Politecnico di Torino

Machine Learning and pattern recognition
01UDNOV

Master's Degree in Computer Engineering

Fingerprint spoofing detection

Project Report

Author:
Bálint Bujtor (s310419)

Professor:
Sandro Cumani

Contents

CHAPTER 1

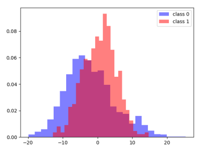
Introduction and preliminary analysis

The goal of the project is the development of a classifier that is capable of detecting whether a fingerprint is authentic or not. The dataset that was given as the basis for the project contains embeddings of fingerprints. An embedding is a low-dimensional representation of an image that is obtained by mapping images to a low-dimensional manifold.

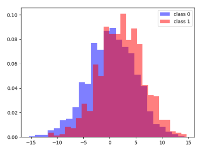
1.1 Analysis of the dataset

The dataset's embeddings are 10-dimensional, continuous-valued vectors, belonging to either the authentic (label 1) or the spoofed (label 0) class. For this project, synthetic data was generated, hence they do not have any particular interpretation. The datasets are imbalanced, with the spoofed fingerprint class having significantly more samples. The spoofed fingerprint samples belong to one of 6 possible sub-classes, corresponding to different spoofing approaches, but the actual spoofing method (sub-class label) is not available. 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. The target application working point is therefore defined by the triplet $\pi_n = 0.5, C_{fn} = 1, C_{fp} = 10$.

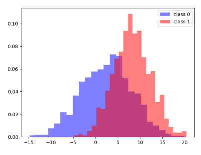
I start the analysis of the dataset by taking a glance at the dataset's features:



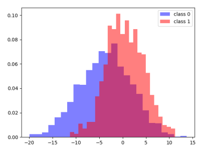
(a) Feature 1



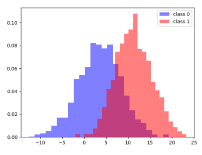
(b) Feature 2



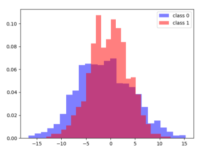
(c) Feature 3



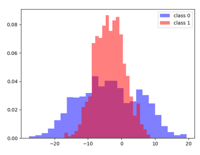
(d) Feature 4



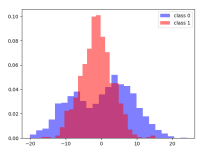
(e) Feature 5



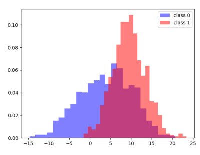
(f) Feature 6



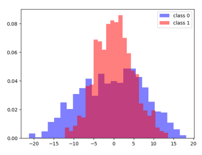
(g) Feature 7



(h) Feature 8



(i) Feature 9



(j) Feature 10

Figure 1.1: Histograms of the dataset features

As it can be seen, some features are strongly correlated, like 1.1b, and there are others, where a relatively good linear separator can be applied 1.1e. We can delve deeper into the details of the dataset by checking the two main principal components of the dataset and its scatter plot. This can give us an initial idea of what kind of classifiers may work well for the dataset.

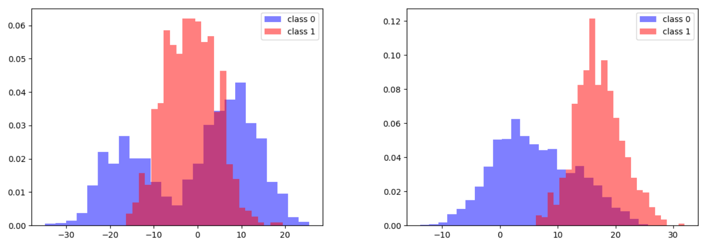


Figure 1.2: Histogram of the principal directions

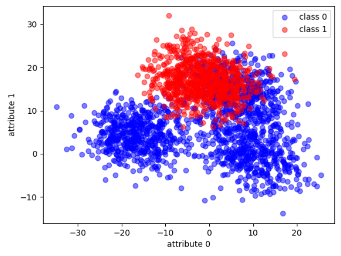


Figure 1.3: Scatter plot of the Principal directions

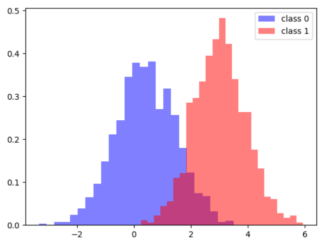
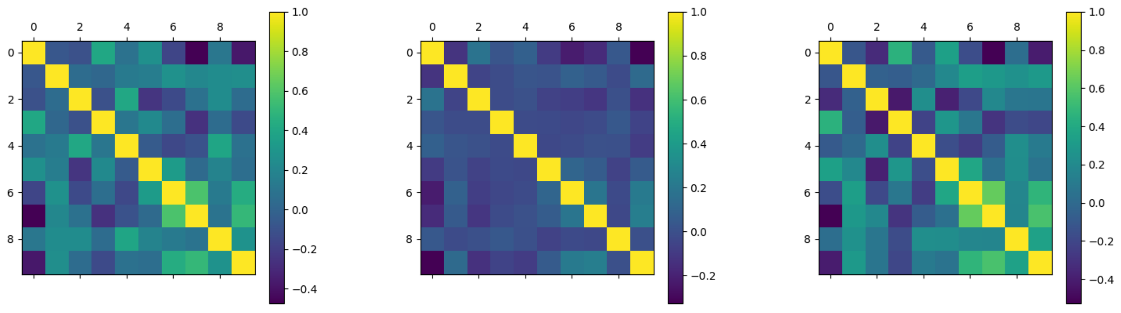


Figure 1.4: LDA direction

Analyzing the PCA directions 1.2 it can be clearly seen that while the target class can be well-characterized by one cluster, that cannot be stated for the non-target class. Even though, by looking at the main LDA direction 1.4 it might be possible to separate the classes with a linear classifier. Nonetheless, by having a look at the scatter plot 1.3 it can be deduced that linear classifiers might not be able to sufficiently model the non-target class and I might achieve better results with non-linear models.

After having a brief look at the features of the dataset I further analyze the possible benefits of applying dimensionality reduction. The Pearson correlation coefficient can tell how much the class features are correlated with each other.

Studying these matrices 1.5 it can be stated that the features between the target class are weakly correlated to one another. However, there is some correlation among the features of the non-target



(a) Dataset (b) Target class (c) Non-target class

Figure 1.5: Pearson correlation coefficient for the dataset features

class. While PCA might help with reducing the number of features and thus some correlation, it has to be remembered that the feature space is small, thus removing too many features might withdraw useful information for the classification. Indeed, the explained variance of the PCA direction can confirm that by keeping 6 features, I lose around 18 % of the dataset's explained variance.

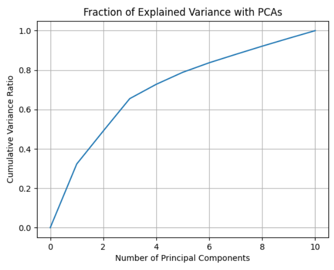


Figure 1.6: PCA - explained variance

In general, PCA performs better in cases where the number of samples per class is less, which is the case for our dataset. Whereas, LDA works better with large datasets having multiple classes. Thus, during the training and model selection phase, PCA is going to be utilized as the dimensionality reduction technique.

CHAPTER 2

Model training and model selection

During the training of the various models A K-fold protocol with K=5 is going to be applied, to increase the amount of data the models are trained on, and to get a more precise performance for each model. The primary metric during model selection is going to be the minimum DCF of each model. The primary working point of the algorithm is $\pi_n = 0.5, C_{fn} = 1, C_{fp} = 10$. To test the model candidates on a wider range of applications, I will also check the performance of each model on a secondary working point with the parameters $\pi_n = 0.5, C_{fn} = 1, C_{fp} = 1$. Nonetheless, the selection of the best models will be based on the primary working point. During the model selection and training, different preprocessing techniques will be utilized (Z-normalization and PCA dimensionality reduction) and their effects on the model performance will be studied.

I evaluate the actual DCFs and perform the score calibration once the most promising model(s) has/have been selected.

2.1 Generative classifiers

2.1.1 Preliminary analysis

I started the training and model selection on Gaussian classifiers. While the target class shows little correlation among its features, this can not be stated for the non-target class. Hence, a Multivariate Gaussian might be able to model these classes better. However, we have to keep in mind, that the low amount of data can make it harder to make accurate predictions.

The tied approach assumes that the target and the non-target classes have the same covariance matrices. As we saw in our initial analysis 1.5, this is not the case. For this reason, I expect the results for this classifier to be worse.

The Naive Bayes approach assumes diagonal covariance matrices. Since the non-target class shows a correlation among its features I anticipate the diagonal covariance matrix approach to be less effective.

The Naive Bayes approach with tied covariance matrices combines the two models. Since there are correlations between the features of the classes this approach is also expected to be less effective.

I will test all approaches together with the effects of applying PCA and Z-normalization on both working points.

