Fingerprint spoofing detection - MLPR

Giacomo Fantino (s310624), Farisan Fekri (s308066)

September 19, 2023

Chapter 1

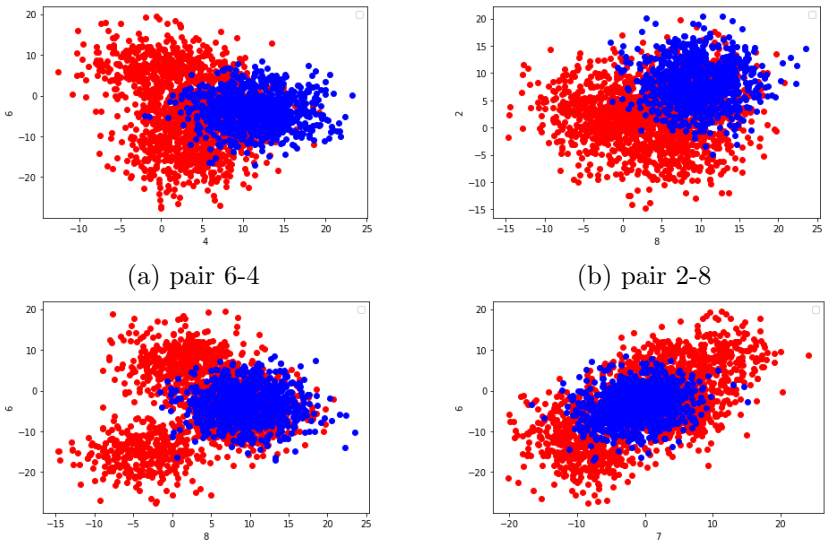
Introduction

1.1 Objective

The goal of the project is the development of a classifier to detect whether a fingerprint image represents an authentic fingerprint or a spoofed fingerprint. This will be done exploiting the most common Machine Learning tools. We will discuss how they perform for the problem we have chosen, explaining pros and cons.

1.2 Overview of the problem

The dataset consists of embeddings extracted from fingerprint images. Each row corresponds to a different embedding and contains 10 features followed by the label (1 for authentic fingerprint, 0 for spoofed fingerprint). The features do not have any particular interpretation. The spoofed fingerprints are generated using one of 6 different spoofing approaches, but that information is not available. The training set consists of 2325 samples, 800 are authentic fingerprint and the remaining 1525 are spoofed. We can thus note that the dataset is imbalanced. The target application considers an application where prior probabilities of authentic and spoofed classes are the same, but labeling a spoofed fingerprint as authentic has a larger cost due to the security vulnerabilities that such errors would create.



(a) pair 6-4 (b) pair 2-8

(c) pair 6-8 (d) pair 6-7

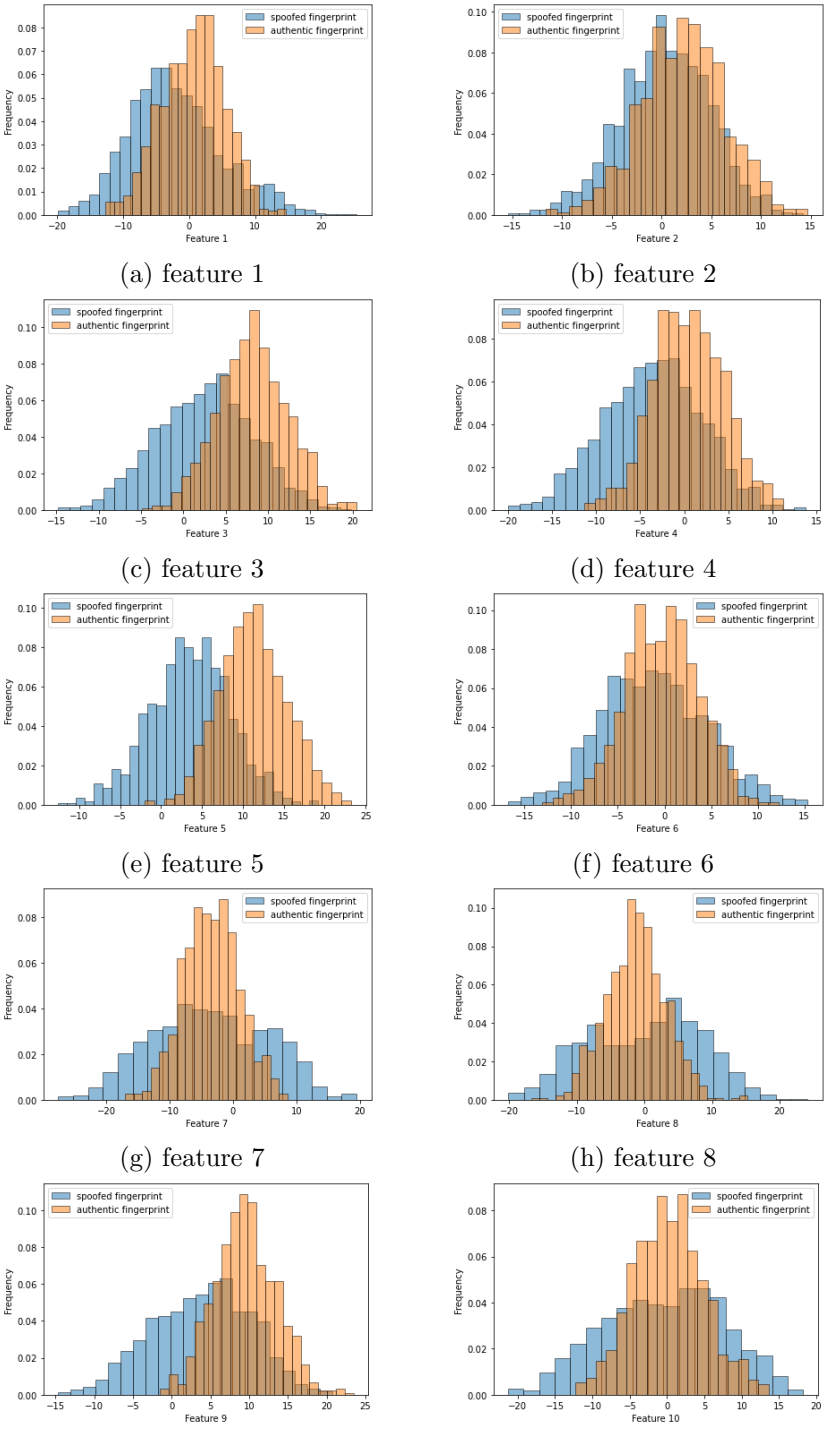
Figure 1.1: Pairs (blue authentic and red spoofed)

1.3 Features Analysis

We can visualize the 10 features using histograms 1.2. We can see that the raw features have an approximated Gaussian distribution for the authentic fingerprints while some features of the spoofed do not.

We can also use the scatter plot to visualize pairs of features to better visualize the correlation and the spread of the samples (we won't show all 100 plots, just a subset to give an idea) 1.1. There are two things that immediately catch the eye: since the spoofed fingerprints are created by different approaches the samples are grouped in what are usually called clusters, while the authentic ones are more compacted. Also looking at the pair 6-7 we can visualize a correlation between the features (especially for the spoofed fingerprints). PCA may remove some features that bring no information but since the number of features is low we can't remove too much or we would end up removing too much information.

