POLITECNICO DI TORINO

Course Project

Fingerprint Spoofing Analysis

by Pasquale Bianco

in

Machine Learning & Pattern Recognition

held by Prof. Sandro Cumani

September 2024

Table of Contents

List of Tables

List of Figures

Introduction

The proposed dataset consists of a binary classification problem. The goal is to perform fingerprint spoofing detection, i.e. to identify genuine (`True`, `label 1`) samples and the fake ones (`False`, `label 0`). The samples are computed by a genuine extractor that summarizes high-level characteristics of a fingerprint image. The data is 6-dimensional, composed of floating point numbers. The format of the file is a `.csv` where each row represents a sample: the first values of each row are the features, whereas the last value of each row represents the class (`1` or `0`). The samples are not ordered.

Chapter 1

Dataset general analysis

A first simple analysis can be done by observing classes and analyzing their distributions. To do this, we will consider original dataset, so without pre–processing, to address all kind of distributions and a pairwise representation: in this way we can at least combine two features at a time, to see if they overlap or if they remain on different clusters.

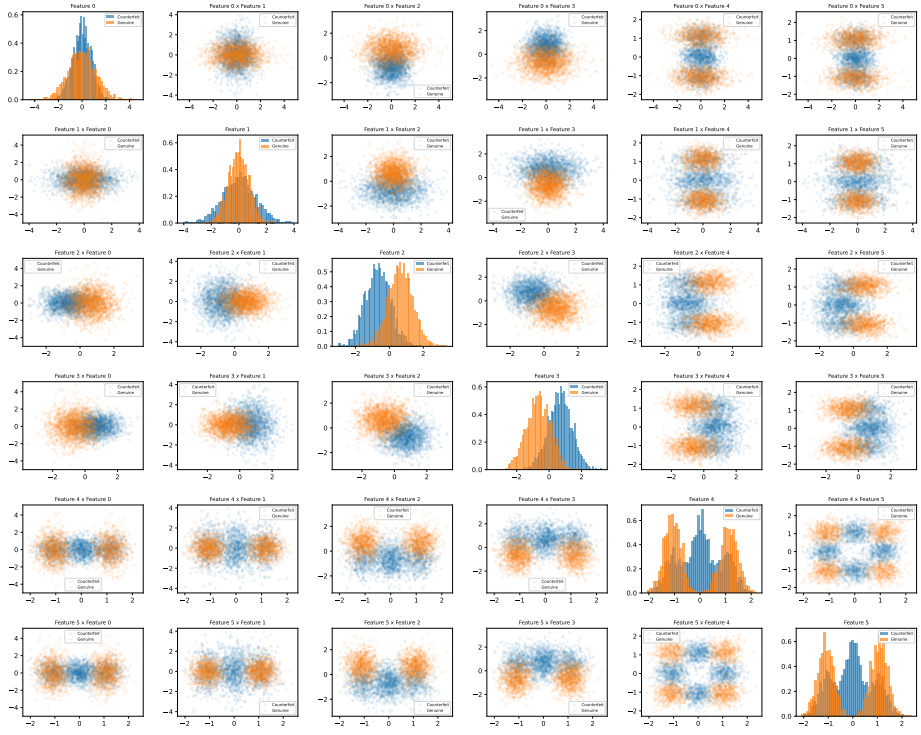

Figure 1.1: Pairwise features representation

This is quite interesting, because there are pairs showing a good cluster discrimination: for instance, scatter plot combining features 2 and 4 contains a very little common application area; on the other hand, scatter plot of features 5 and 6 could be just perfect: clusters are perfectly visible and well defined. Collision areas are limited and could represent a boundary case at classification time, resulting in a 5–10% accuracy error rate.

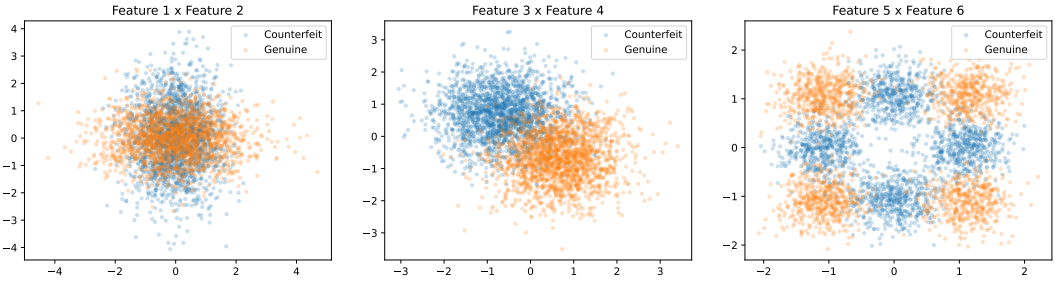

Figure 1.2: Pairwise representation of pairs of features: (1,2), (3,4), (5,6)

On the other side, looking at the histograms, so the per-feature plotting, we can analyze that:

- **features 1 and 2** are almost perfectly overlapped, meaning that best dimensions selected by a PCA would be almost perpendicular to these one;
- **features 3 and 4** have an almost Gaussian distribution, i.e. histograms remind the shape of a Gaussian distribution;
- **features 5 and 6** do not follow at all a Gaussian distribution, but they can at least give us some information to discriminate between classes. In fact, they clearly create very distinct clusters: 4 for each class, as shown above.

Chapter 2

PCA and LDA Analysis

In this chapter, we will analyze the dataset by applying Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA). The common aim of this strategies is to reduce the dimensionality of the dataset and to visualize the data in a lower dimensional space. This will help us to understand if the dataset is linearly separable and if the classes are well defined.

2.1 PCA Analysis

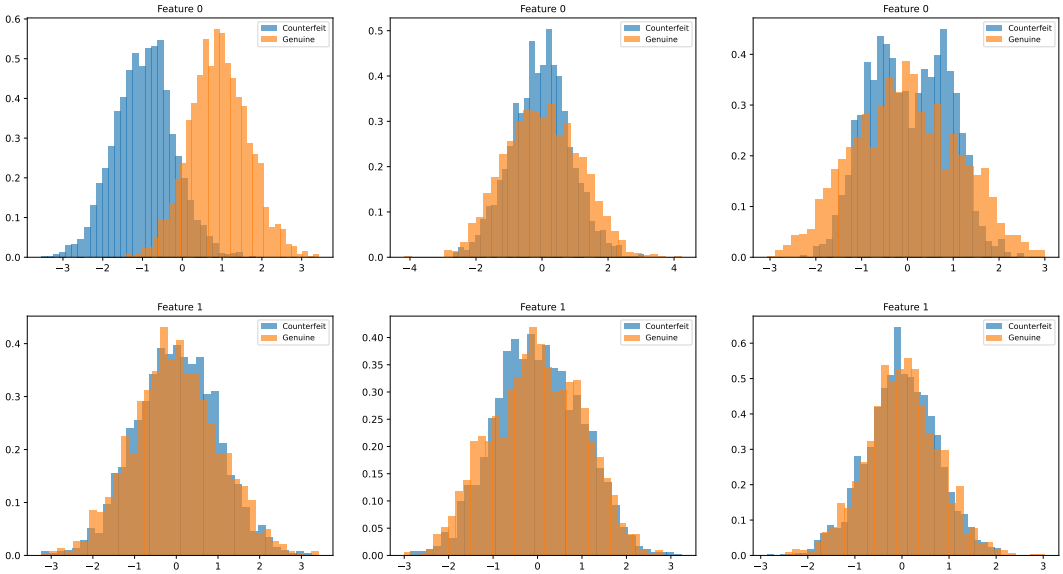

Figure 2.1: PCA: data plotting over all directions produced by PCA

A first attempt to apply PCA can be achieved by plotting the dataset in all directions, sorted by decreasing degree of variance. This will help us to understand if there is a direction that is able to separate classes.

As expected, only the first direction and, somehow, the second and the third ones, can be really useful to separate classes. This is a good result and a good starting point, because it means that the dataset is linearly separable. All of them are Gaussian distributed, but the third direction, that has a double peak.

Moreover, we can try to use PCA as intended, so as a dimensionality reduction:

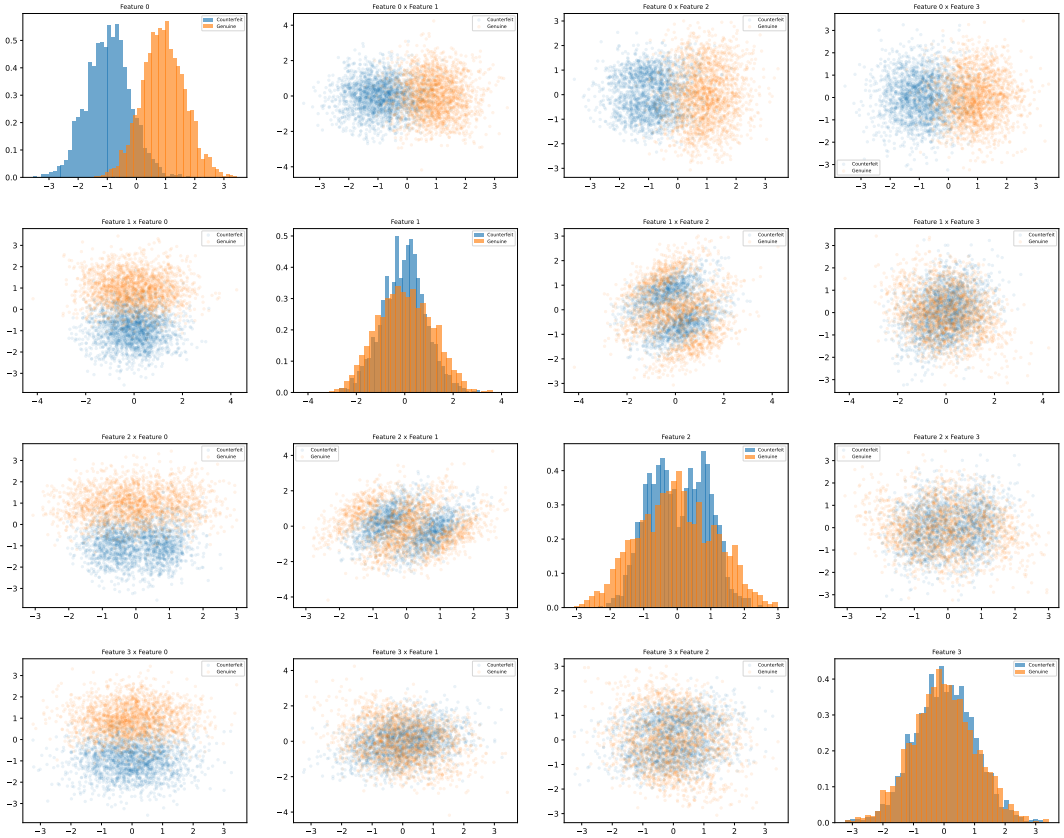

Figure 2.2: PCA: pairwise plotting of data over 4 features (after reduction)

The results are pretty good, because we can see that all scatter plots in the upper diagonal are well separated in most of the feature pairs. This means that this 4 features are good candidates to be used in a classification task. We will see in following chapters which is the best number of dimensions to be used, according to the given classifier.

2.2 LDA Analysis

Linear Discriminant Analysis, instead, is a supervised technique able to find not the directions with the highest variance, as in PCA, but directions where classes are the most separated. Since there are 2 classes, we can only find one discriminant direction, because the other ones do not provide additional information. Anyway, those directions can be used as dimensionality reduction, as an enhanced PCA.