2.1.2 Results

In general, a conclusion can be made that all of the models perform worse in our primary working point than in the secondary. This is due to the high cost of false positive errors in the primary working

Working point	(0.5, 1, 10)		(0.5, 1, 1)	
	Raw	Z-normalized	Raw	Z-normalized
No PCA	0.331	0.331	0.104	0.104
PCA 10	0.331	0.331	0.104	0.104
PCA 9	0.338	0.358	0.102	0.107
PCA 8	**0.329**	0.337	0.105	0.113
PCA 7	0.336	0.330	0.106	0.115
PCA 6	0.337	0.353	0.104	0.116

Table 2.1: MVG minDCF scores with different configurations

Working point	(0.5, 1, 10)		(0.5, 1, 1)	
	Raw	Z-normalized	Raw	Z-normalized
No PCA	0.486	0.486	0.173	0.173
PCA 10	0.486	0.486	0.173	0.173
PCA 9	**0.472**	0.486	0.171	0.178
PCA 8	0.476	0.476	0.171	0.173
PCA 7	0.475	0.476	0.170	0.175
PCA 6	0.486	0.500	0.170	0.181

Table 2.2: Tied Gaussian minDCF scores with different configurations

Working point	(0.5, 1, 10)		(0.5, 1, 1)	
	Raw	Z-normalized	Raw	Z-normalized
No PCA	0.472	0.472	0.135	0.135
PCA 10	0.390	0.375	0.107	0.115
PCA 9	**0.381**	0.384	0.105	0.112
PCA 8	0.391	0.382	0.106	0.112
PCA 7	0.388	0.382	0.108	0.110
PCA 6	0.391	0.389	0.105	0.113

Table 2.3: Naive Bayes minDCF scores with different configurations

Working point	(0.5, 1, 10)		(0.5, 1, 1)	
	Raw	Z-normalized	Raw	Z-normalized
No PCA	0.551	0.551	0.135	0.186
PCA 10	0.557	0.502	0.107	0.183
PCA 9	0.569	0.500	0.105	0.188
PCA 8	0.568	**0.499**	0.106	0.187
PCA 7	0.571	0.505	0.108	0.187
PCA 6	0.563	0.543	0.105	0.188

Table 2.4: Tied Naive Bayes minDCF scores with different configurations

point.

As expected, the Multivariate model performed the best out of the different models 2.1. It can be deduced that Z-normalization does not improve the model's performance. The identical results of the Z-normalized and the raw data without PCA are not by accident. Z-normalization transforms the dataset by subtracting the mean from each point and dividing these results by the standard deviation. The MVG model computes and uses the same metrics to predict the samples, hence the match. Similarly, there is no change between the raw and the PCA preprocessed results with 10 directions.

Employing PCA until the 8th direction proves to be beneficial because it can remove some unwanted correlation while keeping the most principal components. However, after that, too much information is removed and the performance starts to deteriorate. For that reason, further PCA dimensionality reductions are not going to be experimented with.

The Tied model, according to my expectations, performed worse than the MVG model. 2.2 This is due to the different covariance matrices of the two classes. Similarly to the MVG, Z-normalization is not helping, but PCA improves the results until a certain point, like in the MVG case.

The Naive Bayes approach also performs worse than the MVG model, 2.3 because as we had seen in the heatmap 1.5 there is a correlation among the features. None of the Naive Bayes configurations can reach such a result as the worst MVG configuration. We could move on to the next model, however, there are some exciting details in the results that are worth our attention. In this case, PCA with 10 directions gives better results than without using this preprocessing technique. This is because PCA projects the features into a subspace that retains the highest variance, which corresponds to diagonalizing the covariance matrices. Thus, the combination of PCA-10 and the Naive Bayes approach (that assumes diagonal covariance matrices) yields better results. This explanation is also applicable to further dimension reductions. With the Naive Bayes case, Z-normalization proves to be helpful.

Combining the Naive Bayes and the Tied assumptions results in a worse model than using them separately, 2.4 as I had anticipated in the preliminary analysis. Z-normalization can help obtain better results and PCA also proves to be helpful even with 10 directions (for the same reason as in the Naive Bayes model).

In the following, I will continue using Z-normalization and PCA, but only until 8 dimensions, since none of the models performed the best with less than 8 directions kept.

2.2 Discriminative probabilistic classifiers

The next family of classifiers that I investigate are discriminative probabilistic classifiers. In particular, we are going to use both the Linear and Quadratic Logistic Regression classifiers. For both models, the effects of Z-normalization and dimensionality reduction with PCA are going to be evaluated.

Because the tied Gaussian model also has a linear separation surface (unlike the MVG model that has a quadrative separation surface) and it performed worse than the MVG model, I initially foresee the Quadratic Logistic Regression model to outperform its Linear relative.

2.2.1 Linear Logistic Regression

As a discriminative model, the Linear Logistic Regression directly estimates the class posterior probabilities. It has a linear separation surface in the form of $s = w^T * x + b$, where w^T and b are the parameters that the model should estimate. With linearly separable classes, the logistic regression solution is not defined, because $||w||$ can be arbitrarily high. To solve this problem, a regularizer term λ was introduced that weighs in the norm of w. This parameter is a hyperparameter whose value should be chosen by using cross-validation, as the optimizer would set it to 0 to minimize the objective function. Moreover, since the classes in the dataset are unbalanced we are going to use the weighted form of the objective function:

$$J(w,b) = \frac{\lambda}{2}||w||^2 + \frac{\pi_T}{n_T}\sum_{i=1|_{z_i=1}}^{n} log(1+e^{-z_i(w^T x_i+b)}) + \frac{(1-\pi_T)}{n_F}\sum_{i=1|_{z_i=-1}}^{n} (1+e^{-z_i(w^T x_i+b)})$$

where:

- z_i is computed from the class label of the ith sample in the form of $z_i = 2c_i - 1$
- π_T represents the prior probability of the true class (label 1)
- n_T and n_F are the number of samples from the True (label 1) and the False (label 0) class, respectively
- λ is the regularizer term.

I performed a grid search of the parameters with the Linear model in both working points with and without Z-normalization with several different PCA dimensionality reductions. At each combination defined, I tried out different λ values in a logarithmic manner between 10^{-6} and 10^{4} to find the best model parameters. When selecting the best model parameters I will consider the primary working point, as I did with the Generative models. During this grid search, I did not consider different π_T prior weighting. Afterwards, a couple of configurations will be selected, on which various π_T weighting will be tried out. However, I do not expect marginal gains in performance. In the following, the results obtained by executing this procedure will be presented.

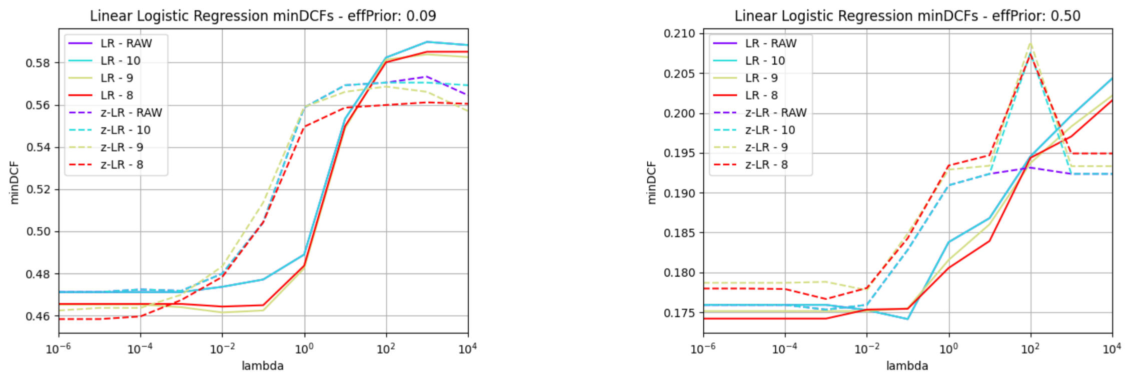


Figure 2.1: MinDCF of Linear Logistic Regression classifier as function of λ in different working points

Results

It can be observed that when the value of λ is too big (10^0 for the Z-normalized LR and 10^1 for the raw LR) the model fails to classify the samples well. This happens because, with a high penalty on the norm of w, the classifier underfits the dataset and achieves an overly general separation rule. However, with λ values under these two points, the classifier's performance noticeably improves until a certain point, after which further decrease of the hyperparameter has little to no effect on the performance. Due to the decrease of the λ hyperparameter the model is less penalized for creating a more specialized separation surface and thus can achieve better results. Nonetheless, we should not go overboard with the decrease of λ because that could lead to overfitting on the training data and poor generalization for unseen samples.

Working point	(0.5, 1, 10)		(0.5, 1, 1)	
	Raw	Z-normalized	Raw	Z-normalized
No PCA				
$\lambda = 10^{-6}$	0.471	0.471	0.176	0.176
$\lambda = 10^{-5}$	0.471	0.471	0.176	0.176
$\lambda = 10^{-4}$	0.471	0.472	0.176	0.176
$\lambda = 10^{-3}$	0.471	0.472	0.176	0.175
$\lambda = 10^{-2}$	0.474	0.480	0.175	0.176
$\lambda = 10^{-1}$	0.477	0.504	0.174	0.183
$\lambda = 10^{0}$	0.489	0.559	0.184	0.191
$\lambda = 10^{1}$	0.553	0.569	0.187	0.192
PCA 9				
$\lambda = 10^{-6}$	0.465	0.462	0.175	0.179
$\lambda = 10^{-5}$	0.465	0.464	0.175	0.179
$\lambda = 10^{-4}$	0.465	0.464	0.175	0.179
$\lambda = 10^{-3}$	0.464	0.470	0.175	0.179
$\lambda = 10^{-2}$	**0.461**	0.483	0.175	0.178
$\lambda = 10^{-1}$	**0.462**	0.514	0.175	0.185
$\lambda = 10^{0}$	0.482	0.559	0.182	0.193
$\lambda = 10^{1}$	0.549	0.566	0.186	0.193
PCA 8				
$\lambda = 10^{-6}$	0.466	0.458	0.174	0.178
$\lambda = 10^{-5}$	0.466	**0.458**	0.174	0.178
$\lambda = 10^{-4}$	0.466	0.460	0.174	0.178
$\lambda = 10^{-3}$	0.466	0.467	0.174	0.177
$\lambda = 10^{-2}$	0.464	0.478	0.175	0.178
$\lambda = 10^{-1}$	0.465	0.504	0.175	0.184
$\lambda = 10^{0}$	0.484	0.550	0.181	0.193
$\lambda = 10^{1}$	0.550	0.559	0.184	0.195

Table 2.5: Linear Logistic Regression minDCF scores with different configurations

The effects of Z-normalization are lopsided; with a small λ the model in the primary working point can achieve slightly better results. At the same time, the performance starts to degrade faster than without Z-normalization.

Interestingly, we can observe that in the secondary working point, Z-normalization does not improve the model performance. The model that works on the raw data can achieve better results.