To test our last observation we can use the heatmap: we will show a heatmap for all the dataset and one for each class 1.3. Target class features are weakly correlated, while non-target class show a large correlation between some features.



(a) feature 1 (b) feature 2

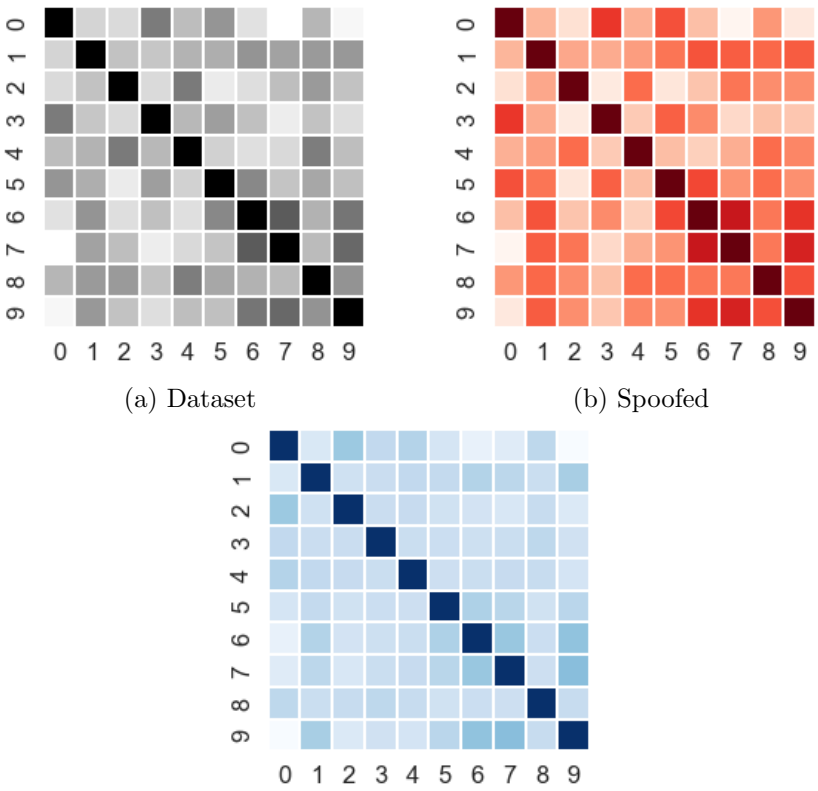
(c) feature 3 (d) feature 4

(e) feature 5 (f) feature 6

(g) feature 7 (h) feature 8

(i) feature 9 (j) feature 10

Figure 1.2: Features



(a) Dataset (b) Spoofed

(c) Authentic

Figure 1.3: Heatmaps

Chapter 2

Validation

Here is a table of all the classifier we have used for this report:

- Generative models
  - Multivariate Gaussian Classifier (we will use MVG for abbreviation)
  - MVG + Diagonal Covariance
  - MVG + Tied Covariance
  - MVG + Diagonal Covariance with Tied Covariance
- Logistic Regression
  - Weighted Linear Logistic Regression
  - Weighted Quadratic Logistic Regression
- Support Vector Machine
- Gaussian Mixture Model

Regarding the validation part we have employed 5-fold cross validation, meaning that after splitting the training set in 5 folds 80% of the training data will be used to train the models and the remaining 20% to validate the hyper parameters.

We have used only PCA to reduce our dimensionality. We were not able to use LDA because the maximum dimension we can reduce is equal to the number of class minus 1. This problem is a binary problem and reduce all the data to one dimension would be useless.

Our working point is ($\pi = 0.5, C_{fn} = 1, C_{fn} = 10$), but we have worked using the **effective** prior probability, where the costs are equal to one. By using the formula we got:

$$\hat{\pi} = \frac{\pi C_{fn}}{\pi C_{fn} + (1-\pi)C_{fp}} = \frac{0.5 \cdot 1}{0.5 \cdot 1 + 0.5 \cdot 10} = \frac{0.5}{5.5} = 0.09 \tag{2.1}$$

Since the cost of mistakenly classify a spoofed sample as authentic is very high the prior probability of authentic samples has dropped.

We have measured the hyper parameters' performance using not only the prior of our application but also a balanced application $\pi = 0.5$ and an unbalanced opposite one with $\pi = 0.9$. All the plot we will show for the classifiers are related to the prior of our application.

2.1 Multivariate Gaussian Classifiers

The first set of classifiers we will analyze is Multivariate Gaussian Classifiers (MVG). Different classifiers use different covariance matrices: Full covariances, Tied covariance and Diagonal covariance. These models are generative, meaning we have computed:

$$P(C_t = c_t | X_t = x_t) = f_{X|C}(x_t|c_t)P(c_t) \tag{2.2}$$

And by using the Gaussian distribution:

$$f_{X|C}(x_t|c_t) \sim \mathcal{N}(x_t|\mu_c, \Sigma_c) \tag{2.3}$$

Also Tied covariance suppose that the covariance matrix, Σ, it's the same for all classes, while diagonal covariance use the naive hypothesis, meaning Σ_c will be a diagonals matrix. From what we have seen in the feature analysis part we expect that the MVG model will outperform the other three, since we have seen correlation between features and different spread between the two classes, but we will still analyze all classifiers.

2.1.1 Results

We can start with the effective prior:

	no PCA	PCA (m = 9)	PCA (m = 8)	PCA (m = 7)
MVG	0.336	0.348	0.34	**0.33**
Naive MVG	0.464	**0.367**	0.37	0.374
Tied MVG	**0.475**	0.482	0.489	0.480
Naive Tied MVG	0.55	0.556	0.552	**0.55**

And for the balanced prior and the one favoring the authentic fingerprints:

	no PCA	PCA (m = 9)	PCA (m = 8)	PCA (m = 7)
MVG	0.104	0.107	0.106	**0.104**
Naive MVG	0.137	**0.105**	0.106	0.106
Tied MVG	0.173	0.170	0.171	**0.169**
Naive Tied MVG	**0.186**	0.189	0.188	0.19

	no PCA	PCA (m = 9)	PCA (m = 8)	PCA (m = 7)
MVG	0.190	0.190	0.187	**0.187**
Naive MVG	0.229	0.18	**0.177**	0.179
Tied MVG	0.402	**0.394**	0.403	0.4
Naive Tied MVG	**0.373**	0.388	0.387	0.389

Lowering our dimensionality below 7 bring worses results for all classifier except for MVG: the best dimensionality is 6 with a minimum DCF of **0.329** for our application.

As expected PCA improve the results for models with the Naive hypothesis and helps our MVG classifier to not over fit. This tells us that the features that aren't removed by PCA have a very low correlation. Lastly the linear separation rule of tied models make those the worst classifiers.

The MVG is the better of three as anticipated but since our spoofed fingerprints don't really follow a Gaussian distribution we can do better.