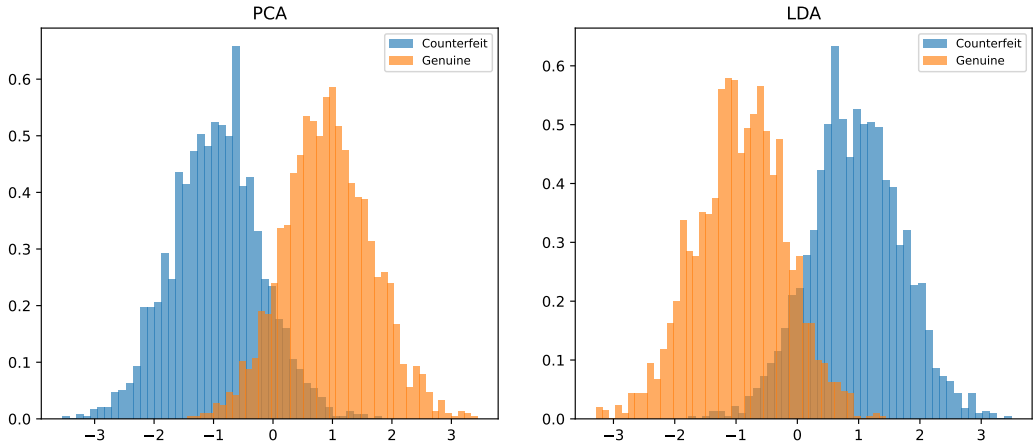

Figure 2.3: PCA and LDA histograms

Figures above show differences between data plotting on PCA vs LDA direction: they are actually pretty similar, but there are some slight differences. In fact, LDA seems to be more robust, representing an higher density of data on the peak of the Gaussian–like curve. Despite this, overlapping areas are still present and they are very similar, so we do not expect a huge improvement in classification accuracy.

Table 2.1 shows the accuracy of the LDA classifier, combined with different thresholds with PCA. For this evaluation, the pure threshold is $t = -0.0136$, while the class means are $\mu_0 = -0.9580$, $\mu_1 = 0.9307$.

As we can see, the best accuracy is achieved with a threshold of 0.05, with a 91% of accuracy with a complete reduction to only one dimension by PCA. Offset experiments revealed that the computed mean is not what the model is actually expecting, probably because classes are not spread perfectly at the same way, i.e. pseudo–Gaussian distributions do not share the same value of standard deviation σ (or variance σ^2). Even if the method has performed well for its standard, we can see that the accuracy is still not very high, so we can expect an error rate of 10% in classification.

PCA dimensions \ Threshold offset	-0.2	-0.1	-0.05	0	+0.05	+0.1	+0.2
1	89.90	90.35	90.35	90.65	**91.00**	90.20	89.85
2	90.25	90.25	90.40	**90.75**	**90.95**	90.55	90.05
3	89.95	**90.75**	**90.75**	**90.75**	**90.60**	90.20	90.20
4	90.00	90.50	**90.65**	**90.75**	90.50	90.15	90.20
5	89.85	90.70	**90.90**	**90.70**	90.50	90.05	90.25
6	89.85	90.65	**90.80**	**90.70**	90.50	90.15	90.25

Table 2.1: LDA accuracy results: combining experimental thresholds and PCA.

Lastly, we can assert that PCA is not modifying the dataset in a way that LDA can improve its performance, because only minor differences showed up. This means that dimensions giving valuable contribution are few and, for this reason, we could get rid of 2/3 vectors, in order to lighten the dataset and, consequently, to avoid overfitting on useless data.

Chapter 3

Multivariate Gaussian fitting

Further classifiers can be built by making some assumptions: one of these is that the dataset is normally distributed. This is a strong assumption, that we can somehow prove by fitting a Gaussian on histograms for each feature.

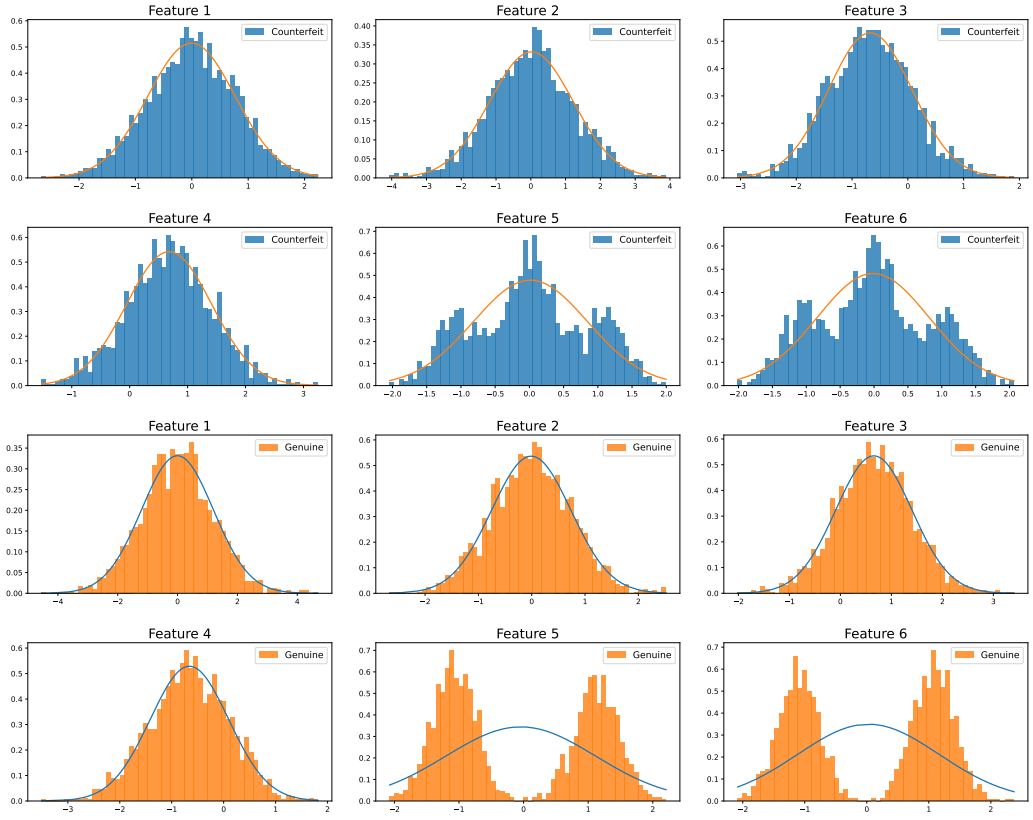

Figure 3.1: MVG fitting performed on all features of all classes

As we can see, the Gaussian Multivariate density fitting is working well with the given data. In particular, it fits well features 1 to 4 for both classes. Instead, features 5 and 6 have another kind of distribution because they act very differently with respect to a Gaussian reference. For this reason, we can expect a quite good result from the Bayesian classifier, even if assumptions are not fully respected. Furthermore, if the Bayesian classifier is built on the first four features, we can expect even better results, if these selected features are discriminant enough (which is not the case, because the last ones are those reflecting differences between classes).

Moreover, this kind of evaluation is based on another characteristic of the data, that is rarely satisfied in real world applications: features should be uncorrelated, in order to be able to draw a good Gaussian curve over each atomically considered feature. This aspect will be discussed in Chapter 4.

Chapter 4

Gaussian Model

Here we will analyze how a Gaussian Model performs on this dataset. As seen in Chapter 3, almost all features (but the last two) are well fitted by a Gaussian distribution; thus we can expect something good and, moreover, some improvements in classification accuracy with respect to the LDA classifier, that is only based on one dimension. Some notes: since this is a binary task, we used the LLR to detect the most likely class, and we used the same prior probability for both classes, because we do not want to introduce any misleading information to the dataset.

%	MVG	MVG Tied	MVG Naive	LDA
PCA: 1	90.75	90.65	90.75	90.65
PCA: 2	91.20	90.75	91.15	90.75
PCA: 3	91.20	90.75	91.00	90.75
PCA: 4	**91.95**	90.75	91.15	90.75
PCA: 5	**92.90**	90.70	91.25	90.70
PCA: 6	**93.00**	90.70	91.10	90.70
Feat: 1..4	**92.05**	90.50	**92.35**	—
Feat: 1, 2	63.50	90.50	—	—
Feat: 3, 4	90.55	90.60	—	—
Feat: 5, 6	72.90	51.05	—	—

Table 4.1: MVG & LDA accuracy results: combining PCA and selected features.

4.1 MVG classifier

MVG classifier performs very well, in particular it achieves 2.30% more accuracy with respect to the LDA model. This is due to the fact that data fits well this distribution, thus, as expected, we got an improvement. In this case, using PCA to remove non–discriminant features is not useful, and, as we can see, the accuracy is slowly decreasing with the number of features. However, the result compared to LDA is always better (but in the case of 1 dimension chosen for the PCA classification because in that case it would be just the same of LDA), meaning that the MVG model is better performing than the LDA model.

4.2 MVG Tied classifier

The MVG Tied model has lower performance with respect to the MVG classifier and, as expected, it is almost perfectly following the LDA model. The reason lies under the common meaning of the two models: MVG Tied model is actually built using as Σ_{Tied} the Within–Class Covariance matrix, which is the same as the LDA model (in LDA it is the argument to be maximized to compute the most discriminant direction).

4.3 MVG Naive classifier

Applying the Naive Bayes model, we can see that the accuracy is not lower than the MVG model, meaning that the naive assumption behind this model, so the independence of the features, is not so far from the actual behavior of the data. This is due to the fact that the features are not so correlated, hence, we expect a very low value of *Pearson's correlation coefficient* in the off–diagonal elements. In the following analysis we will evaluate this aspect, in order to verify this hypothesis.

Moreover, as we saw in all these experiments, considering PCA is not a valuable choice, because all features have their importance: all of them do contribute in a valuable way to the variance of the data and all of them are somehow equally discriminant. In fact, eigenvalues related to the 6 eigenvectors all have similar values, meaning that all of them are important in the classification.

4.3.1 Pearson's Correlation Coefficient

Here the reason for the pretty good performance of the Naive Bayes model is clear: features are very weakly correlated, so the assumption of independence is actually satisfied, and, moreover, we could establish a much lighter model on the

computational point of view: in fact, the difference between Naive and MVG model is just a 0.20% of accuracy.

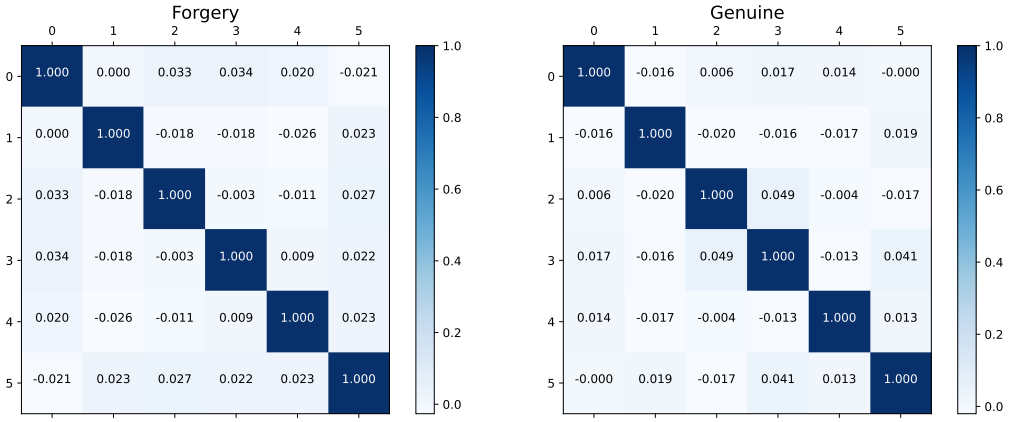

Figure 4.1: Pearson's Correlation Coefficient of both classes

This is a very rare situation, whose only advantage is the possibility to use a Naive model without having a huge loss in accuracy. In fact, in real world applications, features are usually correlated and this is a big issue for this type of models because they are not able to capture this kind of information by themselves. In this case, we can see that the correlation is very low, so the Naive model is actually a good choice.

4.4 MVG on selected features

In the second half of the table, features are selected manually, according to evaluations in Chapter 1:

- **Feat. 1.. 4**: This are the only features reporting a Gaussian-like distribution, so they are the most likely to provide the best non–misleading information to a Bayesian classifier. Results are good, but also quite disappointing: even if the last two parameters are not fitting into a Gaussian distribution, they are still important in the evaluation of the classes. In fact, the accuracy is decreasing (from 0.20% to 0.80%), even if it used only 67% of the original features. Moreover, as seen in Fig. 1.1, last two features are the most separated, so they should be the most important, too.

- **Feat. 1, 2**: here the tied model performs much better than the MVG model. This is actually quite weird, because the two models are performing on a large amount of data, so the MVG should be better in every case. Probably the

given amount is not enough to have a good model evaluation, so the tied model is actually better by chance.