Finally, by analyzing the effects of dimensionality reduction we can deduce that it has a beneficial impact on the performance. with 9 and 8 dimensions kept. In the following table 2.5, the minDCF values are going to be summarized. I omit to mention the PCA with 10 dimensions kept, as it has very similar results to the one without employing PCA, at all. Furthermore, the λ values are only going to be included until 10^1.

The most promising models after my initial grid search are the ones with PCA 9, $\lambda = 10^{-2}$, on raw data and PCA 8, $\lambda = 10^{-5}$ on Z-normalized data. Even though, the latter performs slightly better we have to keep in mind that due to the small value of lambda, it might generalize worse than the prior configuration. For this reason, the configuration with PCA 9, $\lambda = 10^{-1}$, on raw data could be considered, too. However, for now, I will keep both models and in the later steps of the project, I will validate whether I made good assumptions or not.

The next step of the procedure that I outlined is to try out these two models with different prior

	(0.5, 1, 10)	(0.5, 1, 1)
$\pi_T = 0.05$	0.469	0.179
$\pi_T = 0.0909$ (effPrior)	0.458	0.178
$\pi_T = 0.5$	0.480	0.178
$\pi_T = 0.9$	0.507	0.181

Table 2.6: minDCFs of Linear Logistic Regression models with PCA 8, Z-normalization, $\lambda = 10^{-5}$ with different prior weighting

	(0.5, 1, 10)	(0.5, 1, 1)
$\pi_T = 0.05$	0.463	0.170
$\pi_T = 0.0909$ (effPrior)	0.461	0.175
$\pi_T = 0.5$	0.481	0.175
$\pi_T = 0.9$	0.504	0.183

Table 2.7: minDCFs of Linear Logistic Regression models with PCA 9, raw data, $\lambda = 10^{-2}$ with different prior weighting

weighting. I consider three values: $\pi_T = 0.05$ (which is very close to the prior defined by our working point, 0.09), $\pi_T = 0.5$, and $\pi_T = 0.9$.

We can mind that different prior weights 2.6, 2.7 do not improve the performance, thus reinforcing our previous assumption. We can also see that the model with bigger lambda 2.7 is more resilient to changes in the dataset composition, implying it could be a better choice.

Conclusions

All in all, the linear logistic regression model performed similarly to the tied Gaussian, which was expected knowing the underlying theory. Z-normalization and different prior weighting proved to be indifferent concerning the performance. However, PCA turned out to be useful. In any case, based on the results of the MVG model, I expect the Quadratic Logistic Regression model to perform better than the Linear one.

2.2.2 Quadratic Logistic Regression

It is possible to achieve quadratic separation surfaces with a Logistic Regression model without modifying the regression solution. To achieve this we need to perform a transformation on the dataset, to obtain an expanded feature space. A quadratic score can be written in the form of $s(x, A, b, c) = x^T Ax + b^T x + c$. We can modify the equation to obtain an equivalent one that uses inner products and $vec(x)$ operation that stacks the columns of x and has the form of: $<A, xx^T> = vec(xx^T)^T vec(A)$. Thus we need to define $\Phi(x) = [vec(xx^T), x]$ and $w = [vec(A), b]$, then our separation rule becomes: $s(x, w, c) = w^T \Phi(x) + c$.

For the Quadratic Logistic Regression model, the same performance evaluation procedure was performed as with the Linear classifier.

Results

In general, similar tendencies can be observed by analyzing the results of the quadratic model. My initial assumption that this model would outperform the linear model turned out to be true. The value of λ has the same effect on the performance as in the linear model, even the turning points are similar.

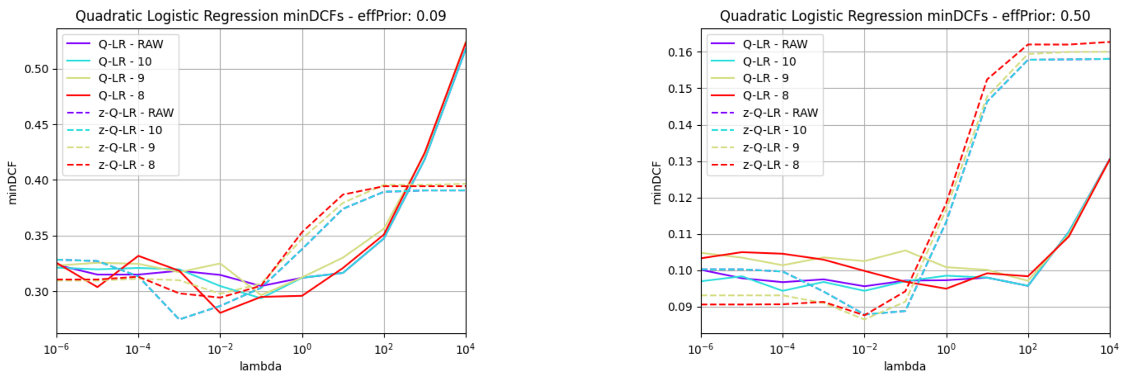


Figure 2.2: MinDCF of Quadratic Logistic Regression classifier as function of λ in different working points

The effects of Z-normalization cannot be decided for the primary working point. Only in some configurations, the usage of this preprocessing technique seems to be practical. In general, it can be stated that the model's performance that works on the Z-normalized data starts to degrade faster as the value of λ increases, but it stops degrading at $\lambda = 10^1$ and as λ is further increased it outperforms the configurations working on raw data. In the secondary working point Z-normalized configuration outperforms the non-normalized ones. However, they lose their performance faster as λ increases and their performance stays stably worse. This could be explained with that the quadratic model's non-linear decision surface is sensitive to a linear transformation due to the dot products that perform the feature expansion. Consequently, the QLR works well with this linear preprocessing technique in the secondary working point.

Similarly, there is no clear conclusion about the advantages of PCA that can be drawn by looking at the diagram of the primary working point 2.2. I will ignore PCA with 10 directions for the same reason as before. We can see that the model can perform well without dimensionality reduction as well as with it. Nevertheless, in the secondary working point the Z-normalized configurations perform visibly better when PCA is also used.

After analyzing the main characteristics of the classifier we turn our attention to the table with the reported minDCF scores 2.8 to pick the most promising configurations for the classifier. We can infer that the best configurations in the primary working point are the one that works on raw data, has PCA with 8 directions kept, and with $\lambda = 10^{-2}$, and the one with no dimensionality reduction, with Z-normalization and with $\lambda = 10^{-5}$. My next action is to check whether changing the prior probability can further improve the performance.

Conclusions

Following the linear model results, the quadratic classifier's performance was not improved by changing the prior weighting, either. In this case, the model without PCA dimensionality reduction is more resistant to changes in the prior weighting than the other candidate. For this reason and for its better overall performance I choose this configuration as the general candidate from this family of classifiers.

In summary, it can be stated that the quadratic model outperformed the linear one which is following our expectations. It also performed better than the best generative model.

Working point	(0.5, 1, 10)		(0.5, 1, 1)	
	Raw	Z-normalized	Raw	Z-normalized
No PCA				
$\lambda = 10^{-6}$	0.324	0.328	0.100	0.100
$\lambda = 10^{-5}$	0.315	0.327	0.098	0.100
$\lambda = 10^{-4}$	0.315	0.313	0.097	0.100
$\lambda = 10^{-3}$	0.318	**0.275**	0.098	0.094
$\lambda = 10^{-2}$	0.315	0.287	0.096	0.088
$\lambda = 10^{-1}$	0.305	0.303	0.097	0.089
$\lambda = 10^{0}$	0.312	0.338	0.097	0.113
$\lambda = 10^{1}$	0.316	0.374	0.098	0.146
PCA 9				
$\lambda = 10^{-6}$	0.323	0.310	0.105	0.093
$\lambda = 10^{-5}$	0.326	0.310	0.103	0.093
$\lambda = 10^{-4}$	0.325	0.311	0.101	0.093
$\lambda = 10^{-3}$	0.317	0.310	0.104	0.091
$\lambda = 10^{-2}$	0.325	0.298	0.103	0.086
$\lambda = 10^{-1}$	0.296	0.307	0.105	0.091
$\lambda = 10^{0}$	0.312	0.347	0.101	0.117
$\lambda = 10^{1}$	0.330	0.379	0.100	0.159
PCA 8				
$\lambda = 10^{-6}$	0.326	0.311	0.103	0.091
$\lambda = 10^{-5}$	0.304	0.311	0.105	0.091
$\lambda = 10^{-4}$	0.332	0.313	0.105	0.091
$\lambda = 10^{-3}$	0.318	0.298	0.103	0.091
$\lambda = 10^{-2}$	**0.281**	0.294	0.100	0.088
$\lambda = 10^{-1}$	0.295	0.305	0.097	0.094
$\lambda = 10^{0}$	0.296	0.353	0.095	0.118
$\lambda = 10^{1}$	0.321	0.387	0.099	0.152

Table 2.8: Quadratic Logistic Regression minDCF scores with different configurations

	(0.5, 1, 10)	(0.5, 1, 1)
$\pi_T = 0.05$	0.303	0.108
$\pi_T = 0.0909$ (effPrior)	0.281	0.100
$\pi_T = 0.5$	0.311	0.097
$\pi_T = 0.9$	0.350	0.098

Table 2.9: minDCFs of Quadratic Logistic Regression models with PCA 8, raw data, $\lambda = 10^{-2}$ with different prior weighting

	(0.5, 1, 10)	(0.5, 1, 1)
$\pi_T = 0.05$	0.281	0.092
$\pi_T = 0.0909$ (effPrior)	0.275	0.094
$\pi_T = 0.5$	0.297	0.088
$\pi_T = 0.9$	0.310	0.90

Table 2.10: minDCFs of Quadratic Logistic Regression models without PCA, Z-normalization, $\lambda = 10^{-3}$ with different prior weighting