2.2 Logistic Regression

The next classifier we have used is the Logistic Regression prior-weighted. This model is a discriminant model, meaning it directly compute:

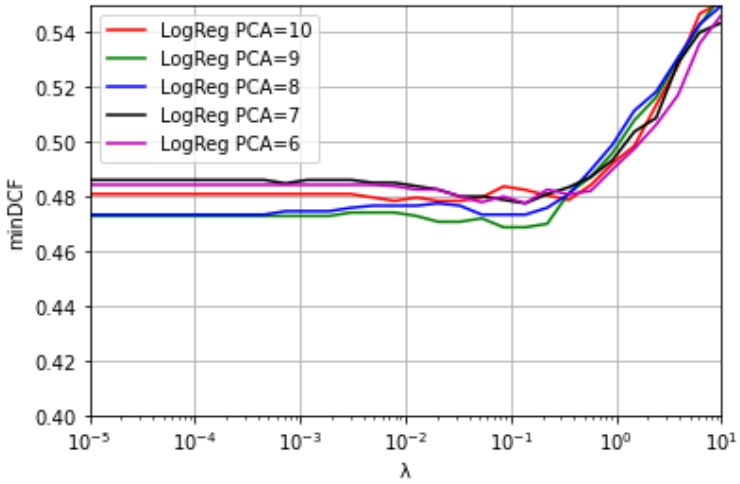


Figure 2.1: Linear Logistic Regression

$$P(C_t = c_t | X_t = x_t) = \sigma(w^T x + b) \tag{2.4}$$

For computing the parameters w and b the training step is minimizing the following function, which include the prior weighting and the λ term for the norm of w:

$$R(w,b) = \frac{\lambda}{2}||w||^2 + \frac{\pi_T}{n_T}\sum_{i|z_i=1} l(z_i s_i) + \frac{1-\pi_T}{n_F}\sum_{i|z_i=-1} l(z_i s_i) \tag{2.5}$$

We won't limit our self to linear separating rules, so we will use the expanded feature space, defined as:

$$\phi(x) = \begin{bmatrix} vec(xx^T) \\ x \end{bmatrix} \tag{2.6}$$

We can thus train the LR model defined above, but this time using the feature vectors $\phi(x)$ rather than x.

We can start by seeing the results of linear logistic regression, even if we expect the results to be worse compared to MVG: 2.1.

And our optimal hyper parameters are $\lambda = 0.0853$ and PCA $= 9$ with a minimum DCF of 0.469. Since our classes are not linearly separable, higher values for λ just create underfitting and higher cost but lower values don't make a difference (from 10^{-5} to 10^{-3} the cost is roughly the same).

We have done this training using for the prior the effective prior of our application. We also tried the **empirical prior** (number of authentic fingerprints divided by the total) and a balanced prior (where $\pi = 0.5$). Here the comparison:

	minDCF
effective prior	**0.469**
balanced prior	0.479
empirical prior	0.472

The other priors have a higher cost.

2.2.1 Quadratic Logistic Regression

Since the cost of the linear logistic regression is higher compared to MVG we will move on to Quadratic Logistic Regression: we tried different dimensionality: 2.2. In the following table for each dimensionality we show the minimum DCF using the different priors:

	no PCA	PCA (m = 8)	PCA (m = 7)	PCA (m = 6)
Quad Log Reg ($\pi = 0.09$)	0.305	0.289	0.287	**0.283**
Quad Log Reg ($\pi = 0.5$)	0.101	0.1	0.096	**0.094**
Quad Log Reg ($\pi = 0.9$)	0.288	0.265	0.216	**0.192**

Since our results seem better by reducing the dimensionality we have tried applying PCA more aggressively. We have found that with a dimensionality equal to 5 the minimum cost becomes:

	PCA (m = 5)
Quad Log Reg ($\pi = 0.09$)	0.276
Quad Log Reg ($\pi = 0.5$)	0.095
Quad Log Reg ($\pi = 0.9$)	0.21

Changing the prior used for the training gave us interesting results: the best cost is obtained by using **the empirical prior of the training set** and a dimensionality equal to 6:

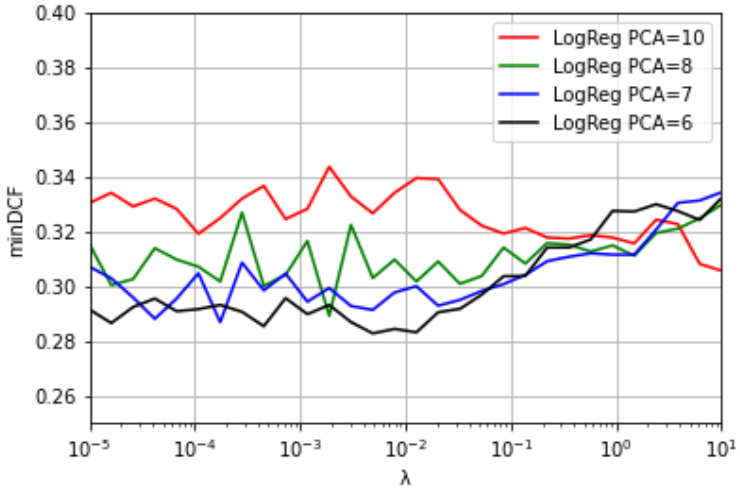


Figure 2.2: Quadratic Logistic Regression

	PCA (m = 6)
Quad Log Reg ($\pi = 0.09$)	0.259
Quad Log Reg ($\pi = 0.5$)	0.089
Quad Log Reg ($\pi = 0.9$)	0.181

We tried adding a z-normalization that worse the result: for our application the cost increased to 0.303.

2.3 Support Vector Machine

We can now move to SVM models. This model classifies by using a discriminant non probabilistic approach, so for each sample we get a score:

$$s(x) = \sum_{i=1}^{N} \alpha_i z_i x_i^T x + b \tag{2.7}$$

The training samples where α_i is not equal to zero are called support vector.

We will start to analyze the linear model, but since the linear separation hasn't worked for our previous models we don't expect to see greater results. We need to estimate an hyper parameter C, which measure how much the cost of samples inside the margin affect the minimization.

As predicted our Linear SVM has a higher cost, which is similar to the linear Logistic Regression 2.3. Our best value is with a dimensionality equal to 8 where

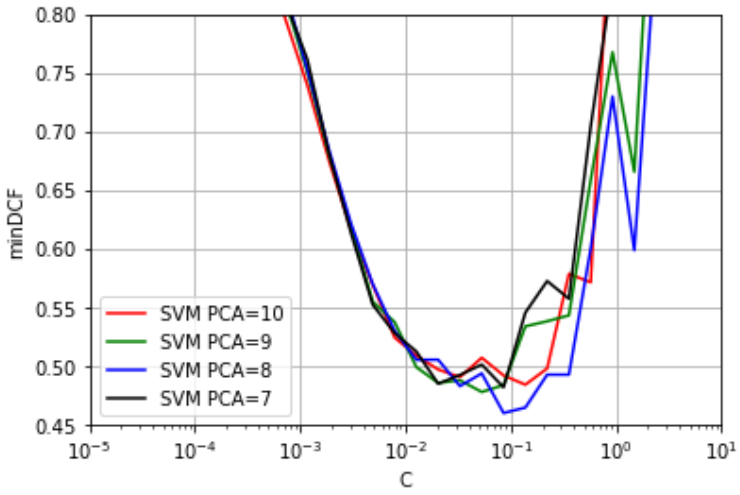


Figure 2.3: Linear SVM

the minimum cost is equal to 0.46.

So we can start using SVM with a kernel function.