- **Feat. 3, 4**: here both models perform similarly and well, meaning that these features are at the same time fitting the Gaussian and contributing smartly, giving information able to discriminate classes.
- **Feat. 5, 6**: in this last case, the MVG model performs better than the tied model, but the result is not so good in both models. This is perfectly expected because Gaussian is not fitting at all in these features. The tied model can be considered a chance classification, so the result is the same of a random classifier $\sim 50\%$.

4.5 Model evaluation

To conclude this chapter, we can analyze the performances of these models according to the Bayes risk.

4.5.1 Triplet (π_T, C_{fn}, C_{fp}) and effective prior $\tilde{\pi}$

Let's consider the following triplet of (π_T, C_{fn}, C_{fp}), where π_T is the prior probability of class *Genuine*, C_{fn} is the cost of a false negative and C_{fp} is the cost of a false positive:

- (0.5, 1.0, 1.0): uniform prior and costs, so an application where both genuine and counterfeit are equally important and distributed;
- (0.9, 1.0, 1.0): the prior probability of a genuine sample is higher (in this application, most users are legit);
- (0.1, 1.0, 1.0): the prior probability of a genuine sample is higher (in this application, most users are impostors);
- (0.5, 1.0, 9.0): the prior is uniform, but the cost of accepting a fake image is larger than the cost of rejecting a genuine one. In this way we grant an higher security level;
- (0.5, 9.0, 1.0): the prior is uniform, but the cost of rejecting a genuine image is larger than the cost of accepting a fake one. In this way we aim for ease of use for legit users;

But, we can re-parametrize this triplet using the effective prior $\tilde{\pi}$:

$$(\pi_T, C_{fn}, C_{fp}) \;\Leftrightarrow\; (\tilde{\pi}, 1, 1) \tag{4.1}$$

where:

$$\tilde{\pi} = \frac{\pi_T C_{fn}}{\pi_T C_{fn} + (1 - \pi_T) C_{fp}} \tag{4.2}$$

So we can actually represents the previous triplets as the only following one, because e.g. $(0.5, 9.0, 1.0) \sim (0.9, 1.0, 1.0)$, because the $\tilde{\pi}$ is exactly the same, i.e. correspond to same application. So, let's analyze the following effective priors:

- $\tilde{\pi} = 0.5$: uniform distribution (Tab. 4.2, Fig. 4.2);
- $\tilde{\pi} = 0.1$: more likely to classify a sample as counterfeit (Tab. 4.3, Fig. 4.4);
- $\tilde{\pi} = 0.9$: more likely to classify a sample as genuine (Tab. 4.3, Fig. 4.4);

4.5.2 Bayes decision model evaluation: $\tilde{\pi} = 0.5$

PCA	MVG		MVG Tied		MVG Naive	
	actDCF	minDCF	actDCF	minDCF	actDCF	minDCF
m = 6	**<u>0.140</u>**	**<u>0.130</u>**	0.186	0.181	**0.144**	**0.132**
m = 5	**0.142**	**0.133**	0.186	0.180	0.175	0.174
m = 4	**0.161**	**0.155**	0.185	0.182	0.177	0.173
m = 3	0.176	0.175	0.185	0.181	0.180	0.174
m = 2	0.176	0.174	0.185	0.179	0.177	0.173
m = 1	0.185	0.177	0.187	0.177	0.185	0.177

Table 4.2: Bayes decision model evaluation with effective prior $\tilde{\pi} = 0.5$.

In the first application, we can see how the model performs pretty well, granting a very low Bayes decision cost. Moreover, as we can see from Fig. 4.2, the calibration loss is very low, meaning that the threshold to compute classes is well computed and all assumptions made on the dataset almost reflect the reality of the data. Anyway, here the application is just allowing a good classification task that is not, in this case, the only aspect the model is aimed for: the application we want have to minimize the number of false positive, so increasing security of the fingerprint recognition: in this scenario false positive and false negative are almost equally distributed. PCA is not performing very well, in fact, as seen in the pure MVG above, the best results are obtained for computations without Principal Component Analysis. Eventually, pure MVG model is the best performing model; Naive is very close (for the reason seen above with Pearson's coefficient).

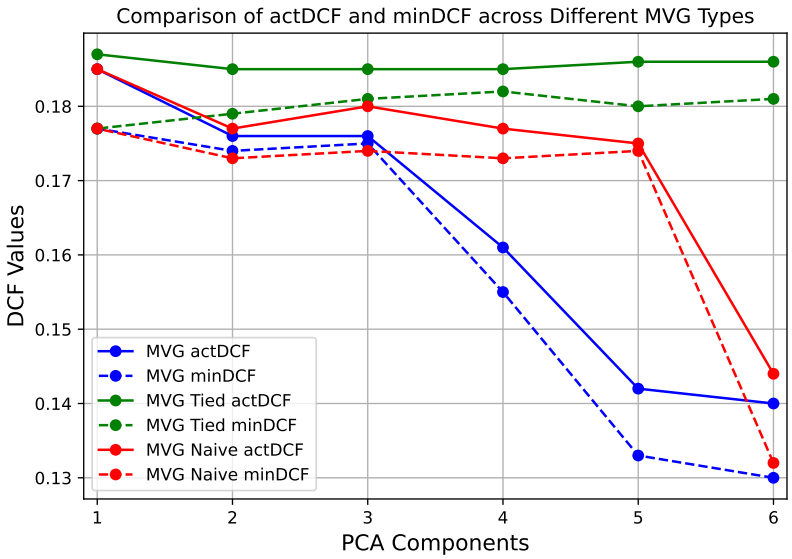

Figure 4.2: Bayes decision model evaluation plot with effective prior $\tilde{\pi} = 0.5$

PCA	MVG		MVG Tied		MVG Naive	
	actDCF	minDCF	actDCF	minDCF	actDCF	minDCF
m = 6	0.305	0.263	0.406	0.363	0.302	0.257
m = 5	0.304	0.274	0.405	0.365	0.393	0.354
m = 4	0.353	0.301	0.403	0.361	0.397	0.361
m = 3	0.388	0.356	0.408	0.368	0.395	0.365
m = 2	0.388	0.353	0.396	0.363	0.387	0.356
m = 1	0.397	0.369	0.402	0.369	0.397	0.369

Table 4.3: Bayes decision model evaluation with effective prior $\tilde{\pi} = 0.1$.

4.5.3 Bayes decision model evaluation: $\tilde{\pi} = 0.1$

In Tab. 4.3 there are, instead, all Bayes decision risk according to prior $\tilde{\pi} = 0.1$, so an application that is aiming to ensure a more secure recognition system. Of course, performance is less satisfying, because the real prior of the model is the one discussed in the previous paragraph, but here we can access a very good improvement in term of false positive reduction: with prior $\tilde{\pi} = 0.5$ there are 65 false positive samples, while using $\tilde{\pi} = 0.1$ we have only 4 false positive samples. This is a very good result, because the model is able to reduce the number of false positive by 93.85%. This is due to an "artificial" translation of the threshold able

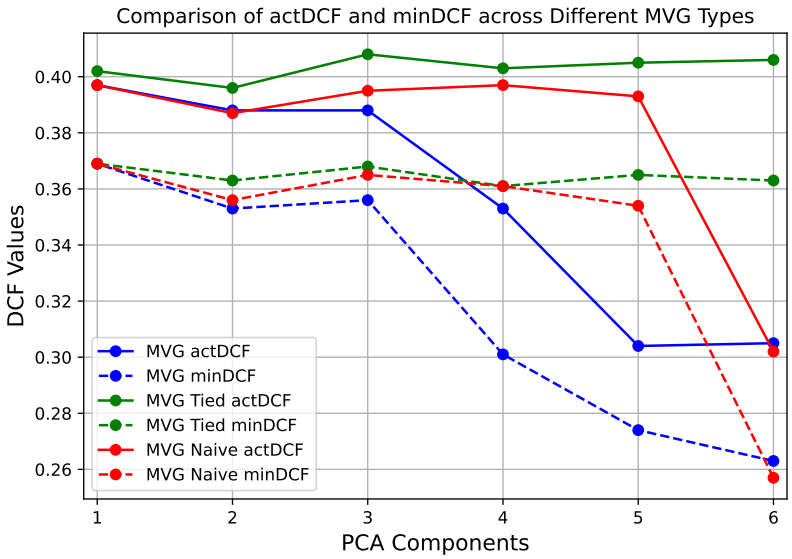

Figure 4.3: Bayes decision model evaluation plot with effective prior $\tilde{\pi} = 0.1$

to enlarge the field of negative acceptance (rejection) of the model. The calibration loss is bigger, and it is completely understandable, because here the prior has been moved even knowing that the real is different from the dataset one. PCA is not helping the classification, because every dimension of the model is needed, so the model without PCA is the best performing one. Eventually, the Naive model is performing slightly better than the MVG, but is probably a matter of randomness, because the two models are very similar and they just perform differently with this amount of data.

4.5.4 Bayes decision model evaluation: $\tilde{\pi} = 0.9$

The last application of interest is the one aiming for ease of use for legit users. It generally performs very similar to the other model, but having a Bayes risk higher than others. This application has the highest calibration loss. MVG has the best minimum DCF, but Naive is the one achieving the lowest actual DCF, meaning that it performs better, in fact it achieves an accuracy of 91.10%, while MVG gets only 86.25% (without PCA). Moreover, PCA is useless, because both minimum DCF and actual DCF are lower than the ones obtained without Principal Component analysis.

PCA	MVG		MVG Tied		MVG Naive	
	actDCF	minDCF	actDCF	minDCF	actDCF	minDCF
m = 6	0.400	0.342	0.463	0.442	0.389	0.351
m = 5	0.398	0.351	0.463	0.445	0.466	0.434
m = 4	0.460	0.415	0.462	0.444	0.463	0.431
m = 3	0.468	0.439	0.457	0.434	0.459	0.434
m = 2	0.443	0.438	0.479	0.435	0.442	0.432
m = 1	0.478	0.434	0.481	0.434	0.478	0.434

Table 4.4: Bayes decision model evaluation with effective prior $\tilde{\pi} = 0.9$.

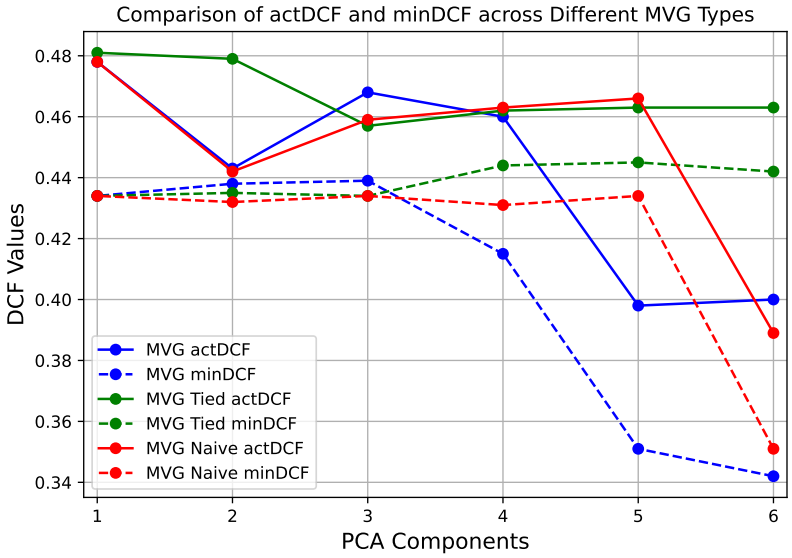

Figure 4.4: Bayes decision model evaluation plot with effective prior $\tilde{\pi} = 0.9$

4.5.5 Bayes error plot

Lastly, we can analyze some more details about how minimum DCF and actual DCF behave with respect to a moving effective prior $\tilde{\pi}$, as shown in Fig. 4.5.

On the x-axis there is $\log \frac{\tilde{\pi}}{1-\tilde{\pi}}$, that is varying in $[-4, 4]$ and on the y-axis the minimum DCF and actual DCF of MVG, Tied and Naive. For our application:

$$\tilde{\pi} = 0.1 \;\Leftrightarrow\; \log \frac{\tilde{\pi}}{1-\tilde{\pi}} \sim -2.20 \tag{4.3}$$

For almost every effective prior, all models seems to be well calibrated, in fact all assumptions have been shown to be very similar to real data.