2.3 Discriminative non-probabilistic classifiers

The third group of classifiers that I am going to consider are discriminative non-probabilistic classifiers. More specifically, Support Vector Machines. The SVM is a discriminative non-probabilistic classifier, which means that it directly estimates the class posterior likelihoods. It is non-probabilistic because the scores that are output do not carry any probabilistic interpretation. The SVM classifier tries to find the best hyperplane that separates the classes from each other. It does so, by finding the hyperplane that has the highest minimal margin (maximum margin hyperplane). For the same reasons, I am going to use the prior weighted version of the objective function of the classifier, which is as follows for the primal problem:

$$L(w,b) = argmin_{\mathbf{w},b}\frac{1}{2}||w||^2 + \sum_{i=1}^{n} C_i[1 - z_i(w^T x_i + b)]_+$$

Or, for the dual problem:

$$L(\alpha) = argmax_{\alpha}\alpha^T\mathbf{1} - \frac{1}{2}\alpha^T H\alpha$$

,

such that:

$$\sum_{i=1}^{n}\alpha_i z_i = 0, \quad 0 \leq \alpha_i \leq C_i, i = 1, ..., n$$

where:

- z_i is computed from the class label of the ith sample in the form of $z_i = 2c_i - 1$
- C_i implements the class balancing. We can set $C_i = C_T$ for the true class and $C_i = C_F$ for the false class. In this case, we can select $C_T = \frac{\pi_T}{\pi_T^{emp}}$ and $C_F = \frac{\pi_F}{\pi_F^{emp}}$. π_T^{emp} and π_F^{emp} are the empirical priors of the two classes over the training set.
- H is the matrix, whose ith and jth element is $H_{ij} = z_i z_j x_i^T x_j$

Similarly to logistic regression, it is possible to achieve a non-linear separation surface with the SVM classifier. To achieve this, we utilise kernel functions that can transform the data into an expanded, large-dimensional Hilbert space, without explicitly computing the mapping. The computation gives a linear separation surface in the expanded space which translates into a non-linear separation surface in the original space. If we find a kernel function that maps the data in such a way then we can simply modify H in our dual objective function to be: $H_{ij} = z_i z_j k(x_i, x_j)$, where k is the kernel function: $k = \Phi(x_i)^T\Phi(x_j)$, and Φ is the function that performs the mapping.

I am going to analyze the performance of both the linear and the non-linear versions of the classifier. From the non-linear version, I am going to employ the polynomial as well as the radial basis function kernels to achieve non-linearity. Similarly to the logistic regression, I expect the linear model to perform worse than the non-linear models.

2.3.1 Linear SVM

The training methodology is going to be similar to the one I utilised for the logistic regression classifier. I am going to try out the classifier with and without Z-normalization with different PCA dimensionality reductions. For each combination, I am going to try out different C values in the range of $10^{-6} - 10^{4}$.

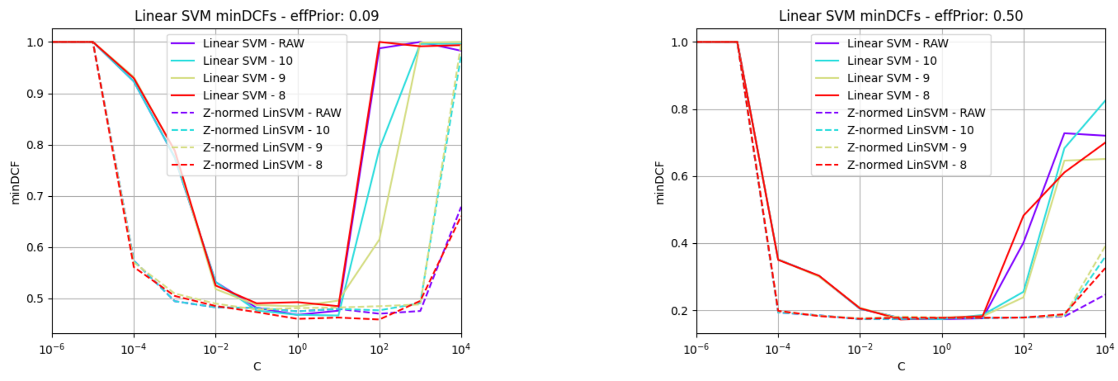


Figure 2.3: MinDCF of Linear SVM classifier as function of C in different working points

Results

We can observe by looking at the diagrams 2.3 that the value of C is of vital importance to obtain acceptable values. Practically, values that are less than 10^{-4} or greater than 10^{2} do not perform good. Too big C overly generalizes the separation surface, whereas too small C overcomplicates the separation surface which is not going to be able to separate unseen data.

Z-normalization appears to be advantageous in the sense that it extends the range of C-s where the classifier performs well. Moreover, it also yields narrowly better DCF performance than the models that did not employ this preprocessing technique.

I cannot deduce a clear judgment for the usefulness of dimensionality reduction. Some directions appear to offer better results than others, but only marginally.

Similarly, as with the other classifiers, our primary working point's classifiers achieve worse results than in the secondary. Moreover, the performance is acceptable only in a narrower range of C-s. Further analysis is going to be performed on the tabular version of the results. In this table, 2.11 values of C-s are going to be restricted to the range between 10^{-4} and 10^{2}. Again due to the similarity in results with the no PCA configuration, I am going to omit to analyze PCA with 10 directions kept.

Looking at the table 2.11 it can be noticed that the best model is the configurations with 8 directions kept, with Z-normalization and $C = 10^{2}$. We can also observe that other C ($C = 10^{1}$, $C = 10^{0}$) values in this configuration perform almost identically both in the primary and the secondary working points. The second best candidate is the configuration without dimensionality reduction, and without z-normalization, with $C = 10^{0}$. This configuration, however, performs slightly worse than the first one.

Conclusions

The value of C is paramount for the SVM classifier as it makes the results greatly fluctuate. Z-normalization is worthwhile and preprocessing can improve the performance as well. Regardless, the overall scores fall behind the scores achieved with non-linear classifiers. For this reason, I expect the non-linear SVM classifiers to outperform the linear ones. For the very same reason, I am not going to study the effects of class rebalancing in the initial grid search.

2.3.2 Polynomial SVM

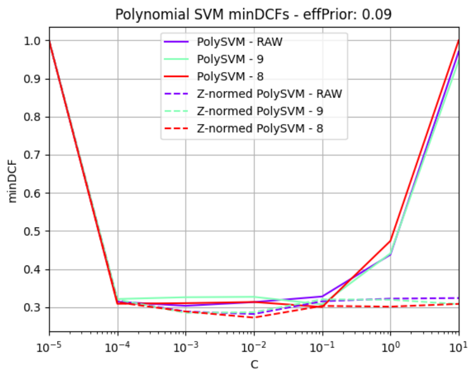
The polynomial SVM uses the following kernel to achieve a non-linear separation rule: $k(x_1, x_2) = (x_1^T x_2 + c)^d$. I am going to use the second-order kernel and set c to one, such that the kernel becomes:

Working point	(0.5, 1, 10)		(0.5, 1, 1)	
	Raw	Z-normalized	Raw	Z-normalized
No PCA				
$C = 10^{-4}$	0.923	0.574	0.351	0.193
$C = 10^{-3}$	0.775	0.495	0.302	0.184
$C = 10^{-2}$	0.532	0.483	0.204	0.173
$C = 10^{-1}$	0.482	0.482	0.176	0.174
$C = 10^{0}$	**0.468**	0.475	0.173	0.177
$C = 10^{1}$	0.476	0.480	0.176	0.178
$C = 10^{2}$	0.988	0.470	0.401	0.181
PCA 9				
$C = 10^{-4}$	0.931	0.570	0.351	0.197
$C = 10^{-3}$	0.787	0.510	0.300	0.183
$C = 10^{-2}$	0.518	0.490	0.207	0.176
$C = 10^{-1}$	0.487	0.478	0.174	0.181
$C = 10^{0}$	0.485	0.481	0.178	0.178
$C = 10^{1}$	0.496	0.482	0.182	0.178
$C = 10^{2}$	0.615	0.485	0.238	0.178
PCA 8				
$C = 10^{-4}$	0.929	0.561	0.350	0.198
$C = 10^{-3}$	0.788	0.505	0.303	0.182
$C = 10^{-2}$	0.525	0.485	0.206	0.174
$C = 10^{-1}$	0.490	0.473	0.172	0.177
$C = 10^{0}$	0.492	0.460	0.177	0.177
$C = 10^{1}$	0.484	0.463	0.182	0.177
$C = 10^{2}$	1.000	**0.459**	0.48	0.178

Table 2.11: Linear SVM minDCF scores with different configurations

$k(x_1, x_2) = (x_1^T x_2 + 1)^2$. Except for using this kernel, the training methodology differs from one parameter: I decided to skip the configuration with PCA with 10 dimensions kept. Moreover, based on the experiences learnt from the linear model I limited the value of C trials to the range of 10^{-5} and 10^1.

Results



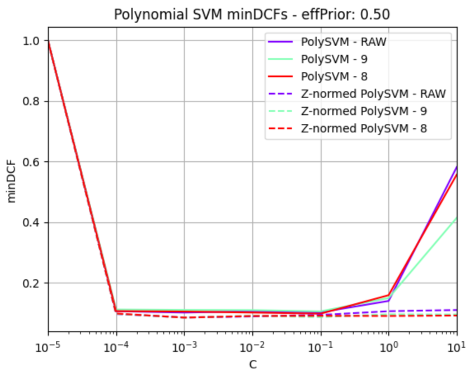


Figure 2.4: MinDCF of Polynomial SVM classifier as function of C in different working points

As expected the polynomial SVM performs better than the linear model which can be seen in the diagrams 2.4. The value of C gives good results in a similar range as was the case with the linear model, between 10^{-4} and 10^1. Z-normalization proves to be advantageous in this case as well, as it gives better performance in a wider variety of C values. PCA, similarly to before, only improves the results marginally. The trends are similar in both working points, except in the secondary, the scores are as usual smaller.

