagonal covariance and 16 components for class 1 outperformed them in terms of minDCF for this application.

As we can observe in Bayes error plot below, the relative ranking of the systems is preserved for all the operating points both for what concerns actual and minimum DCF. Moreover, while the GMM seems to be calibrated for most of the operating points, while SVM and quadratic logistic regression show significant miscalibration, with the actual DCF tending to 1 at the extremes of the Bayes error plots, confirming that they're harmful for some applications.



-SCORE CALIBRATION AND FUSION

This is what we obtain when we apply a K-fold approach to compute and evaluate the calibration transformation on the best promising models we selected in the previous section (considering GMM,LR and SVM). I tested different priors for training the logistic regression model, and evaluated the performance of the calibration transformation in terms of actual DCF for the target application.

Actual DCF and minimum DCF have been evaluated on the target application (0.1,1,1), the prior that changes is the one used to train the prior weighted logistic regression model for score calibration.

GMM:

p_T	Minimum DCF	Actual DCF
0.1	0.131	0.153
0.2	0.127	0.147
0.3	0.127	0.147
0.4	0.127	0.147
0.5	0.126	0.147
0.6	0.126	0.147
0.7	0.126	0.146
0.8	0.126	0.145
0.9	0.134	0.139

LR:

p_T	Minimum DCF	Actual DCF
0.1	0.252	0.257
0.2	0.252	0.260
0.3	0.252	0.252
0.4	0.249	0.252
0.5	0.249	0.254
0.6	0.249	0.254
0.7	0.250	0.256
0.8	0.250	0.258
0.9	0.250	0.257

SVM:

p_T	Minimum DCF	Actual DCF
0.1	0.179	0.185
0.2	0.180	0.192
0.3	0.181	0.196
0.4	0.182	0.196
0.5	0.182	0.199
0.6	0.182	0.192
0.7	0.184	0.194
0.8	0.186	0.194
0.9	0.189	0.198

As we can observe, we should select the best performing calibration transformation based on the lowest value of actual DCFA obtained on the target application we considered so far (0.1,1,1).

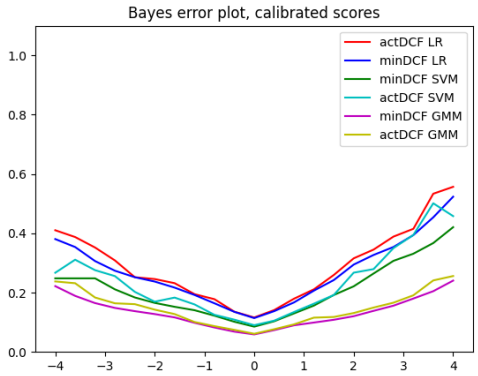
GMM: the best actual DCF reached was 0.139 with a prior equal to 0.9

LR: the best actual DCF reached was 0.252 with two different priors: 0.3,0.4.

SVM: the best actual DCF reached was 0.185 with a prior qual to 0.1.

If we compare the miscalibration rate of the three calibrated models with respect to the raw ones, we observe that calibration has gone through a significant improvement. Without score calibration, miscalibration rate for GMM,SVM and LR was, respectively, 26%,141% and 104%. After calibration, we observe a miscalibration rate that is: 6%,3% and 1% (following the same order as before).

We can better observe the aforementioned improvements through a Bayes error plot, taking into account the same operating points we have considered in the section before.



The improvements we were talking about are clear in the Bayes error plot also for all the operating points considered between -4 and 4.