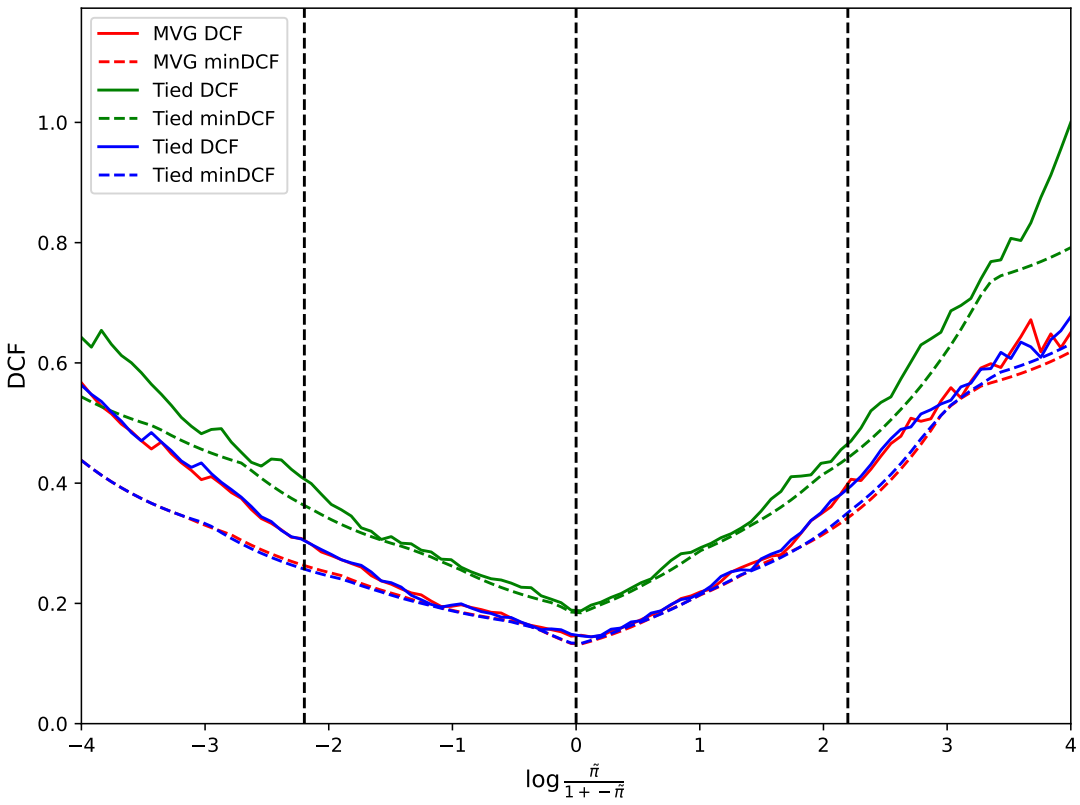

Figure 4.5: Bayes error plot of MVG, Tied and Naive and $m = 6$

Chapter 5

Logistic Regression Model

5.1 Logistic Regression – λ

We now analyze the effectiveness of a logistic regression model. We know that this model contains also a meta–parameter λ: in order to choose the best one, we'll try different values on a log–scale, computing also DCF and minDCF to verify which kind of model actually performs better. In Tab. 5.1 there are most significative values to be analyzed. Using $\lambda = 3e - 3$ we can achieve a slightly better accuracy of 90.80%.

λ	accuracy	actDCF	minDCF
10^{-4}	**90.70%**	**0.402**	**0.364**
10^{-3}	**90.65%**	**0.413**	**0.365**
10^{-2}	**90.75%**	0.457	**0.361**
10^{-1}	**90.75%**	0.852	**0.364**
10^{0}	**90.75%**	1.000	**0.364**
10^{1}	90.50%	1.000	**0.363**
10^{2}	87.15%	1.000	**0.362**
10^{3}	50.40%	1.000	**0.362**

Table 5.1: Logistic Regression model results on different values of λ; the effective prior for Bayes decision model evaluation is $\tilde{\pi} = 0.1$ (the security application)

Moreover, we can discuss about how changing λ affects the Bayes decision evaluation and both actual DCF and minimum DCF. To do it, we can compute

the same evaluation above on a wider range to obtain the following result:

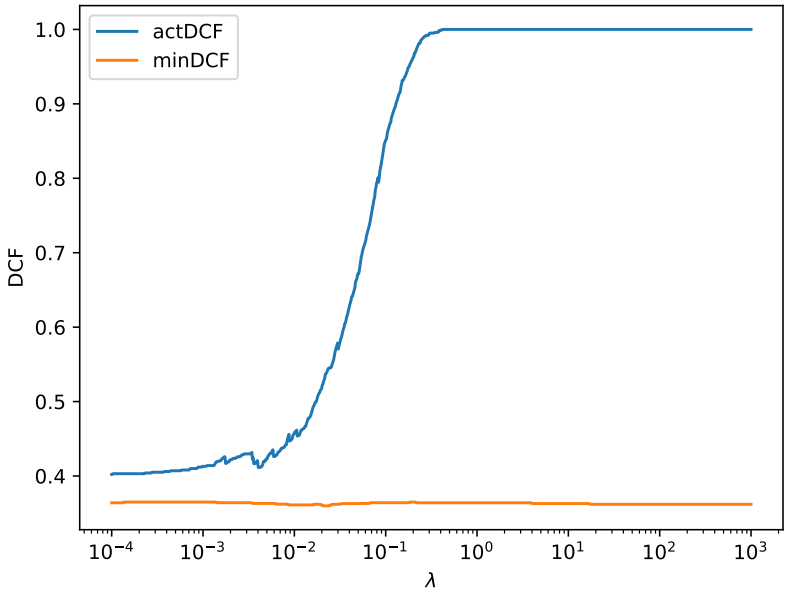

Figure 5.1: DCF and minDCF plot over λ, using $\tilde{\pi} = 0.1$

We can state that this the meta–parameter λ is not improving at all performance of the model. The reason is due to the large number of samples, that makes regularization ineffective, and actually degrades actual DCF, because regularized models tend to lose the probabilistic interpretation of scores. A proof of this can be achieved by considering a few amount of samples in the training set, in order to make the regularization effective.

In Fig. 5.2 λ has the expected behavior of a parameter to be optimized, that fluctuates around a minimum value. In this case, the best $\lambda \sim 0.02$, while lower values produces overfitting and higher values improve overfitting but leading to scores that have lost their probabilistic interpretation.

5.2 Weighted Logistic Regression – λ, π

Let's now analyze how a weighted logistic regression model performs with the full dataset. For this evaluation, for the sake of visualization, we will iterate over a few amount of weights, also the behavior of the other ones can be safely expected from the given ones (by interpolation).

In Fig. 5.3 we can see that weights are non changing the performance of the

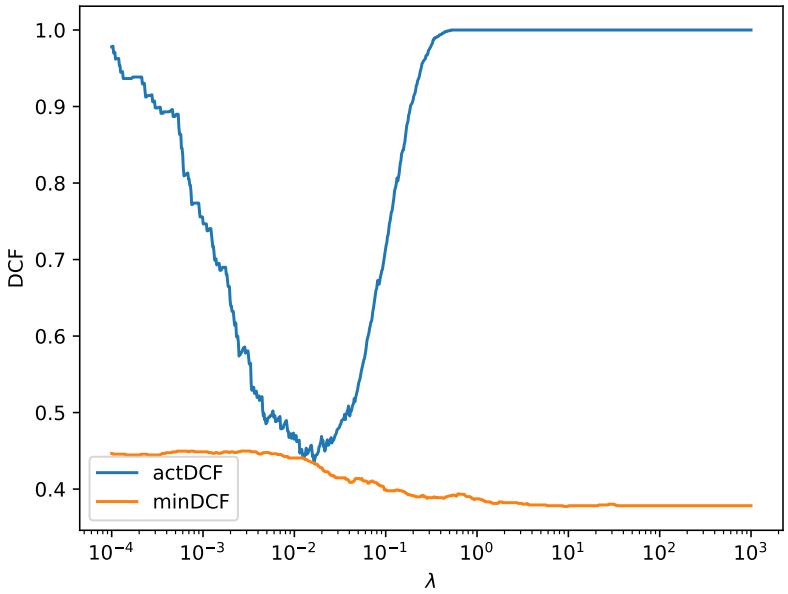

Figure 5.2: DCF and minDCF plot over λ, using $\tilde{\pi} = 0.1$ and only 50 samples

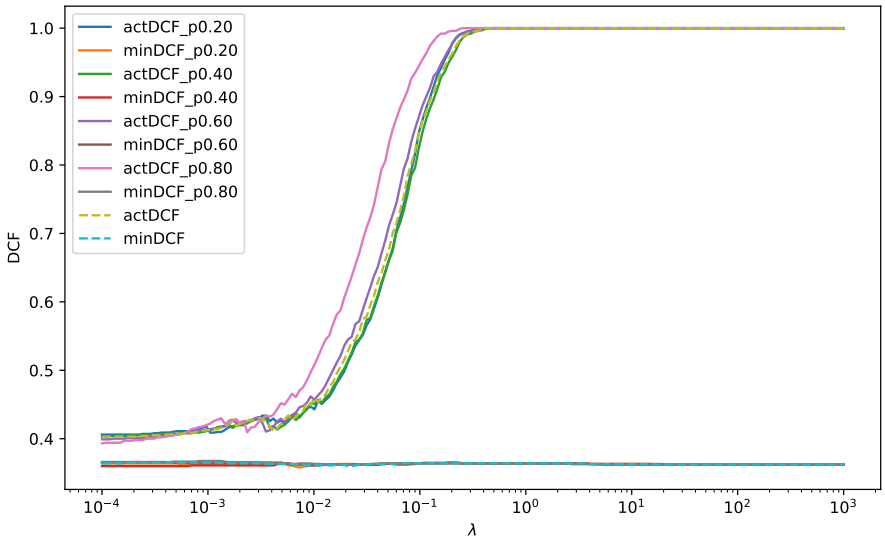

Figure 5.3: Weighted Logistic Regression DCF and minDCF plot over λ and π

model that much and even accuracy is very close. A rough estimation can be done by comparing actDCFs with the one in dashed line, that represent the empirical

prior of the dataset. Reasons are the same discussed above for the non–weighted model. In Tab. 5.2 the best values achieved for each weight are reported.

π	best actDCF	best minDCF
0.20	0.406	**0.358**
0.40	0.403	0.360
0.60	0.399	0.361
0.80	**0.393**	0.362
π_{EMP}	0.402	0.360

Table 5.2: Weighted Logisitc Regression best actDCF and minDCF

5.3 Quadratic Logistic Regression – λ

The quadratic logistic regression model is a much more complex model providing even better results. The origin of the model is the same as the linear one, but here we assume a quadratic function (super set of the linear function) to be the decision boundary, which allows more accurate estimates. Since the difference between standard and weighted models is negligible, we can focus only on the standard one.

λ	accuracy	actDCF	minDCF
10^{-4}	93.95	**0.277**	0.260
10^{-3}	**94.05**	**0.277**	0.259
10^{-2}	**94.10**	0.346	0.249
10^{-1}	93.95	0.752	**0.247**
10^{0}	93.60	1.000	0.284
10^{1}	92.90	1.000	0.324
10^{2}	90.70	1.000	0.326
10^{3}	52.00	1.000	0.327
$2 * 10^{-2}$	**94.25**	0.384	**0.247**

Table 5.3: Quadratic Logistic Regression model results on different values of λ; the effective prior for Bayes decision model is $\tilde{\pi} = 0.1$ (the security application)

In Tab. 5.3 we can see that the quadratic logistic model is the best performing model so far, granting an accuracy up to 94.25% using effective prior $\tilde{\pi} = 0.1$. The best λ is $2 * 10^{-2}$, that is the best trade–off between overfitting and under fitting. The best DCF achieved, instead, is $DCF = 0.266, DCF_{min} = 0.261$, obtained using $\lambda = 3 * 10^{-4}$. In Fig. 5.4 we can see the DCF and minDCF plot over λ.