2.3.1 Polynomial kernel

The first one we have tried is the model with a polynomial degree, using both degree of 2 and 3. We tried computing without PCA and with the three best dimensionality for our quadratic classifiers: 8, 7 and 6.

Here is the result with a degree equal to 2:

	no PCA	PCA (m = 8)	PCA (m = 7)	PCA (m = 6)
SVM Poly(2) ($\pi = 0.09$)	0.321	0.296	**0.286**	0.303
SVM Poly(2) ($\pi = 0.5$)	0.1	0.102	0.103	**0.095**
SVM Poly(2) ($\pi = 0.9$)	0.26	0.253	**0.238**	0.241

The best dimensionality for the two degree is 7 with a C value of 0.0008 2.4.

Here a comparison between a degree equal to 2, 3 and a degree of 2 with z-normalization:

	PCA (m = 7)
SVM Poly(2)	0.286
SVM Poly(3)	0.357
SVM Poly(2) z-norm	**0.282**

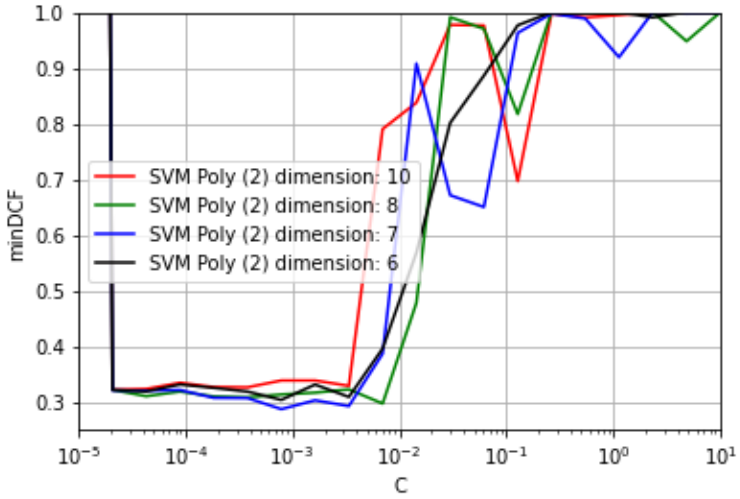


Figure 2.4: Svm polynomial degree 2

It's clear that in this case the z-normalization is effective at reducing our minimum cost.

2.3.2 Radial basis kernel

Let's now try the SVM with Radial basis. We started by estimating the two hyper parameters (C and γ) and then applying PCA and z-normalization. We used this approach since the estimation of three hyper parameters may requires a lot of time. Here our results without PCA:

	no PCA
SVM RBF (log γ =-4.0)	0.42
SVM RBF (log γ =-5.0)	0.322
SVM RBF (log γ =-6.0)	0.299

As we can also see by 2.5 it seems that reducing the value of log γ brings better results. We have found that for dimensionality 10 the best result is log γ =-7.0 with C between 0 and 10. We now want to find the optimal value C and dimensionality for log γ =-7.0. By using dimensionality reduction we got:

	no PCA	PCA (m = 8)	PCA (m = 7)	PCA (m = 6)
SVM RBF ($\pi = 0.09$)	0.284	**0.272**	0.274	0.276
SVM RBF ($\pi = 0.5$)	0.083	0.094	0.088	**0.083**
SVM RBF ($\pi = 0.9$)	0.171	0.166	0.160	**0.159**

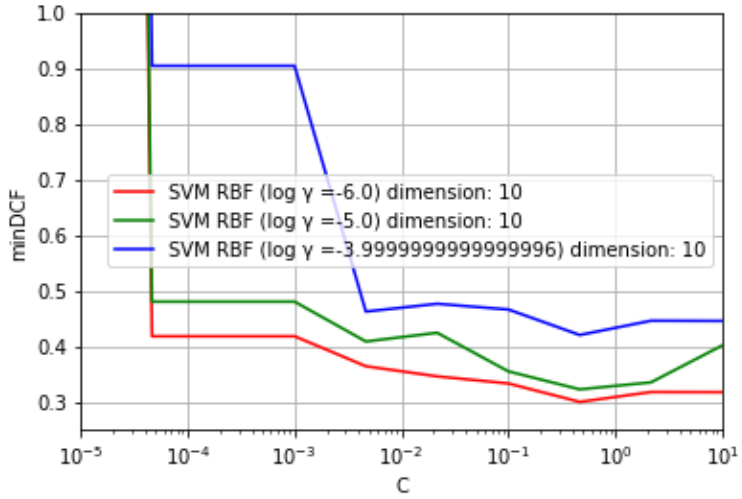


Figure 2.5: SVM RBF different Gammas

Each application has a different optimal set of hyper parameters so we need to find a combination of PCA and C value that is a good compromise between the three. Since it's in the middle we chose to set the dimensionality to 7 and now we need to estimate a good C value:

C	SVM RBF ($\pi = 0.09$)	SVM RBF ($\pi = 0.5$)	SVM RBF ($\pi = 0.9$)
1	0.287	0.097	0.175
2	0.277	0.096	0.171
3	0.281	0.093	**0.166**
4	**0.272**	**0.088**	**0.166**
4.5	0.273	0.089	0.168
5	0.273	0.09	0.167

With a dimensionality equal to 7 and C equal to 4 the SVM classifier with a radial basis kernel seems the best SVM classifier we can have. Applying z-normalization in this case worse the results: for the dimensionality 7 the cost is equal to 0.458.

2.4 Gaussian mixture model

For this model instead of computing a single Gaussian distribution for each class we compute a number G of Gaussians. For our dataset we expect to see good results, for sure better than MVG. The reason comes from the class of spoofed

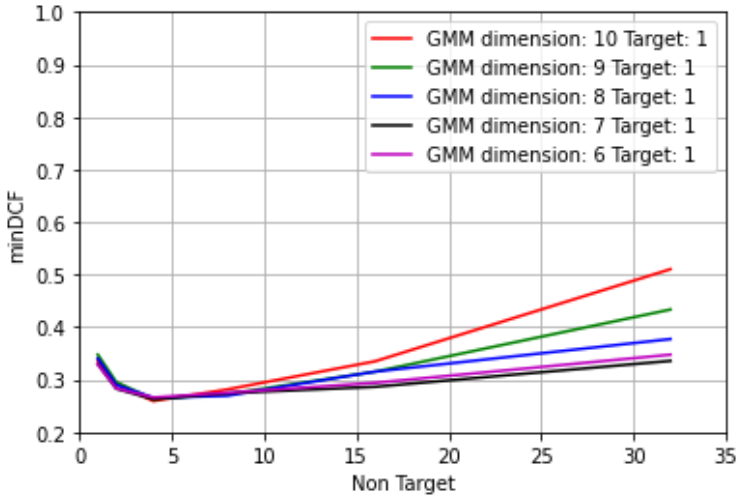


Figure 2.6: GMM K=1 with dimensionality

fingerprint: being generated by different technique we expect to be able to capture each one by a Gaussian distribution. We nonetheless tried different numbers for both authentic and spoofed fingerprints and then we will try to use diagonal and tied GMM.