Finally, in order to choose the final model to be delivered, we still have to try fused model and evaluate their performances. I tested the three models together and then all possible combinations, in the following I report the result obtained:

Fusion GMM + quadratic LR:

p_T	Minimum DCF	Actual DCF
0.1	0.125	0.146
0.2	0.120	0.156
0.3	0.119	0.146
0.4	0.120	0.146
0.5	0.121	0.146
0.6	0.122	0.145
0.7	0.123	0.144
0.8	0.125	0.141
0.9	0.131	0.137

Fusion GMM + SVM:

p_T	Minimum DCF	Actual DCF
0.1	0.128	0.148
0.2	0.122	0.148
0.3	0.123	0.148
0.4	0.125	0.148
0.5	0.126	0.148
0.6	0.125	0.147
0.7	0.125	0.146
0.8	0.125	0.144
0.9	0.133	0.138

Fusion SVM + quadratic LR:

p_T	Minimum DCF	Actual DCF
0.1	0.171	0.197
0.2	0.171	0.196
0.3	0.169	0.198
0.4	0.170	0.201
0.5	0.173	0.203
0.6	0.176	0.195
0.7	0.182	0.196
0.8	0.185	0.189
0.9	0.186	0.195

Fusion SVM + quadratic LR + GMM:

p_T	Minimum DCF	Actual DCF
0.1	0.122	0.146
0.2	0.119	0.165
0.3	0.118	0.166
0.4	0.117	0.165
0.5	0.119	0.165
0.6	0.120	0.164
0.7	0.122	0.161
0.8	0.123	0.143
0.9	0.133	0.137

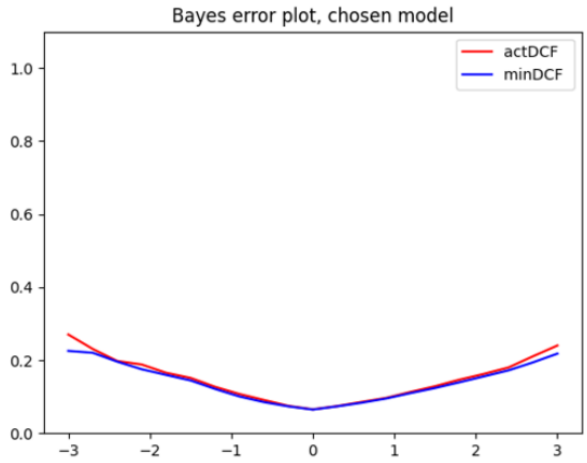
As we can observe in the tables above, the fusion is performing slightly better than the single models in some cases. For example, GMM + quadratic LR provides, for pi_T=0.9, an actual DCF equal to 0.137, which is the best result we have reached so far. The miscalibration rate for that specific application is around 5%, that is comparable with the results obtained before, but, in general, the miscalibration rate is a little bit higher in this case. The best resulti s obtained for GMM + SVM with pi_T=0.9 and by GMM +SVM + LR with the same prior, we choose the first one because, although the actual DCF is the same, the minimum DCF is lower in the second case, which means that the model could reach even better performances with an optimal threshold.

Finally, based on the consideration we have done so far, we can assess that the final delivered model will be composed of a **GMM** with **16 components** for class 1 and **8 component** for class 0, fused with a **quadratic logistic regression** with **lambda = 3.16227766e-02,** calibrated through a prior weighted logistic regression model with **pi_T=0.9**.

-Delivered model evaluation

In the following, we start evaluating the performance of the delivered system on an evaluation dataset.

By applying the chosen model for classification on the considered evaluation set, we obtain a minimum DCF and an actual DCF that are, respectively, 0.181 and 0.198, with a miscalibration rate for the target application equal to 9%, which means that scores are well calibrated. The same goes if we plot actual DCF and minimum DCF for other operating points, as shown in the following Bayes error plot:

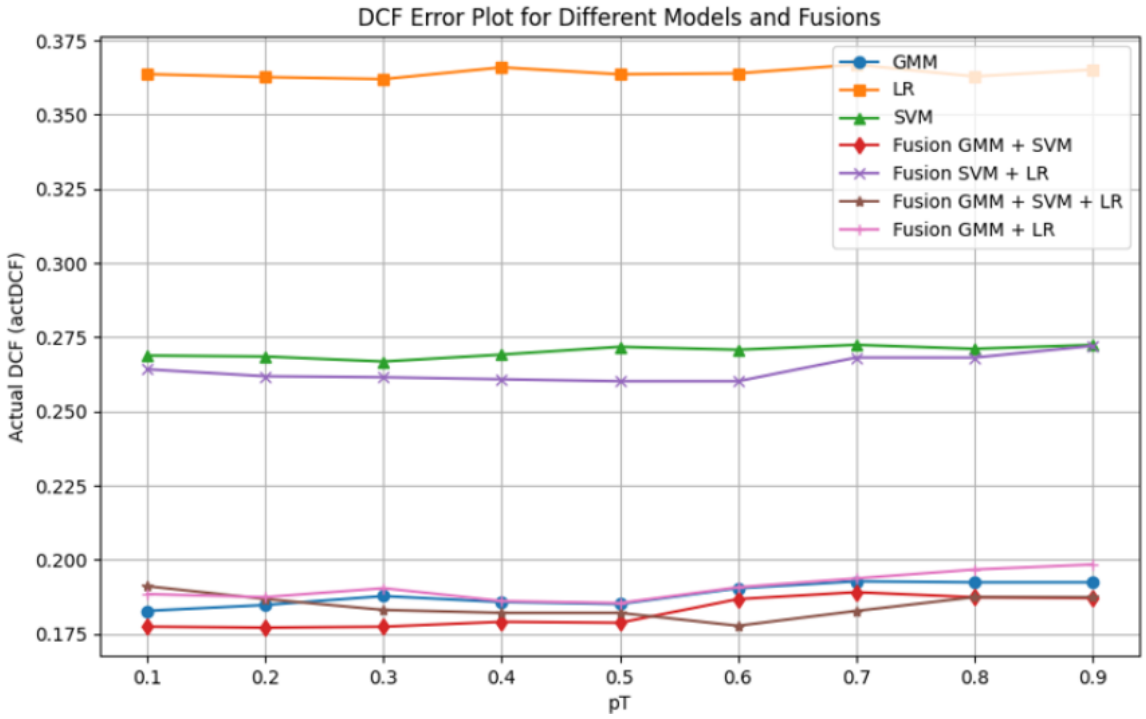


We now consider all the best models we have discussed so far and all their possible fusions to see if my choice has been the best one. Each model has been calibrated with the best prior obtained with the K-fold approach, more information are contained in the tables above.

Model	Actual DCF	Minimum DCF
GMM (pi_T=0.9)	0.192	0.181
LR (pi_T=0.4)	0.366	0.351
SVM (pi_T=0.1)	0.269	0.262
GMM+LR (pi_T=0.9)	0.198	0.181
GMM+ SVM (pi_T=0.9)	0.187	0.178
SVM + LR (pi_T=0.8)	0.268	0.258
GMM + LR + SVM (pi_T=0.9)	0.187	0.187

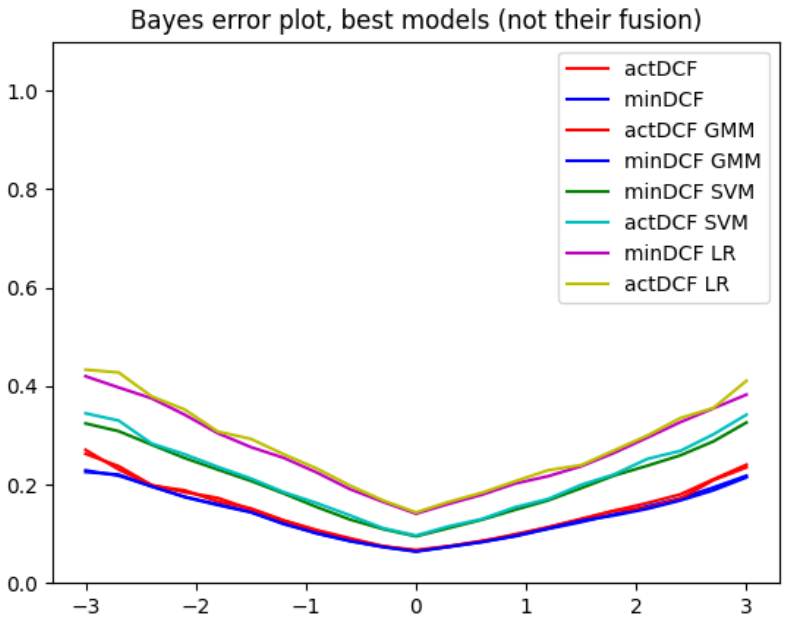
As we can observe, there would have been other choices that were more effective in terms of actual DCF, such as the fusion of all the three models and the fusion of GMM and SVM.