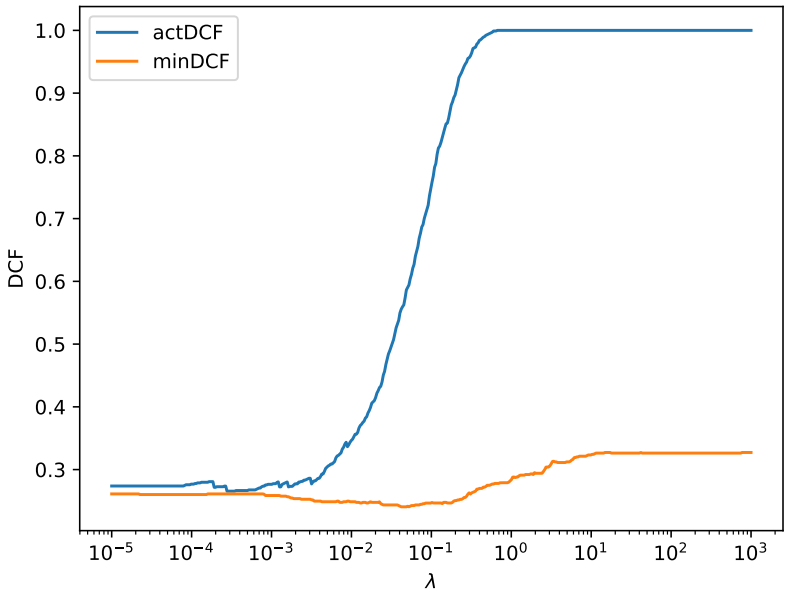

Figure 5.4: Quadratic Logistic Regression DCF and minDCF plot over λ, using effective prior $\tilde{\pi} = 0.1$

Also in this case, regularization does not improve the model performance, but it is still necessary to avoid overfitting. The best λ is the one that minimizes the DCF, that is the one that is able to generalize the best the model.

5.3.1 Preprocessing: centering data

We can also try to center the data, in order to make the model more robust to the presence of outliers. In this case, we can see that the model is not improving at all, but it is still able to generalize the best the model, because original features were already almost standardized. The accuracy remains the same 94.25%, while actual DCF and minimum DCF are only 1% different.

5.3.2 Preprocessing: PCA

As seen also in the MVG model, PCA is not useful because features are already very low, so this kind of preprocessing should be avoided at all for these kind of model. In Tab. 5.4 we can see results of this evaluation with respect to the best accuracy achieved (using the most performing λ); effective prior is still $\tilde{\pi} = 0.1$.

PCA	accuracy	actDCF	minDCF
m = 6	**94.25**	0.396	**0.247**
m = 5	**94.15**	**0.298**	**0.259**
m = 4	92.25	0.847	0.296
m = 3	91.50	0.392	0.351
m = 2	91.10	0.431	0.347
m = 1	90.70	0.404	0.369

Table 5.4: Quadratic Logistic Regression model evaluation on PCA preprocessing; the effective prior for Bayes decision model is $\tilde{\pi} = 0.1$ (the security application)

Results are very interesting, because we can see a significative improvement of about 25% in actual DCF and 2% in minimum DCF. This is due to the fact that the model is able to generalize better the data, because the number of features is reduced; moreover, the model is able to capture the most significative information. The best DCF achieved is $DCF = 0.298, DCF_{min} = 0.259$, obtained using $\lambda = 2 * 10^{-3}$ and $m = 5$.

5.4 Final considerations

Analyzing the three Logistic Regression models, the best performing is the quadratic model, achieving $DCF_{min} = 0.247$, that is a very good outcome. This can be justified by the addition complexity of the model, that takes the square of the time required by the linear logistic model for training. The quadratic logistic model can adapt the best on data distributions that are very intricate for the most discriminative features, as seen in Chapter 1. In the previous model, we tried to avoid the usage of those features, but in this case the model can afford a training set containing also non–Gaussian distributions, enhancing the performance of the classification task.

Chapter 6

SVM

6.1 Linear SVM

In this first section we will analyze how a Linear Support Machine performs with the dataset. Again, we will will consider an effective prior $\tilde{\pi} = 0.1$ for the Bayes decision model evaluation. The results are shown in Table 6.1.

K	C	accuracy	actDCF	minDCF
1.0	1.00e-05	49.60	1.000	1.000
1.0	3.16e-05	90.85	1.000	0.362
1.0	1.00e-04	90.80	1.000	0.364
1.0	3.16e-04	90.75	0.999	0.362
1.0	1.00e-03	90.70	0.959	0.362
1.0	3.16e-03	90.65	0.845	0.365
1.0	1.00e-02	90.75	0.672	0.362
1.0	3.16e-02	90.85	0.579	0.359
1.0	1.00e-01	90.85	0.516	0.358
1.0	3.16e-01	90.90	0.497	0.358
1.0	1.00e+00	**90.95**	**0.489**	**0.358**

Table 6.1: Linear SVM model over different C and $K = 1.0$; the effective prior for Bayes decision model evaluation is $\tilde{\pi} = 0.1$ (the security application)

As we can see from Tab. 6.1, the best DCF is $DCF = 0.489, DCF_{min} = 0.358$,

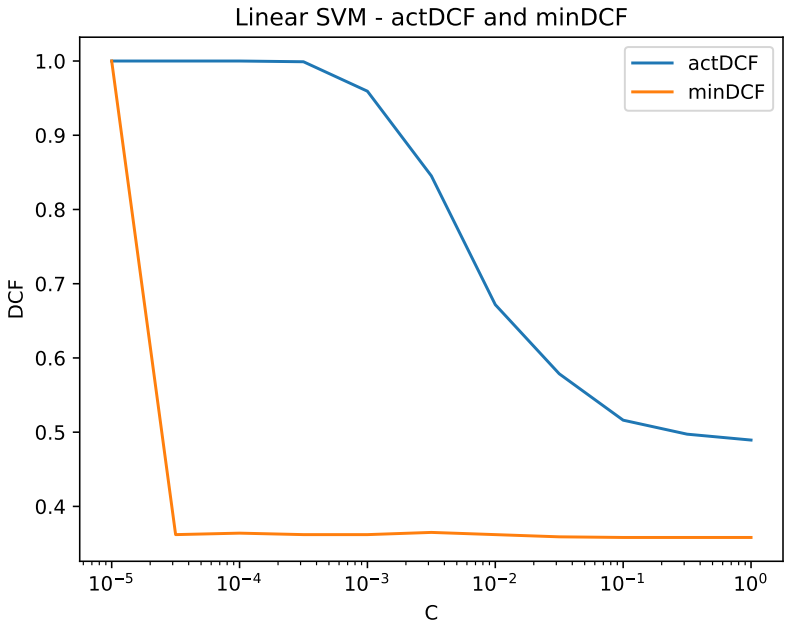

Figure 6.1: Linear SVM – DCF and minDCF plot over C, using $\tilde{\pi} = 0.1$

obtained using $C = 1.0$: small value of C results in underfitting the model because of usage of too wide margin for the classification w.r.t effective prior $\tilde{\pi} = 0.1$. In Fig. 6.1 we can see the DCF and minDCF plot, that can make us aware of a a possible lower bound for the minDCF, that is just the same for all the values of C. On the other side, regularization strongly affect actual DCF, that come to be very inefficient, comparable to a random classifier in case of low values of C. Moreover, calibration is really messy, but this is also due to SVM itself: while considering geometric ideas to divide data with an hyperplane, all probabilistic considerations are lost. In the end, this model's performance are not very interesting because both accuracy and minimum DCF are almost equal to those obtained using linear logistic regression.

Centering data does not affect the performance of the model.

6.2 Polynomial kernel SVM

Polynomial kernel SVM allows the separation hyperplane to be a polynomial expression, but without facing exponential time complexity (as, instead, happen in logistic regression). We will use a second and a third degree polynomial degree for the computations.

Degree	K	C	Accuracy (%)	actDCF	minDCF
2	1.0	1.00e-05	49.60	1.000	1.000
2	1.0	3.16e-05	92.20	1.000	**0.245**
2	1.0	1.00e-04	92.80	1.000	**0.250**
2	1.0	3.16e-04	92.95	0.994	**0.252**
2	1.0	1.00e-03	93.55	0.903	**0.250**
2	1.0	3.16e-03	93.70	0.739	0.264
2	1.0	1.00e-02	93.80	0.571	**0.259**
2	1.0	3.16e-02	**94.10**	0.461	**0.239**
2	1.0	1.00e-01	**94.10**	0.414	**0.253**
2	1.0	3.16e-01	**94.05**	**0.389**	**0.260**
2	1.0	1.00e+00	**94.05**	**0.382**	0.265
3	1.0	1.00e-05	49.60	1.000	1.000
3	1.0	3.16e-05	92.35	0.941	0.259
3	1.0	1.00e-04	92.75	0.842	**0.249**
3	1.0	3.16e-04	93.05	0.723	0.260
3	1.0	1.00e-03	93.65	0.609	0.263
3	1.0	3.16e-03	93.70	0.508	0.257
3	1.0	1.00e-02	**94.10**	0.446	0.262
3	1.0	3.16e-02	**94.00**	0.405	0.289
3	1.0	1.00e-01	**94.00**	**0.383**	0.306
3	1.0	3.16e-01	**94.00**	**0.380**	0.311
3	1.0	1.00e+00	93.95	**0.379**	0.303

Table 6.2: Polynomial kernel SVM model over different C and polynomial degrees, and $K = 1.0$; the effective prior for Bayes decision model evaluation is $\tilde{\pi} = 0.1$ (the security application)

This model performs much better than the linear version and it is somehow similar to the quadratic logistic regression. The best DCF is $DCF = 0.379, DCF_{min} =$

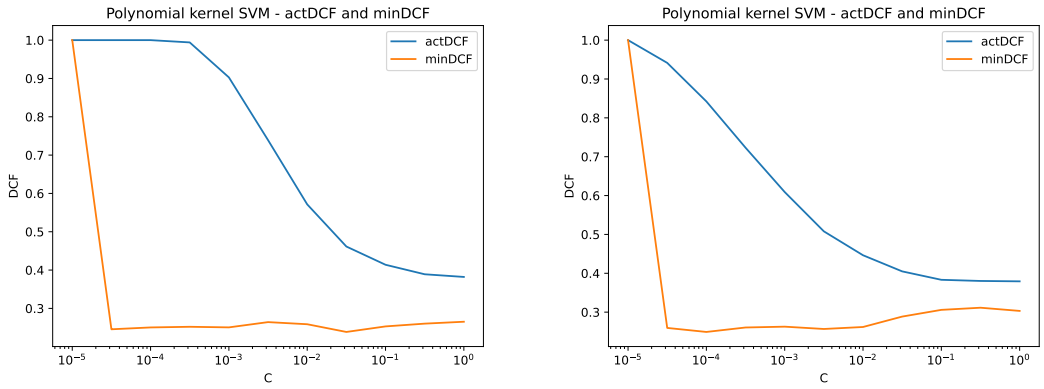

Figure 6.2: Polynomial SVM – DCF and minDCF plot over C, using $\tilde{\pi} = 0.1$

0.239, obtained using $C = 3.16e-02$ and degree 2. Minimum DCF obtained here is almost the same of the one obtained with the quadratic logistic regression, that is, in fact, a very similar model if we consider the separation rule they account for. In Fig. 6.2 we can see the DCF and minDCF plots of second and third degree polynomial kernel SVM. The third degree polynomial kernel SVM is not performing as well as the second degree one, but it is still better than the linear model. Actual DCF starts performing well only with a high enough value of C, meaning that this model needs more regularization to be able to generalize the best the model.

6.2.1 Polynomial kernel of degree 4

However, the best results using this model are obtained using a degree 4 for the polynomial expansion. The reason relies on the fact that the features distribution seen in Ch. 1 is not linear, but it is more similar to a polynomial one: in particular to a degree fourth polynomial, that would tailor the dataset in a very efficient way. The results are shown in Table 6.3.

This model is the best so far, achieving an accuracy of 95.85% and gaining a minimum DCF $DCF_{min} = 0.174$, which is very low compared with the previous best one: it is an improvement of 30%! In addition, actual DCF is very low and the calibration loss is not significative, even if a calibration would get even better results.

6.3 RBF kernel SVM

The Radial Basis Function kernel is a very powerful kernel that is able to tailor the dataset in a very efficient way.