We started by computing the cost for the application's prior for different number of Gaussians without applying PCA. We can see the best number of Gaussians for the spoofed fingerprints:

	minDCF	num. Non Target
GMM Target = 1	**0.259**	4
GMM Target = 2	0.275	4
GMM Target = 4	0.28	4

Our authentic fingerprints can be already modeled with a single Gaussian, an higher number will increase our error. With 4 Gaussians for the spoofed fingerprints we are able to classify those with a low cost. We now need to test the dimensionality reduction. These are the results we got: 2.6. The results are pretty similar to each other (all the minimum costs are with 4 component for the non target), but the best result is by remaining with a dimensionality equal to 10:

	no PCA
GMM ($\pi = 0.09$)	0.259
GMM ($\pi = 0.5$)	0.075
GMM ($\pi = 0.9$)	0.143

We can now use these results when applying diagonal and tied covariance matrices. We tried different combinations for the non target and the diagonalized covariance for the target (using tied would be useless since we have a single Gaussian for the target class):

covariance Target	covariance non-Target	($\pi = 0.09$)	($\pi = 0.5$)	($\pi = 0.9$)
full	full	0.259	**0.075**	**0.143**
full	diag	**0.257**	0.077	0.157
full	tied	0.269	0.082	0.153
diag	full	0.288	0.083	0.169
diag	diag	0.272	0.087	0.163
diag	tied	0.283	0.082	0.175

The diagonalization hypothesis for the non target class is giving the lower cost for the target application but for the other two priors the full covariance is better. We have decided to keep the diagonalized covariance matrix for the non target class.

At this point we have set the covariance matrices of our model and we have tried changing the dimensionality to see if the results got better for the applications:

	no PCA	PCA (m = 9)	PCA (m = 8)	PCA (m = 7)
GMM ($\pi = 0.09$)	0.257	0.26	**0.253**	0.268
GMM ($\pi = 0.5$)	**0.077**	0.081	0.081	0.080
GMM ($\pi = 0.9$)	**0.157**	0.165	0.162	0.165

We have decide to accept the worse result for the unbalanced application favoring the authentic fingerprints and to use GMM with 1 component for target full covariance and 4 non target component diagonalized and a dimensionality equal to 8.

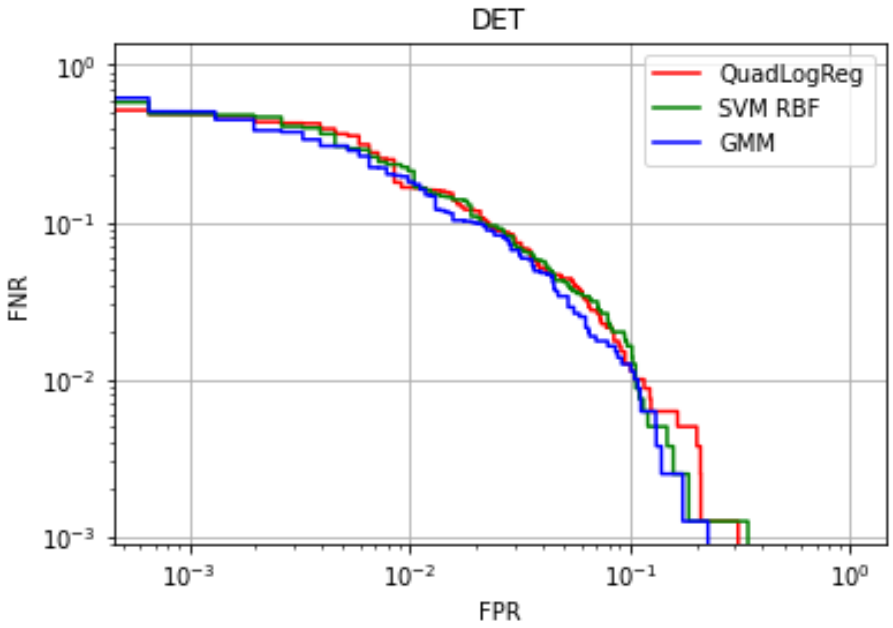


Figure 2.7: DET Plot

2.5 Final consideration

We have summarized our results in this table:

Classifier	($\pi = 0.09$)	($\pi = 0.5$)	($\pi = 0.9$)
Weighted Quad Log Reg PCA = 6 $\lambda = 0.0016$	0.259	0.089	0.1813
SVM RBF PCA = 7 C = 4 log γ =-7	0.272	0.088	0.166
GMM PCA = 8 Tar: 1 full Non-tar: 4 diag	**0.253**	**0.081**	**0.162**

We haven't considered quadratic SVM with polynomial kernel since the version with radial basis kernel has a better minimum cost.

We now need to compare these classifiers: we will measure them in terms of **DET plot** (we will see the false positive and false negative ratios for different thresholds) and **Bayes error plot** (where we compute the actual and the minimum costs for different thresholds).

We can start with the DET plot (scaled in a logarithm scale) for our three models: 2.7. In general GMM is the best classifier, but the results are similar. Only thing to notice is that the Quadratic Logistic Regression classifier has a higher FPR for some thresholds.

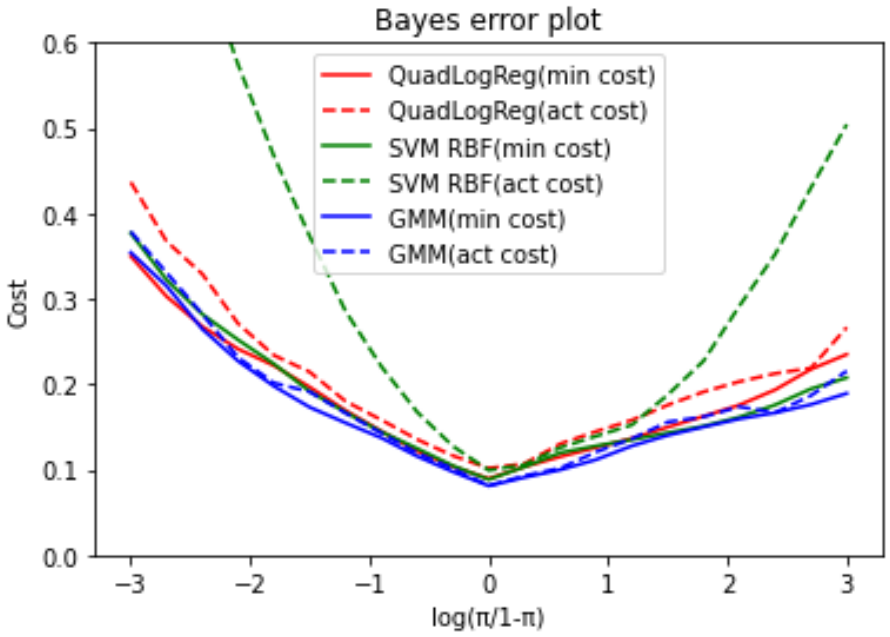


Figure 2.8: Bayes Error Plot

We now need to understand how much mis-calibration is effecting our models by comparing the actual cost with the minimum cost for different applications: 2.8.

The GMM classifier has a lower minimum cost for almost all applications compared to the other models (but it was predictable since it had a lower cost for the three applications we considered). The minimum costs are similar for the other two classifiers but the SVM RBF classifier is very mis-calibrated (due to the non-probabilistic interpretation). The GMM and Quad LogReg are almost well-calibrated. Here the results:

Classifier	$(\pi = 0.09)$		$(\pi = 0.5)$		$(\pi = 0.9)$	
	minDCF	actDCF	minDCF	actDCF	minDCF	actDCF
Quad LogReg	0.259	0.311	0.089	0.102	0.181	0.195
SVM	0.272	0.656	0.088	0.099	0.166	0.315
GMM	0.253	**0.266**	0.081	**0.081**	0.162	**0.186**

Looking at the actual DCF the best model is the GMM : it's well-calibrated and since it has the lowest minimum cost also the actual cost is good. We now need to see what happens by applying a calibration to the scores of the classifiers.