Looking at the numerical values 2.12 we can observe that the best models are clearly with 8 PCA directions. As mentioned before, Z-normalization provides better performance. The ideal C values are also very close to each other in this case, being 10^{-2} for the best model with Z-normalization and 10^{-1} for the best model without Z-normalization. For now, the best model candidate is with Z-normalization, with 8 directions kept and with $C = 10^{-2}$. Now, I am going to analyze the effects of class rebalancing. As it can be seen in the table reporting the results 2.13 varying the prior weighting does not improve the performance. On the contrary, we can observe a greater decrease in performance than in the case of the best logistic regression models. This might suggest that this classifier could react worse to an unknown dataset than the logistic regression model.

2.3.3 RBF SVM

The Radial Basis Function (RBF) SVM uses another kernel to achieve non-linearity, specifically: $k(x_1, x_2) = e^{-\gamma||x_1 - x_2||^2}$, where γ is a hyperparameter to tune. Practically, it defines the width of the kernel, a small γ gives a wider kernel, and a bigger γ gives a narrower kernel. During my assignment, I tried out several different γ values.

Results

As we can see from the figures 2.5 the general tendencies are similar independently from the value of the principal directions kept. We can notice that the value of γ greatly influences the prospects of the model in terms of the best possible performance which is polished out by the tuning of the C

Working point	(0.5, 1, 10)		(0.5, 1, 1)	
	Raw	Z-normalized	Raw	Z-normalized
No PCA				
$C = 10^{-5}$	1.000	1.000	1.000	1.000
$C = 10^{-4}$	0.314	0.318	0.106	0.097
$C = 10^{-3}$	0.304	0.289	0.102	0.086
$C = 10^{-2}$	0.313	0.282	0.106	0.089
$C = 10^{-1}$	0.328	0.316	0.103	0.094
$C = 10^{0}$	0.438	0.322	0.140	0.106
$C = 10^{1}$	0.971	0.324	0.582	0.110
PCA 9				
$C = 10^{-5}$	1.000	1.000	1.000	1.000
$C = 10^{-4}$	0.321	0.320	0.112	0.098
$C = 10^{-3}$	0.326	0.285	0.111	0.085
$C = 10^{-2}$	0.327	0.287	0.110	0.092
$C = 10^{-1}$	0.307	0.321	0.106	0.087
$C = 10^{0}$	0.441	0.320	0.150	0.096
$C = 10^{1}$	0.947	0.308	0.415	0.095
PCA 8				
$C = 10^{-5}$	1.000	1.000	1.000	1.000
$C = 10^{-4}$	0.309	0.313	0.107	0.099
$C = 10^{-3}$	0.311	0.289	0.105	0.085
$C = 10^{-2}$	0.314	**0.273**	0.102	0.090
$C = 10^{-1}$	**0.300**	0.303	0.098	0.091
$C = 10^{0}$	0.474	0.301	0.159	0.091
$C = 10^{1}$	1.000	0.309	0.557	0.092

Table 2.12: Polynomial SVM minDCF scores with different configurations

	(0.5, 1, 10)	(0.5, 1, 1)
$\pi_T = 0.05$	0.306	0.166
$\pi_T = 0.0909$ (effPrior)	0.273	0.090
$\pi_T = 0.5$	0.300	0.123
$\pi_T = 0.9$	0.375	0.133

Table 2.13: minDCFs of Polynomial SVM models with PCA 8, Z-normalization, $C = 10^{-2}$ with different prior weighting

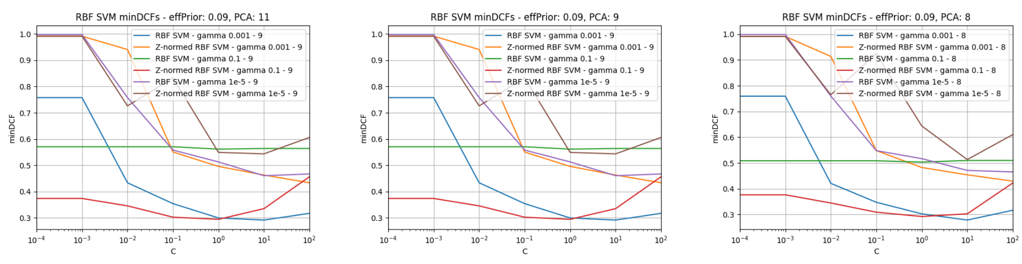


Figure 2.5: MinDCF of RBF SVM classifier as function of C with different γ values in the primary working point

No PCA				
Working point	(0.5, 1, 10)		(0.5, 1, 1)	
	Raw	Z-normalized	Raw	Z-normalized
$\gamma = 0.001$				
$C = 10^{-2}$	0.439	0.943	0.164	0.413
$C = 10^{-1}$	0.352	0.543	0.116	0.190
$C = 10^{0}$	0.301	0.488	0.092	0.175
$C = 10^{1}$	0.293	0.461	0.081	0.168
$C = 10^{2}$	0.332	0.436	0.094	0.146
$\gamma = 0.1$				
$C = 10^{-2}$	0.601	0.344	0.151	0.120
$C = 10^{-1}$	0.601	0.295	0.151	0.090
$C = 10^{0}$	0.601	0.303	0.151	0.089
$C = 10^{1}$	0.601	0.324	0.151	0.099
$C = 10^{2}$	0.601	0.414	0.151	0.144

Table 2.14: RBF SVM minDCF scores without PCA, with different λ and C configurations

parameter. We can see that the best models are the ones with $\gamma = 0.1$ with Z-normalization, and $\gamma = 0.001$ without Z-normalization. The configurations with $\gamma = 1e - 5$ underperform the other models and for this reason, I will not consider them further. Moreover, I also omit the lowest values of C, those that certainly do not perform the best.

By looking at the tabular form of the results 2.14, 2.15, 2.16 we can determine the best configuration, which is with 8 PCA directions on raw data, with $\gamma = 0.001$ and $C = 10^1$. This is going to be my candidate from this subtype of the SVM classifier.

Finally, we investigate the effects of the class rebalancing of the SVM on the selected candidate model. On the table reporting the results 2.17 we can see that again, the class rebalancing does not prove to be effective.

2.3.4 Conclusions

As we expected the non-linear SVM classifiers outperformed the linear model. For both the polynomial and the RBF kernel version of the classifier PCA dimensionality reduction with 8 directions proved to be the best configuration. The results of the non-linear SVM classifier were pretty similar to the performance of the quadratic logistic regression classifier.

2.4 Gaussian Mixture Models

The last group of classifiers that I am going to explore are the Gaussian Mixture Models (GMM). This type of classifier was originally created to estimate densities, but it has proven to be valuable in classification tasks as well. The assumption that this classifier makes is that the data is distributed according to one or more Gaussian distributions. I expect that this classifier is going to perform very well on this dataset because of the dataset's characteristics. We have seen in the initial analysis 1.2 and 1.3, that the distribution of the false class can be well estimated with Gaussians. Moreover, the description of the project also mentions that the false fingerprints were generated with 6 different techniques. Hence, I expect the GMM with 4 or 8 components to have the best performance. As we have seen with generative models, the multivariate Gaussian model performed the best 2.1, thus, I expect this one to be the best out of the GMM variants.

PCA 9				
Working point	(0.5, 1, 10)		(0.5, 1, 1)	
	Raw	Z-normalized	Raw	Z-normalized
$\gamma = 0.001$				
$C = 10^{-2}$	0.434	0.941	0.161	0.416
$C = 10^{-1}$	0.354	0.550	0.118	0.193
$C = 10^{0}$	0.300	0.496	0.098	0.176
$C = 10^{1}$	0.292	0.463	0.096	0.170
$C = 10^{2}$	0.317	0.434	0.102	0.143
$\gamma = 0.1$				
$C = 10^{-2}$	0.571	0.346	0.139	0.120
$C = 10^{-1}$	0.571	0.303	0.139	0.094
$C = 10^{0}$	0.562	0.295	0.143	0.093
$C = 10^{1}$	0.564	0.335	0.143	0.098
$C = 10^{2}$	0.564	0.457	0.143	0.124

Table 2.15: RBF SVM minDCF scores with PCA 9, with different λ and C configurations

PCA 8				
Working point	(0.5, 1, 10)		(0.5, 1, 1)	
	Raw	Z-normalized	Raw	Z-normalized
$\gamma = 0.001$				
$C = 10^{-2}$	0.421	0.914	0.162	0.392
$C = 10^{-1}$	0.347	0.548	0.116	0.193
$C = 10^{0}$	0.302	0.482	0.097	0.175
$C = 10^{1}$	**0.278**	0.454	0.094	0.169
$C = 10^{2}$	0.316	0.430	0.106	0.145
$\gamma = 0.1$				
$C = 10^{-2}$	0.509	0.345	0.128	0.118
$C = 10^{-1}$	0.509	0.309	0.128	0.094
$C = 10^{0}$	0.504	0.292	0.128	0.092
$C = 10^{1}$	0.510	0.303	0.132	0.101
$C = 10^{2}$	0.510	0.422	0.132	0.128

Table 2.16: RBF SVM minDCF scores with PCA 8, with different λ and C configurations

	(0.5, 1, 10)	(0.5, 1, 1)
$\pi_T = 0.05$	0.319	0.156
$\pi_T = 0.0909$ (effPrior)	0.278	0.094
$\pi_T = 0.5$	0.300	0.118
$\pi_T = 0.9$	0.358	0.118

Table 2.17: minDCFs of RBF SVM models with PCA 8, on raw data, with $\gamma = 0.001$, $C = 10^1$ with different prior weighting

2.4.1 Results

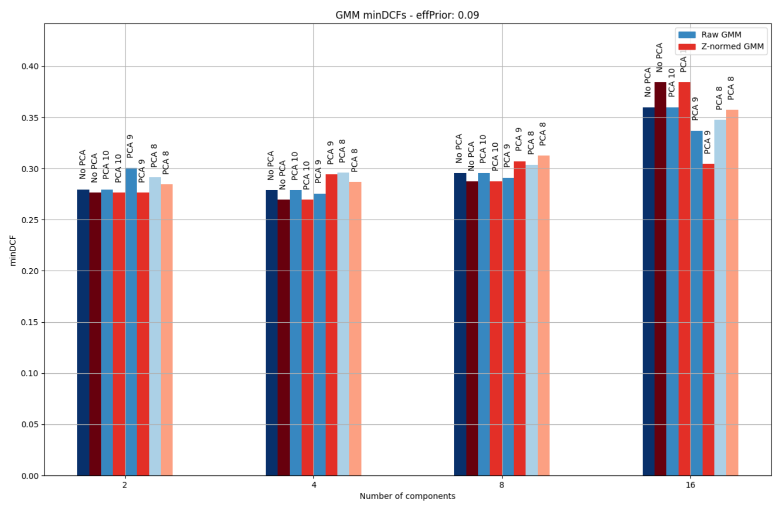


Figure 2.6: minDCF of GMM classifier with different configurations in the primary working point

As anticipated, the GMM classifiers provide the best results from all of the classifiers that I have analyzed so far. The most promising configurations are the ones with 4 and 8 components, which is also in line with my predictions.