K	C	Accuracy (%)	actDCF	minDCF
1.0	1.00e-05	49.60	1.000	1.000
1.0	3.16e-05	93.55	0.806	0.215
1.0	1.00e-04	94.50	0.652	0.211
1.0	3.16e-04	95.10	0.495	0.192
1.0	1.00e-03	**95.45**	0.395	**0.179**
1.0	3.16e-03	**95.65**	0.319	**0.174**
1.0	1.00e-02	**95.85**	**0.274**	0.190
1.0	3.16e-02	**95.65**	**0.259**	0.210
1.0	1.00e-01	**95.55**	**0.281**	0.235
1.0	3.16e-01	**95.70**	**0.272**	0.257
1.0	1.00e+00	**95.45**	0.302	0.262

Table 6.3: Polynomial kernel ($d = 4$) SVM model over different C and $K = 1.0$; the effective prior for Bayes model evaluation is $\tilde{\pi} = 0.1$ (the security application)

This model performs in a very effective way, achieving an accuracy of 95.95% and a minimum DCF $DCF_{min} = 0.175$, that is the best result so far. The actual DCF is also very low, and the calibration loss is big: it is almost the double of the DCF_{min}. The best results are obtained using $\gamma = e^{-2}$ and $C = 3.16e + 01$. In Fig. 6.4 we can see the DCF and minDCF plot, that shows how the model is able to generalize very well the dataset, with a very low minimum DCF. This model, compared to the polynomial one, can better discriminate through clusters of data, that are those belonging to feature 5 and 6 and that are also those being the most difficult to actually use in a classification model.

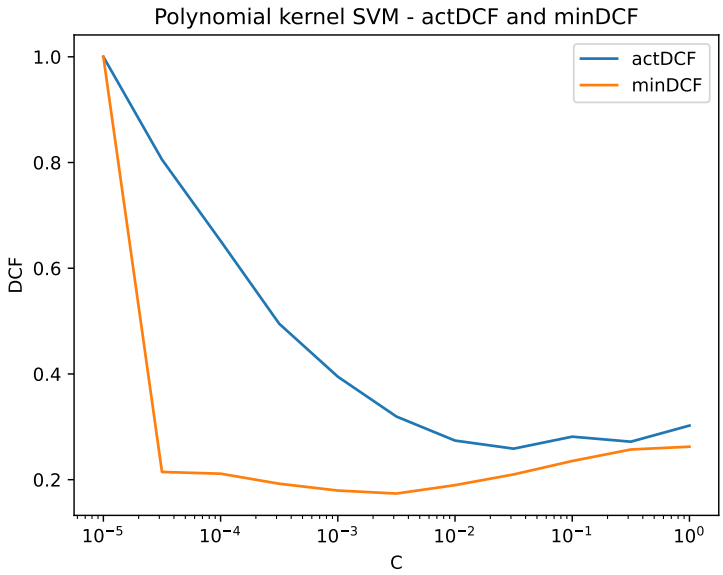

Figure 6.3: Polynomial SVM $d = 4$ – DCF and minDCF plot over C, using $\tilde{\pi} = 0.1$

γ	C	**Accuracy (%)**	**actDCF**	**minDCF**
e^{-4}	1.00e-03	90.10	1.000	0.357
e^{-4}	3.16e-03	90.40	1.000	0.357
e^{-4}	1.00e-02	91.15	1.000	0.346
e^{-4}	3.16e-02	91.65	0.988	0.337
e^{-4}	1.00e-01	91.85	0.882	0.327
e^{-4}	3.16e-01	92.10	0.708	0.315
e^{-4}	1.00e+00	92.80	0.595	0.281
e^{-4}	3.16e+00	93.40	0.527	0.251
e^{-4}	1.00e+01	94.00	0.445	0.230
e^{-4}	3.16e+01	94.25	0.413	0.247
e^{-4}	1.00e+02	94.20	0.399	0.257
e^{-3}	1.00e-03	90.25	1.000	0.336
e^{-3}	3.16e-03	91.85	1.000	0.334
e^{-3}	1.00e-02	91.90	1.000	0.329
e^{-3}	3.16e-02	92.35	0.976	0.313
e^{-3}	1.00e-01	92.90	0.836	0.290
e^{-3}	3.16e-01	93.30	0.678	0.253
e^{-3}	1.00e+00	94.10	0.550	0.243
e^{-3}	3.16e+00	94.45	0.467	0.239
e^{-3}	1.00e+01	94.50	0.441	0.241
e^{-3}	3.16e+01	94.45	0.441	0.243
e^{-3}	1.00e+02	94.60	0.433	0.250

Table 6.4: RBF kernel SVM model, part 1; the effective prior for Bayes model evaluation is $\tilde{\pi} = 0.1$ (the security application)

γ	C	**Accuracy (%)**	**actDCF**	**minDCF**
e^{-2}	1.00e-03	91.20	1.000	0.327
e^{-2}	3.16e-03	92.15	1.000	0.324
e^{-2}	1.00e-02	92.70	1.000	0.298
e^{-2}	3.16e-02	93.15	1.000	0.276
e^{-2}	1.00e-01	93.35	0.984	0.257
e^{-2}	3.16e-01	94.25	0.798	0.249
e^{-2}	1.00e+00	94.70	0.675	0.236
e^{-2}	3.16e+00	95.05	0.600	0.237
e^{-2}	1.00e+01	95.30	0.501	0.196
e^{-2}	3.16e+01	**95.50**	0.422	**0.175**
e^{-2}	1.00e+02	**95.85**	0.321	0.199
e^{-1}	1.00e-03	93.25	1.000	0.329
e^{-1}	3.16e-03	93.20	1.000	0.331
e^{-1}	1.00e-02	92.75	1.000	0.290
e^{-1}	3.16e-02	93.70	1.000	0.272
e^{-1}	1.00e-01	94.75	1.000	0.236
e^{-1}	3.16e-01	95.55	0.991	0.202
e^{-1}	1.00e+00	**95.75**	0.901	**0.183**
e^{-1}	3.16e+00	**95.80**	0.715	**0.188**
e^{-1}	1.00e+01	**95.95**	0.537	0.229
e^{-1}	3.16e+01	95.50	0.411	0.268
e^{-1}	1.00e+02	94.90	0.375	0.329

Table 6.5: RBF kernel SVM model, part 2; the effective prior for Bayes model evaluation is $\tilde{\pi} = 0.1$ (the security application)

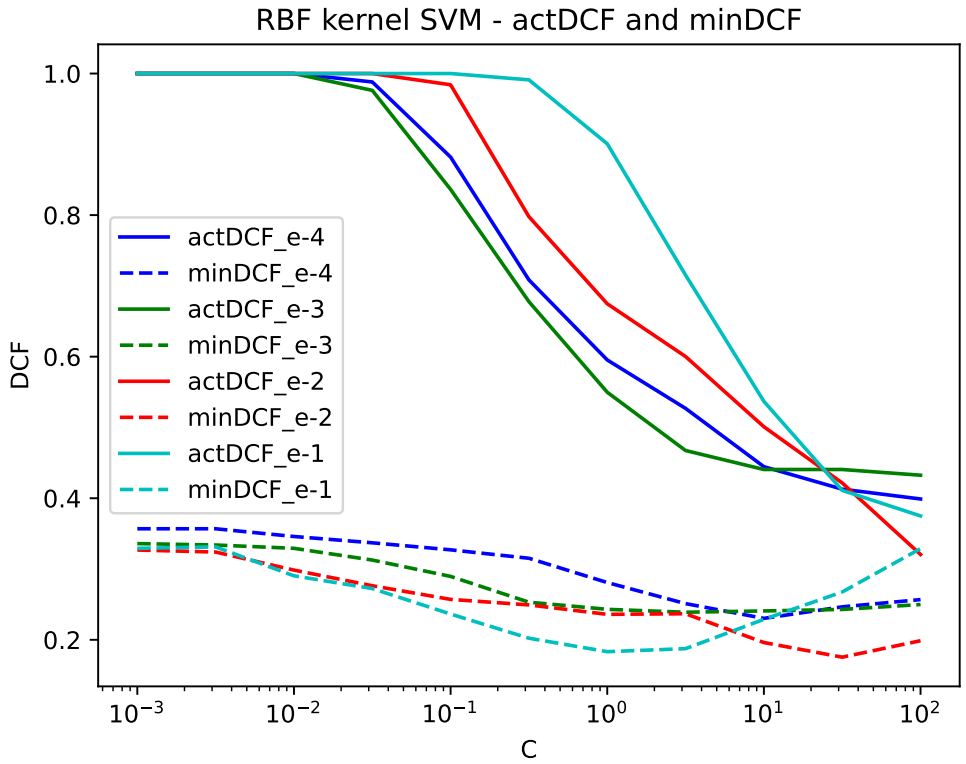

Figure 6.4: RBF SVM $d = 4$ – DCF and minDCF plot over C, using $\tilde{\pi} = 0.1$

Chapter 7

GMM

In this chapter we will analyze the effectiveness of another generative model for the classification task: the Gaussian Mixture Model (GMM). The GMM is a probabilistic model that assumes that the data is generated by a mixture of several Gaussian distributions: it can be considered as an enhanced version of the MVG, in which data are fit in multiple weighted Gaussian distributions. The GMM is a powerful model that can capture complex patterns in the data, and we could be finally able to make a function fit last two features of this dataset, that are those having most discriminative and complex distribution at the same time. We will analyze all three covariance matrix type, that are: *Full*, *Diagonal* and *Tied* in different tables.

7.1 GMM with Full Covariance Matrix

In Tab. 7.1 we can see the results of the GMM with *Full* covariance matrix. The best results are obtained with 8 components for class 0 and 8 components for class 1: best accuracy is 95.35% for both classes, and the best result for minimum DCF is 0.154. The minimum DCF is obtained with 8 components for class 0 and 16 components for class 1. The results are very good, and the GMM with *Full* covariance matrix is able to capture the complex patterns in the data. Moreover, the minimum DCF obtained is the lowest so far and can be used to obtain the best classification accuracy for effective prior $\tilde{\pi} = 0.1$.

7.2 GMM with Diagonal Covariance Matrix

In Tab. 7.2 we can see the results of the GMM with *Diagonal* covariance matrix. The best results are obtained with 4 components for class 0 and 16 components for class 1: best accuracy is 95.05% and the best result for minimum DCF is

	1	**2**	**4**	**8**	**16**	**32**
	86.05%	86.80%	92.35%	94.30%	94.35%	—
1	0.664	0.668	0.483	0.312	**0.251**	—
	0.263	0.265	0.214	0.185	**0.159**	—
	88.20%	88.55%	93.10%	95.05%	94.80%	—
2	0.654	0.615	0.391	0.273	0.252	—
	0.218	0.216	0.223	0.186	0.170	—
	87.95%	88.20%	92.95%	94.90%	94.55%	—
4	0.653	0.653	0.426	0.293	0.271	—
	0.233	0.232	0.216	0.190	0.168	—
	89.30%	89.75%	93.20%	**95.35%**	**95.05%**	—
8	0.601	0.592	0.358	0.297	0.273	—
	0.183	0.180	0.194	0.184	**0.154**	—
	88.45%	88.70%	92.50%	94.85%	94.60%	—
16	0.672	0.654	0.422	0.312	0.298	—
	0.180	0.185	0.188	0.179	**0.158**	—
	88.45%	88.75%	92.70%	94.80%	94.60%	—
32	0.637	0.620	0.430	0.313	0.296	—
	0.189	0.190	0.190	0.185	0.163	—

Table 7.1: Accuracy, actual DCF and minimum DCF for each couple of clustering components for class 0 and class 1. Covariance type: *Full*; effective prior $\tilde{\pi} = 0.1$.

0.136, that is the best so far. The results are very good, and the GMM with *Diagonal* covariance matrix is able to capture the complex patterns in the data. The reason behind this surprising result relies behind the non perfect way by which this algorithm performs optimization and features of this dataset. As shown in Sec. 4.3.1, features are almost independent, so using a diagonal covariance matrix can be a good choice. because it would automatically set to zero the off-diagonal elements of the covariance matrix, that are the ones we can assume to be zero, indeed. In the GMM version with covariance, instead, the optimum can be found not by setting to zero the off-diagonal elements, but by finding the values for them, that is a misleading way to find a solution, which ends in a worse result; probably, in the full covariance matrix case, the algorithm maximizing the likelihood just finds a local maximum, whilst in the Diagonal case, the constraint about off-diagonal elements helps in finding better parameters.