For the DCF error plot, I considered values of p_T changing between 0.1 and 1 with a step of 0.1, this is what I obtained:



Models and fusions that maintain lower and more stable actDCF values are generally preferable. In this case, we can observe that my choice was not the best one on the evaluation dataset, but still obtains a good performance in terms of actual DCF and its stability over different possible applications.

To evaluate if the calibration strategy has been indeed effective, I tried to represent actual DCF and minimum DCF for the three best systems (their fusion has not been taken into account in this analysis). The prior used for calibration, is the same as in the table above, chosen based on the best result obtained on the actual DCF with the K-fold approach. As we can observe in the graph below, scores are calibrated over a wide range of operating points (-3,3 has been considered in this case).



To conclude the analysis, I considered one of the three approach and I tried to vary whether those that have been chosen were the best ones or not. In practice, this is a way to evaluate the training strategy. I used the minimum DCF as parameter, actual DCF would have been better, but it would have required score calibration for all the possible models. Let's see the result we obtain if we evaluate the minimum DCF over the evaluation dataset by varying the hyperparameter of the Gaussian Mixture Model.

As I did in the previous section, I try a different number of components for each class (up to 32) and the two different variations of the model: with full or diagonal covariance matrices.

Components for the class 0:1, components for class 1 varies from 1 up to 32.

Components Class 1	minDCF Full	minDCF diagonal
1	0.4073	0.4098
2	0.4101	0.4121
4	0.2716	0.2774
8	0.2182	0.2767
16	0.2331	0.2136
32	0.2352	0.2175

Components for the class 0:2, components for class 1 varies from 1 up to 32.

Components Class 1	minDCF Full	minDCF diagonal
1	0.2985	0.3663
2	0.2968	0.3657
4	0.2363	0.2686
8	0.2120	0.2722
16	0.2041	0.2152
32	0.2260	0.2170

Components for the class 0:4, components for class 1 varies from 1 up to 32.

Components Class 1	minDCF Full	minDCF diagonal
1	0.3252	0.2498
2	0.3295	0.2515
4	0.2462	0.1947
8	0.2123	0.1940
16	0.2153	0.1782
32	0.2308	0.1828

Components for the class 0:8, components for class 1 varies from 1 up to 32.

Components Class 1	minDCF Full	minDCF diagonal
1	0.2571	0.2536
2	0.2592	0.2553
4	0.2106	0.1922
8	0.1802	0.2026
16	0.1850	0.1808
32	0.1953	0.1838

Components for the class 0:16, components for class 1 varies from 1 up to 32.

Components Class 1	minDCF Full	minDCF diagonal
1	0.2537	0.2770
2	0.2517	0.2790
4	0.2015	0.2075
8	0.1876	0.2115
16	0.2025	0.1895
32	0.1959	0.1881

Components for the class 0:32, components for class 1 varies from 1 up to 32.

Components Class 1	minDCF Full	minDCF diagonal
1	0.3158	0.2563
2	0.3148	0.2616
4	0.2541	0.2193
8	0.2266	0.2234
16	0.2295	0.1925
32	0.2294	0.1955

The hyperparameters we selected when evaluating the performance of GMM on the evaluation set were 16 components for class 1 and 8 components for class 0, using diagonal covariance. The previous selection on the current evaluation data gives a minimum DCF equal to 0.1808, which is the second lowest minimum DCF we obtained by evaluating all possible combinations. The best result is 0.1802 with 8 components for the class 1, 8 for the class 0 and full covariance matrices. Given the fact that it's really close to be the best one, I'd conclude that there couldn't have been, regarding this particular choice, a better option.