For the GMM model 2.6 the ideal number of components is 4. We can note that as the number of components grows the classifier loses its performance. Z-normalization may help, especially with more dimensions removed. Dimensionality reduction in general, decreases the performance except for a few outliers.

With the tied model 2.7, the configuration with 2 components is clearly worse than the other 3. The model performs the best with 4 and 8 directions kept. This could be due to the tied model's identical covariance matrices of the various components that are easier to fit when having more components that estimate the data. For this classifier, Z-normalization does not prove to be effective and reducing the dimensionality drops the performance which can be due to the insufficient amount of data left.

For the diagonal variant 2.8, Z-normalization clearly makes the performance drop. Again, the best component numbers are 4 and 8, that could be due to the fact that the distribution (especially of the false class) is quite complex, hence more components are needed to fit the data better. In this case, dimensionality reduction is somewhat effective.

The tied diagonal variant 2.9 also performs the best with 4 and 8 components.

I reported the best configurations of each variant with and without Z-normalization, in both working points in the following table 2.18 to better compare them. I selected the best models from each variant based on their performance in the primary working point.

2.4.2 Conclusion

As it can be seen 2.18, the best model is the tied diagonal model with 8 components, without PCA and Z-normalization. This result is reasonable, based on the distribution of the scatter plots 1.3.

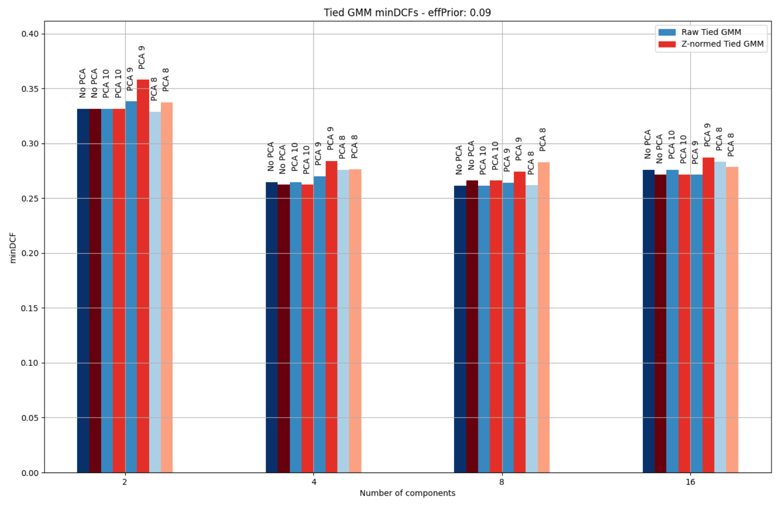


Figure 2.7: minDCF of Tied GMM classifier with different configurations in the primary working point

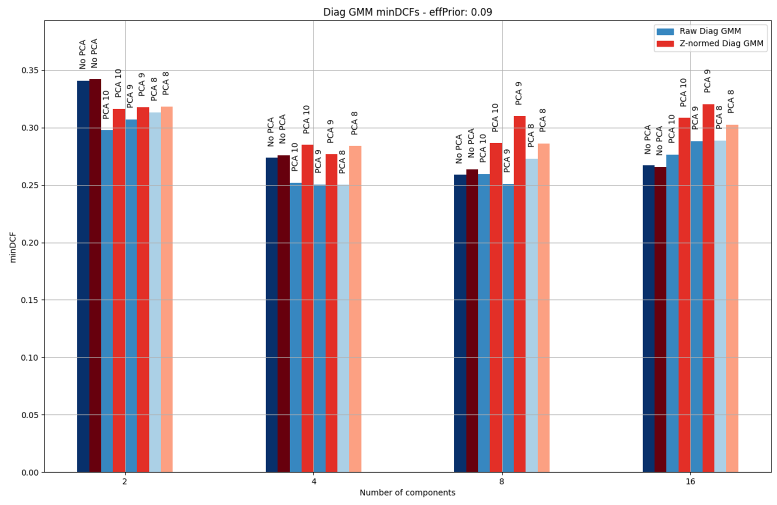


Figure 2.8: minDCF of Diag GMM classifier with different configurations in the primary working point

Working point	(0.5, 1, 10)		(0.5, 1, 1)	
	Raw	Z-normalized	Raw	Z-normalized
GMM (4 components, PCA 9)	0.276	0.294	0.088	0.081
Tied GMM (8 components, no PCA)	0.261	0.266	0.082	0.082
Diag GMM (4 components, PCA 8)	0.250	0.284	0.081	0.080
Tied Diag GMM (8 components, no PCA)	**0.244**	0.259	0.086	0.082

Table 2.18: Best GMM variants in different working points, with and without Z-normalization

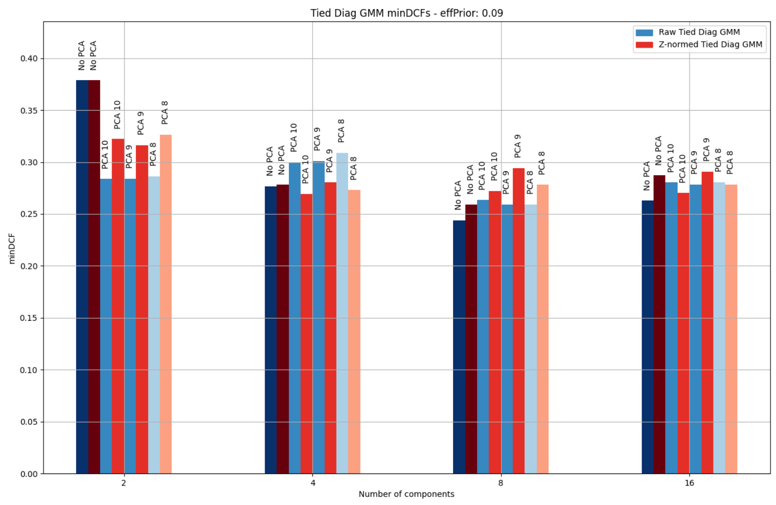


Figure 2.9: minDCF of Tied Diag GMM classifier with different configurations in the primary working point

	(0.5, 1, 10)	(0.5, 1, 1)
MVG (PCA 8, raw)	0.329	0.105
Q-LogReg (no PCA, Z-normed, $\lambda = 10^{-3}$)	0.275	0.094
Poly-SVM (PCA 8, Z-normed, $C = 10^{-2}$)	0.273	0.090
Tied Diag GMM (no PCA, raw, 8 components)	**0.244**	0.086

Table 2.19: Best classifiers from each type in both working points

2.5 Model selection

After the training phase of the project, the best models from each type of classifier are reported in table 2.19.

The GMM model is significantly better than the other models, hence I consider this as my main candidate. As secondary candidates, I choose the quadratic logistic regression and the poly-SVM. Since there is a bigger performance drop for the best Gaussian classifier I am going to omit this as a candidate. In the following chapter, I check whether the chosen models need to be calibrated or not. After that, I will evaluate the best models on the test dataset, and I will assess the choices that I made during the training phase.

CHAPTER 3

Calibration and fusion

3.1 Calibration of best models

During the training phase, I only considered the minDCF as the evaluation metric, which represents the best achievable score. However, most of the costs that we have to pay are due to the suboptimally chosen thresholds. In order to understand whether the threshold that the model uses is the theoretical one or not I am going to compare the minimum DCF metric with the actual DCF metric. The difference between these two metrics for a given working point gives the costs due to mis-calibration. This cost can be eliminated with the use of a calibrator model that is trained on the scores provided by the primary classifier. The calibration methodology that I employed takes the classifier's folds of scores and corresponding labels and shuffles them. Then these serve as the input for a simple Logistic Regression classifier that produces the calibrated scores. This way, ideally, I am going to be able to eliminate the miscalibration costs.

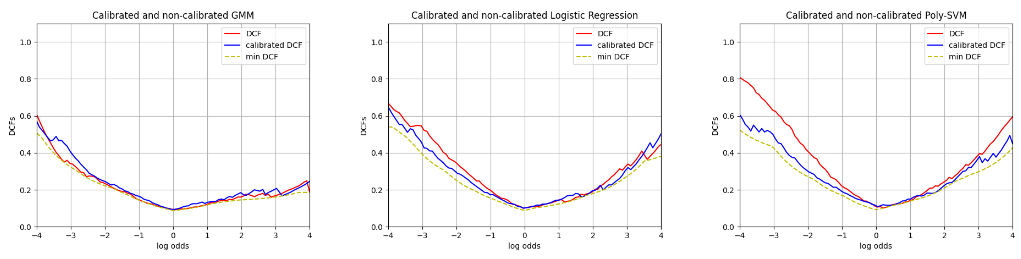


Figure 3.1: Uncalibrated and calibrated actual DCFs compared to minimum DCFs of the best 3 models during the training phase

It can be observed that the GMM model is well-calibrated on the majority of the working points. In fact, in most cases, the calibrated model performs slightly worse than the non-calibrated model. For the best Logistic Regression model (Quadratic) the calibrated scores are slightly better than the non-calibrated ones, except for some working points. In the case of the best SVM model, we can observe noteworthy miscalibration, especially on the extremes of the search area. In this case, calibration appears to be powerful, it is able to reduce the costs significantly.