	1	**2**	**4**	**8**	**16**	**32**
1	85.80% 0.684 0.257	85.80% 0.696 0.261	92.05% 0.463 0.210	92.10% 0.481 0.204	94.45% 0.284 0.141	— — —
2	86.35% 0.660 0.245	86.25% 0.659 0.249	91.90% 0.466 0.199	91.95% 0.482 0.203	94.60% 0.266 0.154	— — —
4	87.90% 0.660 **0.145**	87.75% 0.660 **0.154**	93.25% 0.418 **0.148**	93.50% 0.416 **0.140**	**95.05%** 0.328 **0.136**	— — —
8	87.50% 0.622 0.174	87.45% 0.631 0.181	93.30% 0.382 0.167	93.35% 0.390 0.151	**95.05%** 0.327 0.139	— — —
16	87.60% 0.651 0.222	87.45% 0.659 0.222	93.15% 0.436 0.202	93.30% 0.442 0.198	94.75% 0.377 0.157	— — —
32	— — —	— — —	— — —	— — —	— — —	— — —

Table 7.2: Accuracy, actual DCF and minimum DCF for each couple of clustering components for class 0 and class 1. Covariance type: *Diagonal*; effective prior $\tilde{\pi} = 0.1$

7.3 GMM with Tied Covariance Matrix

In Tab. 7.3 we can see the results of the GMM with *Tied* covariance matrix. Here outcomes are not very interesting, but still a 95.60% accuracy and a minimum DCF of 0.164 are obtained, that is a very good result if compared with previous models such as linear logistic regression, MVG and LDA.

	1	**2**	**4**	**8**	**16**	**32**
	86.05%	86.05%	86.05%	94.30%	94.50%	94.50%
1	0.664	0.664	0.664	0.321	0.320	0.334
	0.263	0.263	0.263	**0.155**	**0.158**	**0.166**
	86.05%	86.05%	86.05%	94.30%	94.50%	94.50%
2	0.664	0.664	0.664	0.321	0.320	0.334
	0.263	0.263	0.263	0.155	0.158	**0.166**
	86.05%	86.05%	86.05%	94.30%	94.50%	94.50%
4	0.664	0.664	0.664	0.321	0.320	0.334
	0.263	0.263	0.263	**0.155**	**0.158**	**0.166**
	89.65%	89.65%	89.65%	**95.60%**	**95.55%**	**95.50%**
8	0.539	0.539	0.539	0.293	0.275	0.285
	0.156	**0.156**	**0.156**	**0.164**	**0.164**	0.171
	89.15%	89.15%	89.15%	95.25%	95.30%	**95.50%**
16	0.547	0.547	0.547	0.285	0.277	0.287
	0.151	0.151	0.151	0.167	0.165	**0.167**
	88.90%	88.90%	88.90%	95.20%	94.95%	**95.30%**
32	0.580	0.580	0.580	0.303	0.285	0.285
	0.164	**0.164**	**0.164**	0.176	0.182	0.186

Table 7.3: Accuracy, actual DCF and minimum DCF for each couple of clustering components for class 0 and class 1. Covariance type: *Tied*; effective prior $\tilde{\pi} = 0.1$

Chapter 8

Best performing models

So far we have analyzed the performance of the models in terms of the accuracy of the predictions and DCF, according to Bayes risk model. In this chapter, we will compare all those models in order to choose the best candidate for our security-centered application.

- **LDA**: this model is computing just directions on which data are the most discriminative possible; best performance is accuracy of 91.00%; (m=1, threshold offset: +0.05)
- **MVG**: this model naively tries to fit a Gaussian on training data, achieving a top performance accuracy of 93.00% and a minimum DCF of 0.253 (no PCA, Naive)
- **Linear Logistic Regression**: this model is actually similar to LDA, but preserving a probabilistic interpretation. Best performance are accuracy 90.75% and minimum DCF 0.361 ($\lambda = 10^{-2}$)
- **Quadratic Logistic Regression**: this is the game changer of this analysis, because it handles to get accuracy of 94.25% and minimum DCF of 0.247 ($\lambda = 2 * 10^{-2}$, no PCA)
- **Linear SVM**: this achieves something similar to LDA and MVG as well: accuracy 90.95% and minimum DCF 0.358 (K = 1.0, C = 1)
- **Polynomial SVM**: this model is able to tailor a separation hyperplane of the desired polynomial degree and can achieve a best result using a degree 4 polynomial: accuracy 95.85% and minimum DCF of 0.174 (k = 1.0, C = 3.16e-3)
- **RBF SVM**: this model is based, instead, on considerations about distances to other points, that can lead in a degenerated case to a clustering algorithm.

Best performance is accuracy 95.95% and minimum DCF of 0.175 ($\gamma = e^{-2}$ C = 3.16e1)

- **GMM full**: this model can easily fit distributions of this dataset, making possible a best accuracy of 95.35% and minimum DCF of 0.154 (no PCA, cl0 = 8, cl1 = 1)
- **GMM diagonal**: here, instead, the best accuracy is 95.05% and minimum DCF of 0.136 (no PCA, cl0 = 4, cl1 = 16)
- **GMM tied**: this model is able to achieve a best accuracy of 95.60% and minimum DCF of 0.155 (no PCA, cl0 = 1, cl1 = 8)

Model	**Accuracy**	**Min DCF**	**Parameters**
LDA	91.00%	—	m = 1, $t_{off} = +0.05$
MVG	93.00%	0.253	No PCA, Naive
Linear LR	90.75%	0.361	$\lambda = 10^{-2}$
Quadratic LR	94.25%	0.247	$\lambda = 2 \cdot 10^{-2}$, no PCA
Linear SVM	90.95%	0.358	K = 1.0, C = 1
Polynomial SVM	**95.85%**	0.174	deg = 4, K = 1.0, C = 3.16e-3
RBF SVM	**95.95%**	0.175	$\gamma = e^{-2}$, C = 3.16e1
GMM Full	**95.35%**	**0.154**	No PCA, $c_0 = 8$, $c_1 = 16$
GMM Diagonal	95.05%	**0.136**	No PCA, $c_0 = 4$, $c_1 = 16$
GMM Tied	**95.60%**	**0.155**	No PCA, $c_0 = 1$, $c_1 = 8$

Table 8.1: Comparison of different models with their best accuracies, minimum DCF, and parameters. Effective prior selected: $\tilde{\pi} = 0.1$ (the security application)

Best performing models are GMM, SVM and Quadratic Logistic Regression, but GMM with Diagonal covariance matrix is the most promising one. We will choose only four of them for last analysis to be made in following chapters and sections, which are:

- **Quadratic Logistic Regression**
- **Polynomial SVM of degree 4**
- **RBF SVM**
- **GMM with Diagonal covariance**

8.1 Performance on different application

Now analyze how these models perform with different applications through the Bayes error plot in Fig. 8.1.

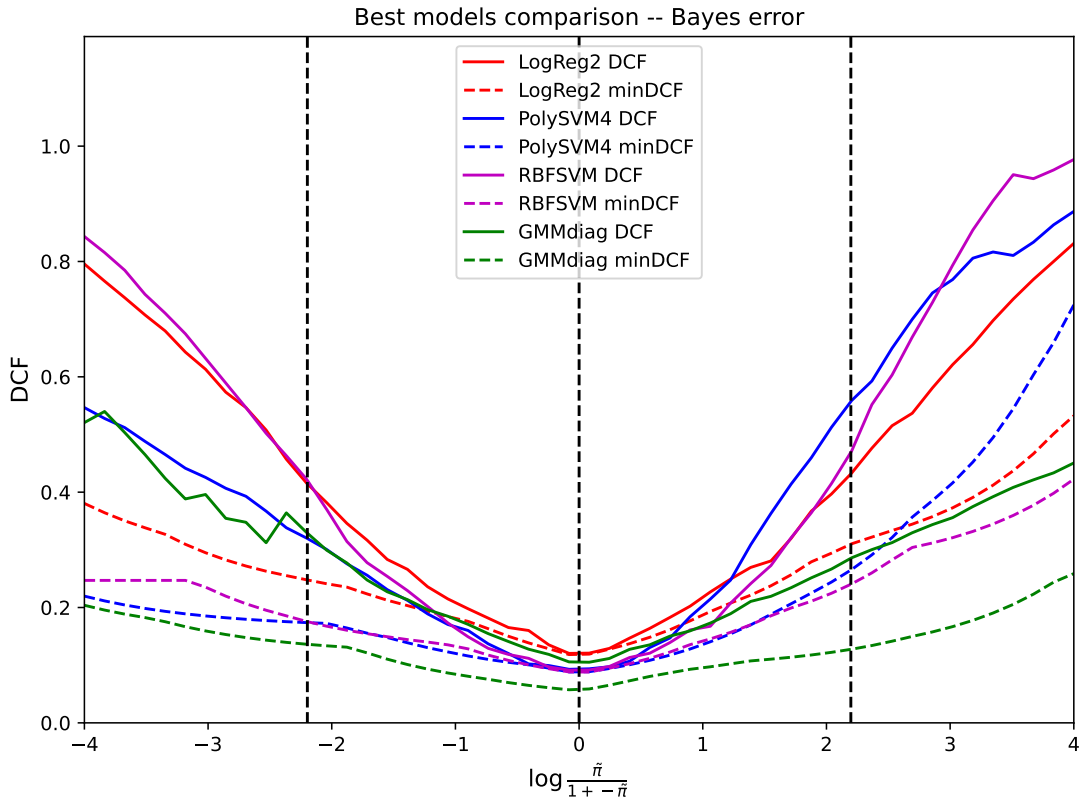

Figure 8.1: Bayes error plot of best performing models

We can clearly see that the best model is the GMM with Diagonal covariance matrix and the worst is the Quadratic Logistic Regression: in all applications these model have, respectively, the lowest and the highest Bayes error. The other two models, instead, perform better on some applications and worse on others: in particular, the RBF SVM is the best model for a easy–to–use application, whilst the the Polynomial SVM of degree 4 is the best for a security–centered application.

However, in all models there are significant mis–calibration errors, that are potentially reducing accuracy of the models by some points.

Chapter 9

Calibration and Fusion

9.1 Calibration

The best model is the GMM diagonal, that is the one having the lowest minimum DCF, so we can think of calibrating its score in order to improve even more results.

Model	Accuracy	actDCF	minDCF
Validation set			
Quad LR	94.25%	0.270	0.254
Poly4 SVM	95.75%	0.187	0.177
RBF SVM	95.50%	0.189	0.178
GMM diag	**96.85%**	**0.166**	0.138

Table 9.1: K-fold calibration results on Validation set

In Tab. 9.1, as expected, we can see a huge improvement when calibrating scores. The best actual DCF achieved in the evaluation set is 0.166 with an accuracy of 96.85%, that is a very interesting result. Other models also improved their results, but are still performing worse than the GMM diagonal. Let's now see how the calibration improved the Bayes error w.r.t. other applications, so not only our target one (for the sake of simplicity and for a better visualization, we will get rid of the quadratic logistic regression model).

Calibration improved the Bayes error of all models: actual DCF is now very close to the minimum possible DCF. Also for other applications the GMM with diagonal covariance matrix is the best choice, in particular it is the best model in fitting the dataset.