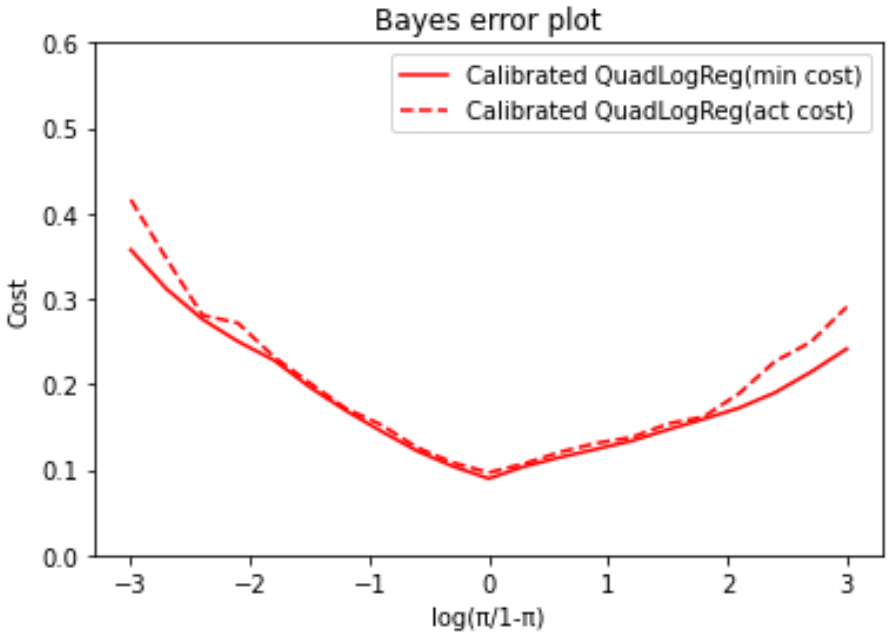


Figure 2.9: Calibrated Quadratic logistic regression

2.5.1 Calibration

We have applied a transformation function to calibrate the scores of the classifiers to see if some classifiers could benefit from it. For the transformation function we have used the weighted linear logistic regression classifier. We have employed for each classifier a K-fold approach for both getting the scores of the model and for training the Logistic regression. This means that for each fold the calibrating classifier will be slightly different and so the minimum DCF may be a little bit different compared to the original non-calibrated classifier.

Let's start with the quadratic logistic regression. We can visualize with bayes error plot the new costs: 2.9.

And the costs for our applications:

Application	minDCF	non-cal actDCF	actDCF
$\pi = 0.09$	0.269	0.311	0.274
$\pi = 0.5$	0.09	0.102	0.096
$\pi = 0.9$	0.178	0.195	0.202

Since the Quadratic Logistic Regression classifier was already almost well-calibrated the results aren't very different. We have lowered the cost for the target application while increasing the cost for the other unbalanced application.

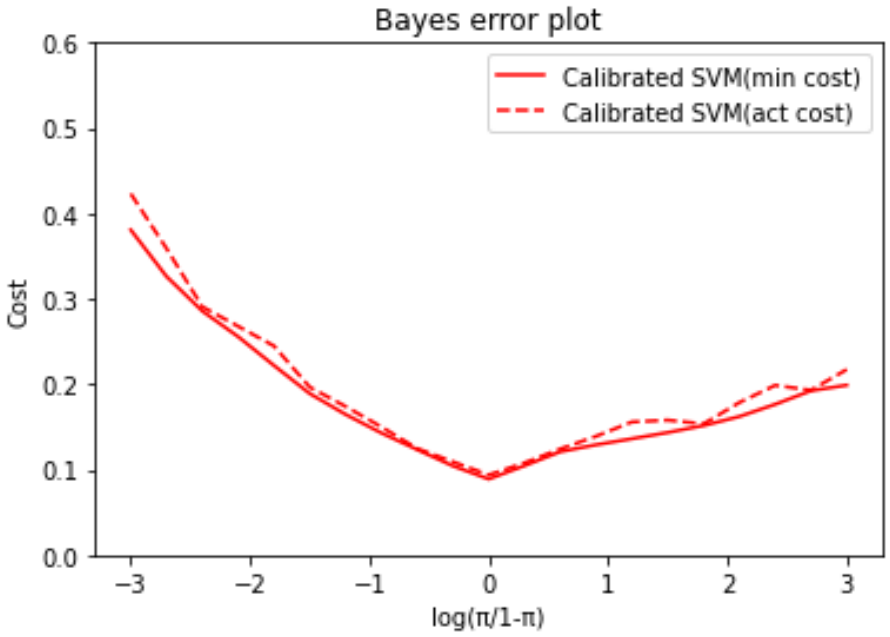


Figure 2.10: Calibrated SVM Radial basis kernel

Here is the results for SVM: 2.10

Application	minDCF	non-cal actDCF	actDCF
$\pi = 0.09$	0.276	0.656	0.281
$\pi = 0.5$	0.089	0.099	0.094
$\pi = 0.9$	0.166	0.315	0.187

The SVM was already well-calibrated in the balanced application but miscalibrated in the unbalanced applications. We have reduced that effect.

We have also tried applying it to GMM to see the differences with the original classifier: 2.11.

Application	minDCF	non-cal actDCF	actDCF
$\pi = 0.09$	0.251	0.266	0.274
$\pi = 0.5$	0.08	0.081	0.081
$\pi = 0.9$	0.163	0.186	0.168

As expected the results are very similar (only the unbalanced application favoring the authentic fingerprints got a lower cost).

We have trained **fusion models**: the calibrated scores are computed by using the scores of more models together. We have tried using the best model (GMM) with another classifier and also using all three classifiers.

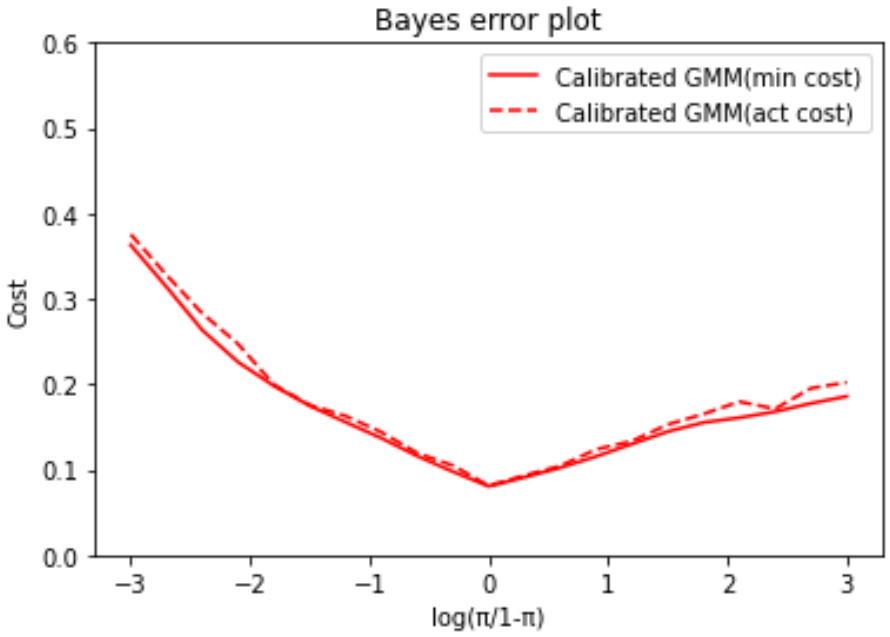


Figure 2.11: Calibrated GMM

Here we have sum up the actual cost for all the classifiers:

Classifier	$(\pi = 0.09)$	$(\pi = 0.5)$	$(\pi = 0.9)$
Cal Quad LogReg	0.274	0.096	0.202
Cal SVM RBF	0.281	0.094	0.187
GMM	**0.266**	**0.081**	0.186
Cal GMM	0.274	**0.081**	0.168
Fusion SVM GMM	0.278	0.086	**0.166**
Fusion QLogReg GMM	0.28	0.086	0.178
Fusion All	0.274	0.087	**0.166**

Fusion models bring better results only for application with an unbalanced prior favoring authentic fingerprints. For our application and for the balanced one the choice is between the non calibrated and the calibrated GMM. We can see from 2.12 how the two classifiers are really close. So we decided to choose by taking into consideration our target application: the GMM classifier without calibration will be our final model.

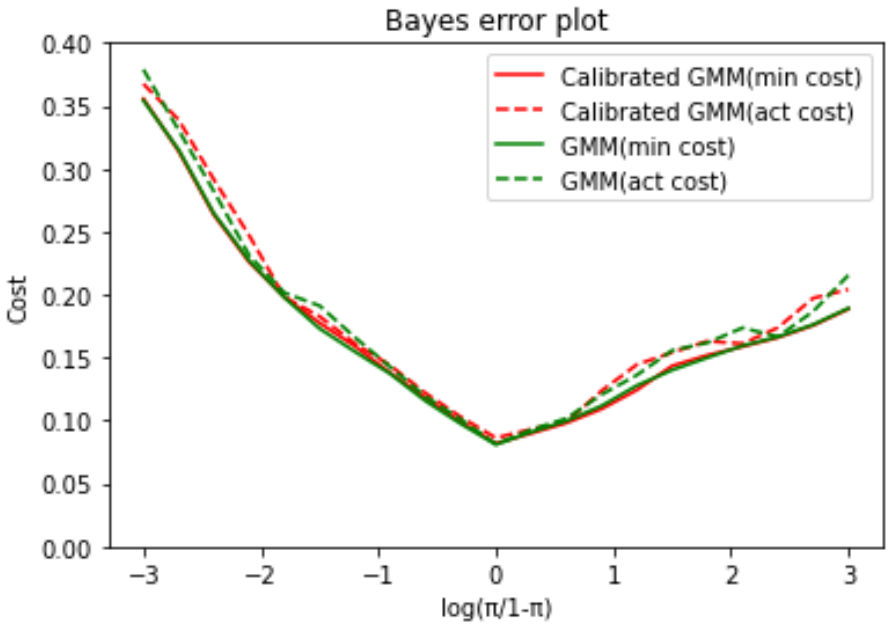


Figure 2.12: Comparison between Calibrated GMM and GMM

Chapter 3

Evaluation

We will now employ our test set to verify our results from the validation phase and understand if the GMM model actually is the best one. We didn't apply anymore K-fold but we used the entire train file as training set and the test file as test set. We have also employed calibration by splitting the training set into two subsets: the first one (around 80% of the original set) is used to train the miscalibrated model while the remaining part is used to train the logistic regression model to calibrate the scores.

3.1 Multivariate Gaussian Models

The evaluation phase gave the following results for the target prior:

	no PCA	PCA (m = 9)	PCA (m = 8)	PCA (m = 7)
MVG	0.277	0.273	**0.272**	0.274
Naive MVG	0.354	**0.311**	0.317	0.314
Tied MVG	**0.456**	0.463	0.458	0.461
Naive Tied MVG	**0.482**	0.485	0.486	0.484

The results are slightly better compared to the validation set and we have confirmed what we already seen: the multivariate model without any assumption is the best model, since we need a quadratic separation rule. Also in this case PCA helps at lowering the cost.

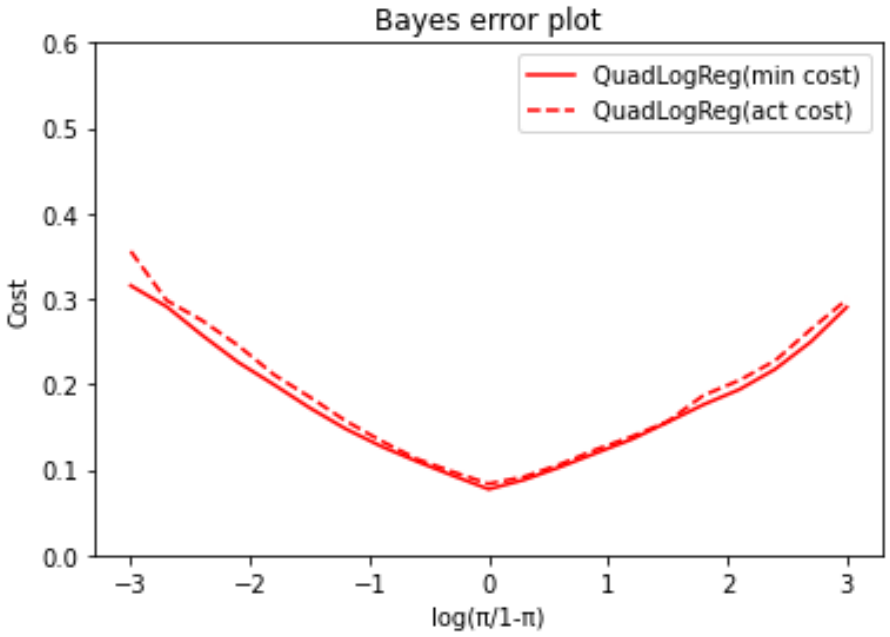


Figure 3.1: Bayes cost in the evaluation set for calibrated Quadratic Logistic Regression

3.2 Logistic regression

Since we already seen that a linear model is not the optimal choice for this dataset we will work with the quadratic logistic regression.

Let's start by analyzing our best model from validation (without changing the hyper parameters): we will compute minimum cost and actual cost using the evaluation set. Note that since the calibration was useful in the validation we also employed it in the evaluation.

Prior	minDCF_{DEV}	actDCF_{DEV}	minDCF_{EVAL}	actDCF_{EVAL}
$\pi = 0.09$	0.269	0.274	0.247	0.268
$\pi = 0.5$	0.09	0.096	0.077	0.087
$\pi = 0.9$	0.178	0.202	0.201	0.205

As already seen with the MVG classifier the costs are lower. As we can see in 3.1 the Quad LogReg model suffers a bit of mis-calibration around our target prior. Now we need to check if the hyper parameters we have chosen are optimal or not for the evaluation set.