Indeed, the numerical results 3.1, 3.2, 3.3 confirm that the GMM does not need calibration. I will deliver the uncalibrated version for simplicity and for better results. The quadratic logistic regression performs better with calibration on the main working point but not across all working points. With the Poly-SVM, calibration pays out, and I am able to achieve better results in every working point checked.

	Tied Diag GMM (no PCA, raw, 8 components)			
	$\pi = 0.05$	$\pi = 0.09$	$\pi = 0.5$	$\pi = 0.9$
minDCF	0.316	0.244	0.086	0.147
Uncalibrated actDCF	0.339	0.270	0.093	0.173
Calibrated actDCF	0.390	0.273	0.093	0.179

Table 3.1: GMM minimum uncalibrated and calibrated actual DCF scores

	Q-LogReg (no PCA, Z-normed, $\lambda = 10^{-3}$)			
	$\pi = 0.05$	$\pi = 0.09$	$\pi = 0.5$	$\pi = 0.9$
minDCF	0.383	0.275	0.094	0.193
Uncalibrated actDCF	0.521	0.392	0.100	0.220
Calibrated actDCF	0.414	0.352	0.114	0.239

Table 3.2: Q-LogReg minimum, uncalibrated and calibrated actual DCF scores

3.2 Fusion

The same idea of a calibrator model can be used to fuse the scores of different models. In this case, two logistic regression classifiers are used, one to fuse the two scores and one to calibrate the fused scores.

As experiments I created two fused classifiers, one is the fusion of the best GMM and best SVM classifiers, and the other is the fusion of the best GMM and best logistic regression classifier.

By looking at the figures 3.2, it can be seen that the fusion increases the performance compared to the standalone SVM and logistic regression models. Calibration, however, does not seem to marginally increase the performance. For precise performance comparison, the tabular form of the results should be used 3.4, 3.5.

It can be seen that compared to the best GMM model 3.1 the fused models underperform. Moreover, calibration does not improve the scores. For this reason, I am going to omit the calibrated version of the models.

All in all, after the calibration phase the best model is still the Tied Diag GMM without calibration. In the evaluation phase of the project, I am going to evaluate the best GMM model, as well as the uncalibrated versions of the fused models, and the best SVM and Q-LogReg models. For the latter two, I am going to use the calibrated versions as they perform better in the primary working point than the uncalibrated versions.

	Poly-SVM (PCA 8, Z-normed, $C = 10^{-2}$)			
	$\pi = 0.05$	$\pi = 0.09$	$\pi = 0.5$	$\pi = 0.9$
minDCF	0.415	0.273	0.090	0.234
Uncalibrated actDCF	0.624	0.473	0.118	0.260
Calibrated actDCF	0.521	0.352	0.118	0.239

Table 3.3: Poly-SVM minimum, uncalibrated and calibrated actual DCF scores

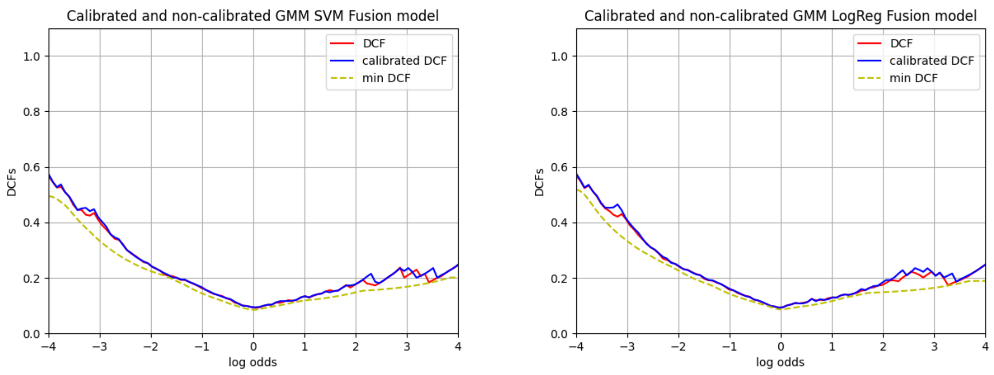


Figure 3.2: Uncalibrated and calibrated actual DCFs compared to minimum DCFs of the 2 fusion models

	Tied Diag GMM - Poly-SVM fusion			
	$\pi = 0.05$	$\pi = 0.09$	$\pi = 0.5$	$\pi = 0.9$
minDCF	0.325	0.248	0.85	0.154
Uncalibrated actDCF	0.389	0.280	0.093	0.180
Calibrated actDCF	0.401	0.277	0.094	0.205

Table 3.4: GMM minimum uncalibrated and calibrated actual DCF scores

	Tied Diag GMM - Q-LogReg			
	$\pi = 0.05$	$\pi = 0.09$	$\pi = 0.5$	$\pi = 0.9$
minDCF	0.325	0.254	0.087	0.151
Uncalibrated actDCF	0.389	0.269	0.093	0.191
Calibrated actDCF	0.395	0.276	0.093	0.202

Table 3.5: Q-LogReg minimum, uncalibrated and calibrated actual DCF scores

CHAPTER 4

Evaluation

Finally, as the last part of the project, I will analyze the performances of the selected models on the held-out data. I will explore the design choices that I have made during development, too. At this point, I do not train any new models, I use the parameters and hyperparameters I defined during the training phase. I am only going to consider those models that performed well during the training phase.

4.1 Best model analysis

First of all, I showcase the performance of the models that I have chosen as best during the training phase as well as the fused models. The best model during the training phase was the tied diagonal GMM with 8 components, without PCA and Z-normalization. Other candidate models were the Poly-SVM (PCA 8, Z-normed, $C = 10^{-2}$) and quadratic logistic regression (no PCA, Z-normed, $\lambda = 10^{-3}$). For these two models, I am going to use the calibrated versions. I am going to include in this analysis the fused models with which I experimented during the development phase.

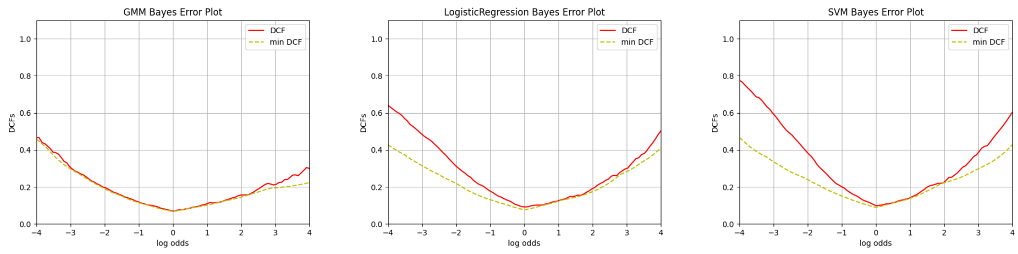


Figure 4.1: Uncalibrated actual DCFs compared to minimum DCFs of the best 3 models

Similarly to the development phase 4.1, the best classifier appeared to be the GMM model, followed by the SVM and logistic regression models. In terms of calibration, my decisions were correct, the GMM is well-calibrated, but, the SVM and logistic regression models need calibration.

By analyzing the DET plot of the best 3 classifiers 4.2 an interesting observation can be made. Even though in our primary working point the GMM model has proven to be the best, in some configurations the Q-LR model outperforms it.

Again, the fusion of SVM and logistic regression with GMM 4.4 boosts the performances of the single discriminative models. In the test data they actually perform slightly better 4.1 than the standalone GMM classifier. This can be due to the fact that in some configurations the logistic

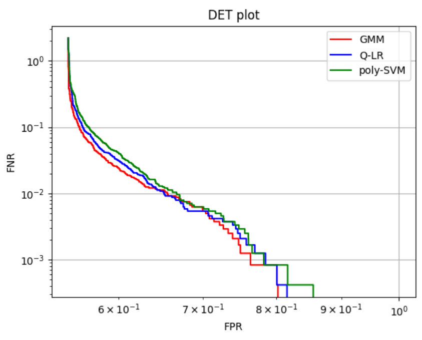


Figure 4.2: DET plot of the best 3 models

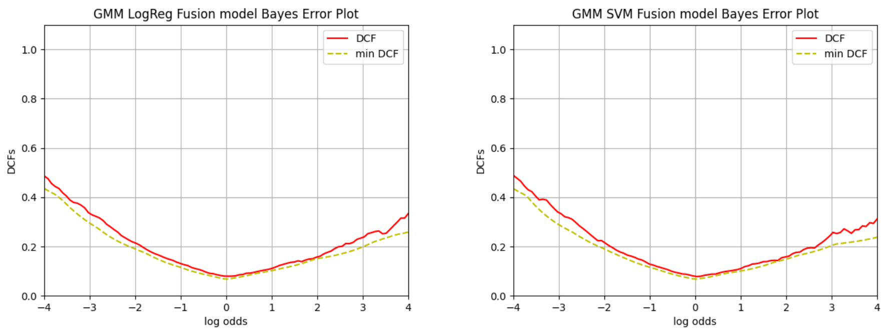


Figure 4.3: Uncalibrated actual DCFs of fusion models compared to minimum DCFs

regression has better performance as I discussed in terms of the DET plot 4.2. This means that my selection of classifier was not optimal. On the other hand, the differences in performance are not significant.

After calibrating the discriminative models I can deduce that it appeared to be useful, and the performances particularly increased.