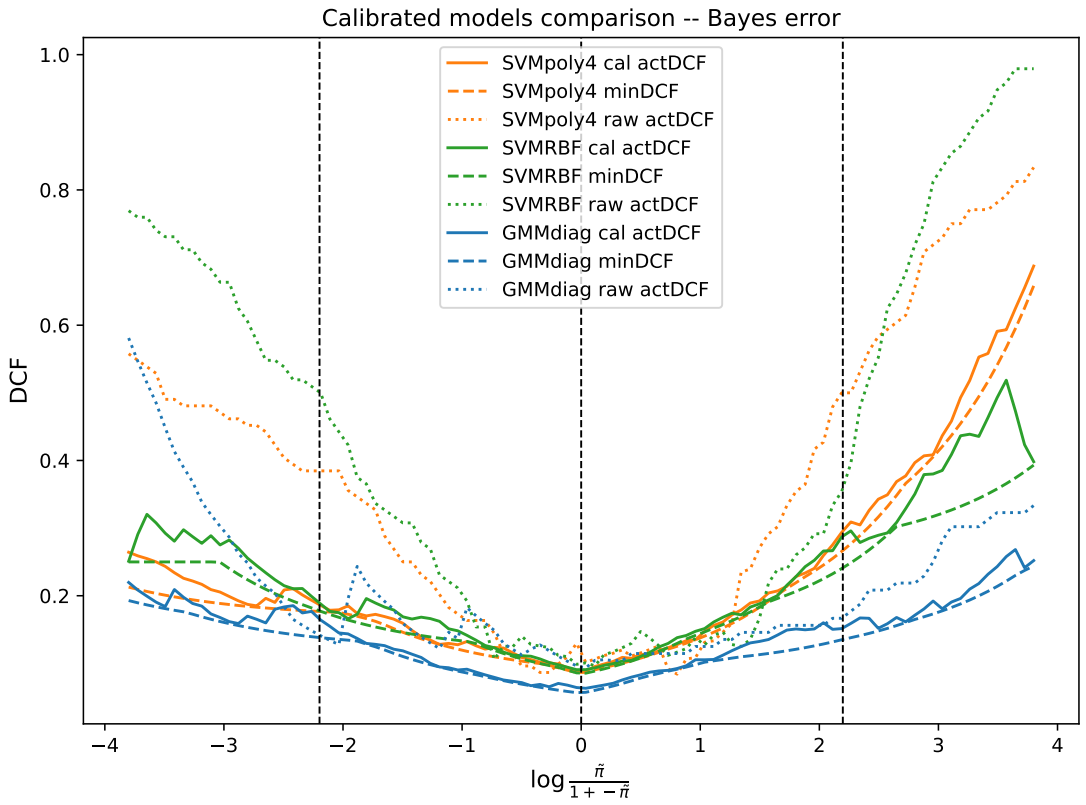

Figure 9.1: Bayes error plot of calibrated best performing models; Validation set

9.2 Fusion

Lastly, we can try to fuse Quadratic Logistic Regression, RBF Support Vector Machine and GMM with diagonal covariance matrix together in order to gather all good contributions, because all of them can detect some aspects that other models can not.

Accuracy	actDCF	minDCF
Validation set		
96.75%	0.160	0.128

Table 9.2: Fusiion calibration results on Validation set

This fused model is the best so far, having the lowest actual DCF, that is 0.160, so we can deliver this as final model of this project. Let's now see how the

calibration improved the Bayes error w.r.t. other applications, so not only our target one.

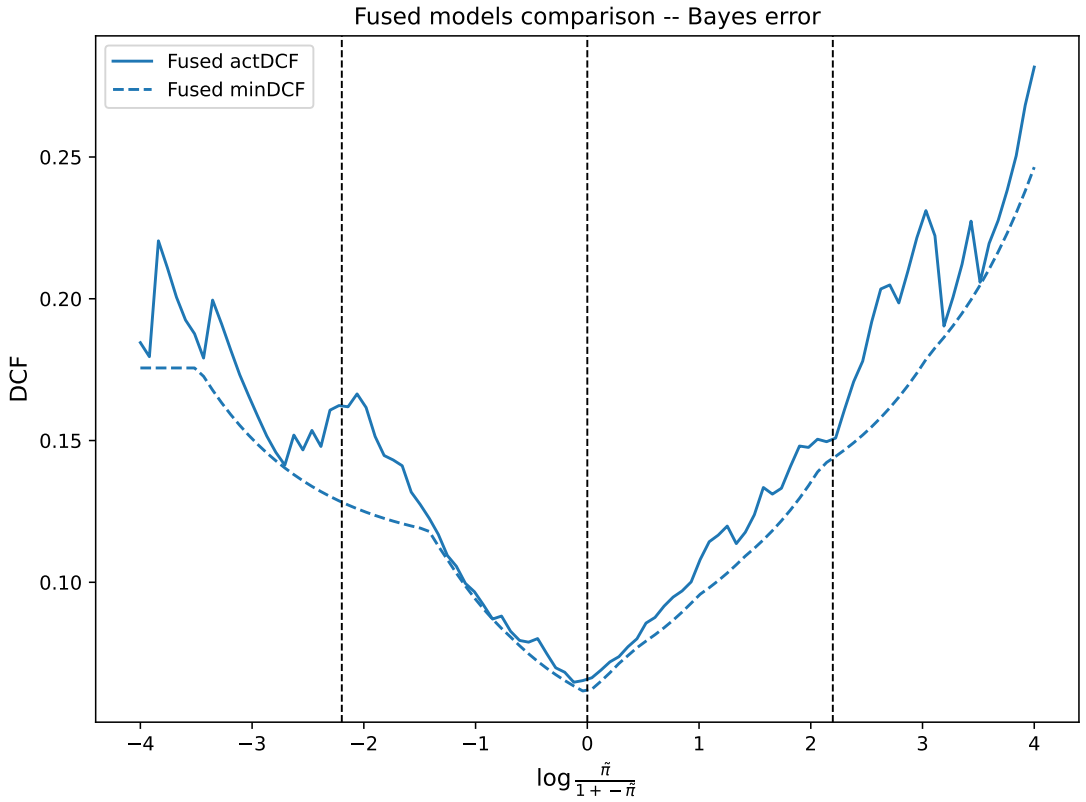

Figure 9.2: Bayes error plot of calibrated best performing models; Validation set

Fusion may lead to a more chaotic Bayes error plot, because we are mixing together different models; however, calibration loss is not very important and the actual DCF measured is still good.

Chapter 10

Final evaluation

The delivered model is the fused version of the three models: Quadratic Logistic Regression, RBF Support Vector Machine and GMM with diagonal covariance matrix. Let's see how it performs with the evaluation set.

Model	Accuracy	actDCF	minDCF
Validation set			
Quad LR	91.97%	0.359	0.351
Poly4 SVM	95.40%	0.277	0.254
RBF SVM	94.93%	0.289	0.262
GMM diag	**96.73%**	0.197	0.178
Fused	96.67%	**0.196**	0.193

Table 10.1: K-fold calibration results on Validation set

In Tab. 10.1, we can see the comparisons with the final models. The fused one is the best because the actual DCF is the lowest, meaning that we are correctly targeting the security application of interest (even if the accuracy is not the best one). In Fig. 10.1 we can also see calibration for all applications of the calibrated models.

After computing analysis also on the evaluation set, we can state that the final choice of the model was pretty good: the fused model is the best performing one, even if a lower minimum DCF can lead to another conclusion: GMM with diagonal covariance matrix could just be better. However, relying on a model that is made over a weighted combination of others seems to be more reliable, because of the different capabilities of each of them.

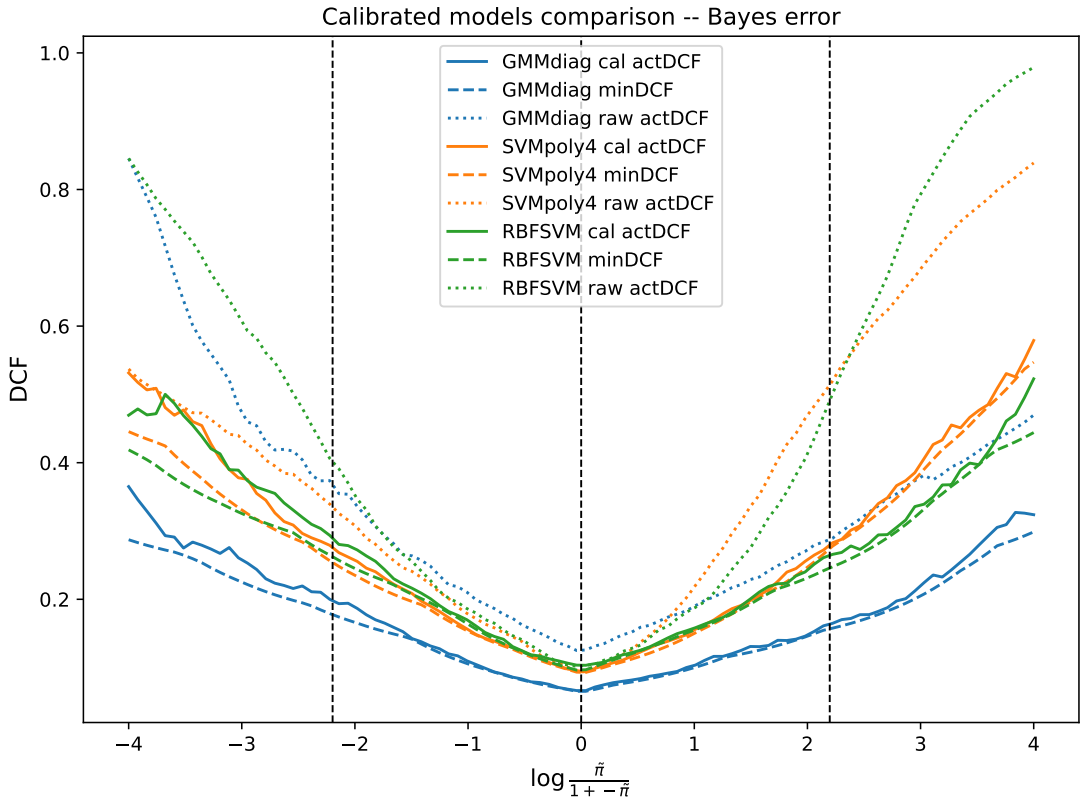

Figure 10.1: Bayes error plot of calibrated best performing models; Evaluation set

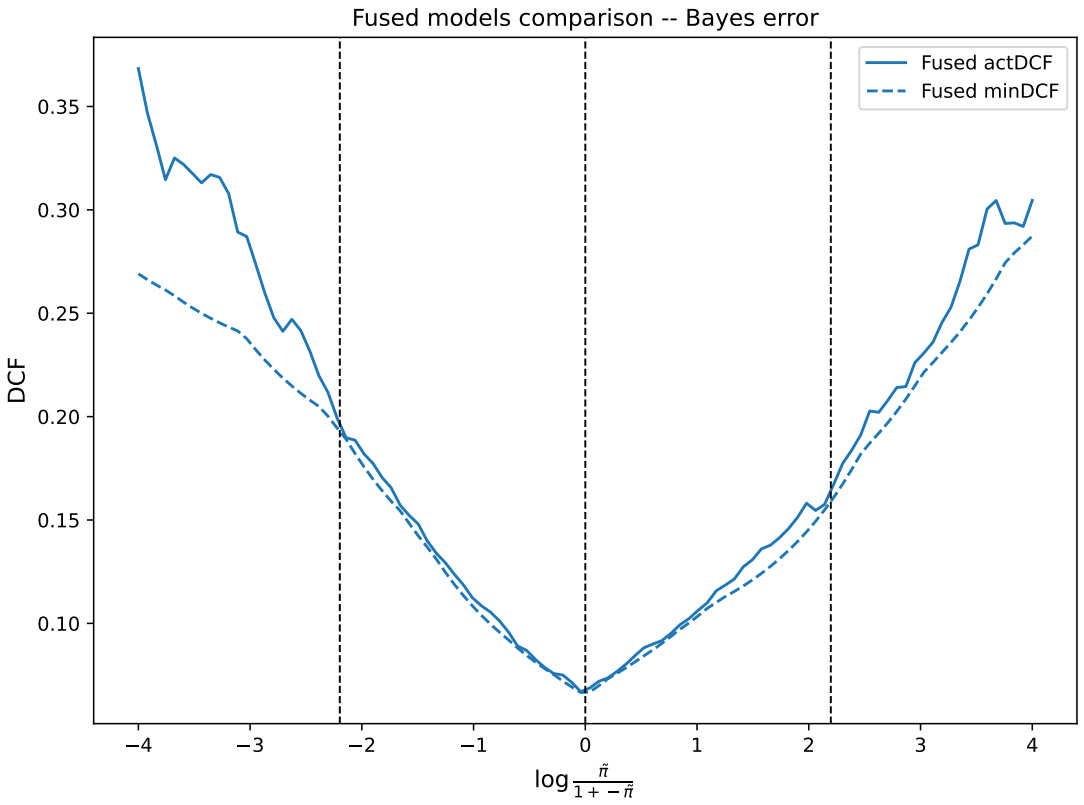

Figure 10.2: Bayes error plot of fused best performing models; Evaluation set