We have found the optimal dimensionality to be 6 and $\lambda = 0.0038$ while training the model with an empirical prior. These are the results comparing the

hyper parameters chosen in the validation set and the optimal for the evaluation set:

	($\pi = 0.09$)	($\pi = 0.5$)	($\pi = 0.9$)
Chosen configuration	0.247	0.077	**0.201**
Optimal configuration	**0.239**	**0.076**	0.210

Our choice provided sub-optimal results for the target prior, but the results are very close.

3.3 Support vector machine

We will skip to RBF SVM since it was one of the best models (the linear classifier won't be good as we have already seen and the quadratic SVM wasn't much efficient).

We will start by analyzing our chosen model as we did before (also in this case including calibration):

Prior	minDCF$_{DEV}$	actDCF$_{DEV}$	minDCF$_{EVAL}$	actDCF$_{EVAL}$
$\pi = 0.09$	0.276	0.281	0.239	0.254
$\pi = 0.5$	0.089	0.094	0.076	0.079
$\pi = 0.9$	0.166	0.187	0.212	0.218

Just like in the Quadratic LogReg model we have a lower cost and there is some mis-calibration around our target prior.

By using the same approach for the validation we can try to estimate a new set of hyper parameters for the classifier (we will start by analyzing γ and then check for C and dimensionality).

We have that with the a value of log γ = -6 and dimensionality 7 the cost is lower for the target prior:

	($\pi = 0.09$)	($\pi = 0.5$)	($\pi = 0.9$)
Chosen configuration	0.239	**0.076**	**0.212**
Optimal configuration	**0.221**	0.078	0.215

Also in this case our hyper parameters were sub optimal for the target prior, but still very close.

3.4 Gaussian Mixture Model

For the GMM we have seen how applying calibration affects our models. We have decided to employ both approaches and see if choosing for our final model to not use calibration was a good idea or not.

Let's start with the model without calibration:

Prior	minDCF_{DEV}	actDCF_{DEV}	minDCF_{EVAL}	actDCF_{EVAL}
$\pi = 0.09$	0.253	0.266	0.23	0.236
$\pi = 0.5$	0.081	0.081	0.076	0.077
$\pi = 0.9$	0.162	0.186	0.18	0.187

The GMM is able to bring a lower minimum cost like the other classifiers and also to remains well calibrated.

We can also checks what happens if we calibrate the scores. We are training the GMM with less data, so we need to see if the calibrate scores are able to re-balance and bring better results nonetheless. Here the comparison with the calibrated GMM from validation:

Prior	minDCF_{DEV}	actDCF_{DEV}	minDCF_{EVAL}	actDCF_{EVAL}
$\pi = 0.09$	0.253	0.274	0.228	0.26
$\pi = 0.5$	0.081	0.081	0.077	0.081
$\pi = 0.9$	0.162	0.168	0.18	0.185

It looks like our choice of not applying calibration to GMM was correct: for the target application and the balanced application the actual cost has increased (and the minimum cost has remained almost the same).

We can now try to estimate optimal parameters for GMM. After checking that the choice of 1 Gaussian for the target class and 4 Gaussians for the other class is the best one we can start applying PCA. Here we have the results for the different types of covariance matrices:

covariance Target	covariance non-Target	($\pi = 0.09$)	Best PCA
full	full	**0.218**	7
full	diag	0.23	7
full	tied	0.229	8
diag	full	0.234	9
diag	diag	0.242	7
diag	tied	0.235	9

And comparing by our chosen configuration using Full covariance for target and Diagonalized for non target:

	$(\pi = 0.09)$	$(\pi = 0.5)$	$(\pi = 0.9)$
Chosen configuration	0.228	0.077	0.18
Optimal configuration	**0.218**	**0.071**	**0.165**

We can immediately see that the chosen configuration from the validation set wasn't optimal and has an increased cost.

3.5 Summarizing

We can now compare our models with the chosen configuration and their version with an optimal configuration (and calibrated scores for Quad LogReg and SVM) on the evaluation set and then use the fusion models.

We want to analyze if our decisions were good and if the GMM non calibrated model can be considered the final model. We will consider only the target application. Since we are now applying calibration to some models the minimum cost for the optimal parameters may change since we are using less data.

Classifier	minDCF_{chosen}	actDCF_{chosen}	$\text{minDCF}_{optmial}$	$\text{actDCF}_{optmial}$
Quad LogReg	0.247	0.268	0.245	0.27
SVM RBF	0.239	0.254	0.242	0.266
GMM	**0.23**	**0.236**	**0.218**	**0.22**
Fusion SVM GMM	0.234	0.268	0.231	0.251
Fusion QLogReg GMM	0.243	0.257	0.231	0.245
Fusion All	0.243	0.253	0.232	0.254

We can see for the Logistic Regression that the optimal hyper parameters in the area of the target application have some mis-calibration costs (also since the model had less data to train the minimum cost is a little bit higher) 3.2.

Also SVM is suffering a lot of mis-calibration around the target prior (in general it's very mis-calibrated for unbalanced priors) and training it with less data because of calibration increased the minimum cost (maybe we got unlucky and removed some support vectors).

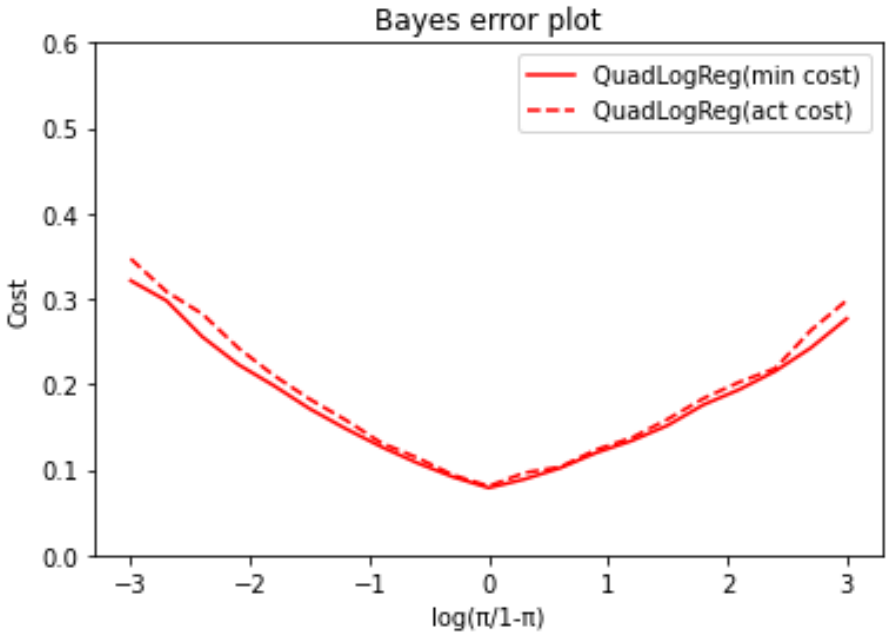


Figure 3.2: Bayes cost in the evaluation set for optimal Quadratic Logistic Regression

With GMM we have already seen that the choice of hyper parameters was sub-optmial, and thus our cost could have been lower. The fusion models act exactly like in the validation phase: the results are almost the average between GMM and the other models.

Summing up we can say that our choice of GMM and the hyper parameters was good, but because in the validation phase we chose sub-optimal hyper parameters we could have better results and a lower cost.