4.2 Design choice analysis

Secondly, I am going to analyze the design choices that I made during the development phase in terms of hyperparameters. In this analysis, I am going to include the best Gaussian as well. I will resort to the minimum DCF as the performance metric, only in the primary working point.

	minDCF				actDCF			
	$\pi = 0.05$	$\pi = 0.09$	$\pi = 0.5$	$\pi = 0.9$	$\pi = 0.05$	$\pi = 0.09$	$\pi = 0.5$	$\pi = 0.9$
(1) Tied Diag GMM	0.291	0.220	0.069	0.153	0.296	0.225	0.070	0.157
(2) Q-LogReg	0.309	0.245	0.078	0.193	0.475	0.370	0.092	0.218
(3) Poly-SVM	0.329	0.265	0.091	0.234	0.582	0.447	0.099	0.252
(1) + (2)	0.289	0.214	0.068	0.156	0.329	0.244	0.080	0.174
(1) + (3)	0.283	0.216	0.068	0.161	0.333	0.247	0.079	0.176

Table 4.1: minimum and actual DCFs of the candidate models on the test data

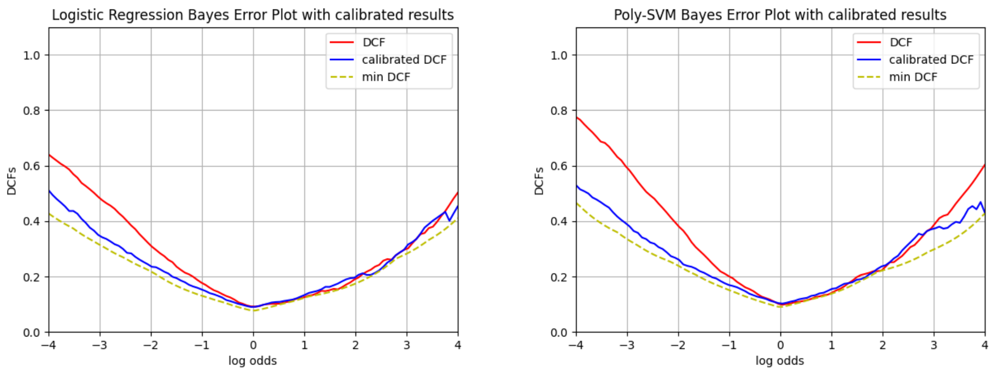


Figure 4.4: Calibrated actual DCFs of fusion models compared to actual and minimum DCFs of the discriminative models

4.2.1 Logistic regression

For the logistic regression model, I bypassed the linear models as this type of classifier performed noticeably worse during the development phase. First, I study the optimal choice of the lambda hyperparameter, without prior weighting.

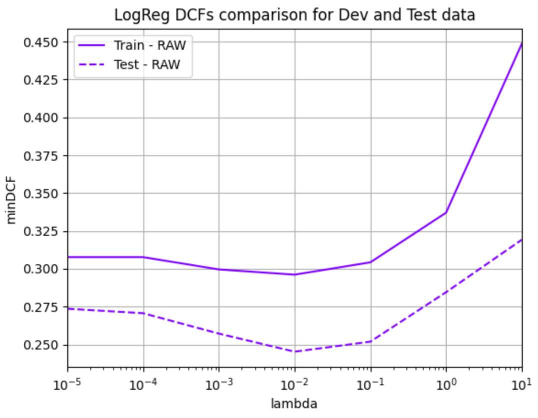


Figure 4.5: Q-LogReg (no PCA, Z-normed) optimal lambda values on train and test data

It can be seen 4.5, 4.3 that on the evaluation set the best lambda hyperparameter is 10^{-2} as opposed to 10^{-3} obtained as best during the development phase. [1] This leads to a sub-optimal choice, the performance in the evaluation set would have been 4 per cent better, did I had chosen the optimal value. The minDCF values in general improved on the evaluation set.

Next, I review the optimal choice of preprocessing and the effects of z-normalization with the lambda that appeared to be the best on the evaluation set.

By looking at the table it can be pointed out that the best configuration on the evaluation data in terms of Z-normalization and choice of dimensionality reduction is the same as the model I chose best during the development phase. This means that the only not optimal choice I made was the choice of the lambda hyperparameter.

[1]The minDCF values on the train set during evaluation slightly differ from the values I obtained during the development phase. This is because I did not fix the random seeds properly. The decisions I made were based on the values obtained during development.

λ	Dev set minDCF	Dev set minDCF during eval	Eval set minDCF
10^{-5}	0.327	0.308	0.273
10^{-4}	0.313	0.308	0.271
10^{-3}	**0.275**	0.300	0.257
10^{-2}	0.287	0.296	**0.245**
10^{-1}	0.303	0.304	0.251
10^{0}	0.338	0.337	0.285
10^{1}	0.374	0.448	0.319

Table 4.2: Q-LR (no PCA, Z-normed) with different λ in dev and eval set

PCA	minDCF on raw data	minDCF on z-normed data
-	0.268	**0.245**
9	0.263	0.247
8	0.253	0.254

Table 4.3: Q-LR with $\lambda = 10^{-2}$ in different configurations

4.2.2 SVM

Poly SVM

As I did for the logistic regression model, I skipped the linear SVM model for this type of classifier too, as this type performed worse during the development phase. First, I study the optimal choice of the C hyperparameter in the case of the polynomial model.

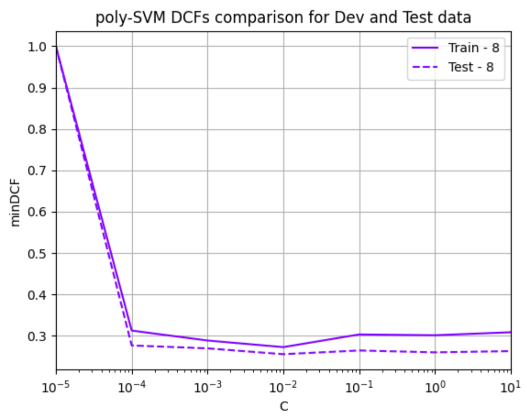


Figure 4.6: Poly-SVM (PCA 8, Z-normed) optimal C values on train and test data

We can see by looking at the picture 4.6 that the optimal value of C was the same for both the test and the train set. For this reason, I am not going to report the results in tabular form.

The best configuration in terms of Z-normalization and preprocessing was different in this case for the evaluation and for the development dataset 4.6. Z-normalization is clearly effective on the dataset, however, the choice of directions kept with PCA preprocessing was not ideal. On the evaluation set the configuration with 9 directions kept performs better. The performance loss due to the non-optimal choice of parameters is again around 4%.

PCA	minDCF on raw data	minDCF on z-normed data
-	0.273	0.257
9	0.279	0.255
8	0.258	0.265

Table 4.4: Poly-SVM with $C = 10^{-2}$ in different configurations (chosen - blue, optimal - red)

RBF SVM

I check the best C values the same way for the RBF SVM model, nonetheless, as this model performed worse on the training set I do not analyze different γ values, only the one that performed best during development, namely 0.001.

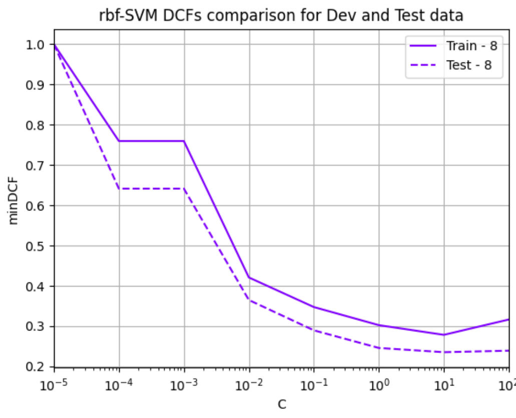


Figure 4.7: RBF-SVM (PCA 8, raw) optimal C values on train and test data

Examining the picture, it can be deduced that the best lambda parameter is the one chosen during the development phase. Nevertheless, the performance, even though similar to the polynomial on the development set is much better on the evaluation set. Hence, I am going to investigate the effects of different preprocessing techniques.

PCA	minDCF on raw data	minDCF on z-normed data
-	0.240	0.438
9	0.240	0.446
8	**0.235**	0.447

Table 4.5: RBF-SVM with $C = 10^1$ in different configurations

It can be seen that the optimal solution is the one determined during the development phase. However, the performance jump of the RBF-SVM on the evaluation set could not have been anticipated based on the development data which resulted in a non-optimal choice of classifier.

4.2.3 GMM

Finally, for the different GMM classifiers, I studied the optimal choice of components and preprocessing techniques (PCA and Z-normalization) on the evaluation set. This analysis also included the Gaussian classifiers as these are GMMs, with 1 component. It should be remembered though, that the Naive-Bayes Gaussian classifier is not the same as the diagonal GMM model, so this classifier cannot be checked in this context. Yet, this is not a problem since the best-performing classifier from the Gaussians was the MVG. I will only report the best configuration from each type of GMM.

	PCA & Z-norm	minDCF on Dev	minDCF on Eval
Full GMM - 4 comp	9 & Z-normed	0.294	0.222
Tied GMM - 8 comp	8 & -	0.261	0.212
Diag GMM - 8 comp	10 & -	0.260	0.213
Tied-diag GMM - 8 comp	- & -	0.244	0.220

Table 4.6: Best GMM classifiers of each type in best configuration (chosen - blue, optimal - red)

It can be seen that even though the best configuration was chosen based on the development phase, it did not turn out to be the best one on the evaluation set. The difference in performance would have been around 4% better. All in all, my choice of classifier was close to the optimal one. Furthermore, it was confirmed that the Gaussian models do not perform better than the GMM models.

4.3 Conclusion

All in all, the design choices, hyperparameters and configurations that I chose were reasonable. The evaluation showed that the chosen model(s) performed well on held-out data, which means that the test data's population was similar to the development one's. Moreover, the classifiers were generic enough to work well on slightly different data. Even though the model that was chosen as the primary classifier (best GMM based on development data) did not turn out to be the most effective classifier on the evaluation data, its performance was only slightly worse than the best classifier's (4 %). It can be said that the approach that I followed was effective.