Biometric Identity Verification

s308754 Leone Andrea
s308762 Pietromatera Aldo

A report presented for the course of
Machine Learning & Pattern Recognition

March 2023 - June 2023

Abstract

The goal of the project is to build a model that best fits for the Biometric Identity Verificaion task (i.e. to verify whether an identity claim is correct based on biometric characteristics of a subject), exploiting the Machine Learning algorithms shown during the *Machine Learning & Pattern Recognition* classes. We will discuss how different models perform, explaining pros and cons.

Overview

A speaker verification system should verify whether a spoken utterance of an unknown person corresponds to a claimed identity. This involves comparing the unknown person utterance with those belonging to the claimed identity, to assess whether the unknown utterance is similar enough to the latters. For simplicity, in this project it is assumed that the claimed identity utterances are just one. The problem can then be cast as a **binary problem** over pairs of utterances, i.e., given the unknown utterance and the claimed identity utterance, the classifier should decide whether the pair belongs to the same speaker (label 1) or different speaker (label 0) class.

The dataset consists of synthetic embedding pairs, i.e., each sample consists of a pair of embeddings, for a total of 10 dimensions, 5 of which are for the unknown identity while the other 5 are for the claimed one. Note that features have no specific physical interpretation.
The training set consists of 3500 samples while the test set has 7500 samples.

For the sake of simplicity, we are going to ignore the fact that samples represent pairs.

Contents

1 Feature Analysis **2**

2 Training & Validation **5**
2.1 Introduction . . . 5
2.2 Multivariate Guassian Classifier . . . 5
2.2.1 Expectations . . . 5
2.2.2 Results . . . 6
2.3 Logistic Regression . . . 6
2.3.1 Expectations . . . 6
2.3.2 Results . . . 7
2.4 Support Vector Machine - SVM . . . 8
2.4.1 Expectations . . . 8
2.4.2 Results . . . 8
2.5 Gaussian Mixture Models - GMM . . . 10
2.5.1 Expectations . . . 10
2.5.2 Results . . . 11
2.5.3 More considerations about GMM models . . . 14
2.6 More considerations about linear models . . . 15
2.7 Best models recap . . . 16
2.8 Score calibration . . . 16
2.9 Model Fusion . . . 19
2.10 Final comparison . . . 20

3 Experimental Results **21**
3.1 Introduction . . . 21
3.2 Best models . . . 21
3.3 MVG . . . 21
3.4 Logistic Regression . . . 22
3.5 SVM . . . 23
3.6 GMM . . . 24

4 Conclusions **26**

1 Feature Analysis

Here are histograms for each of the 10 features.

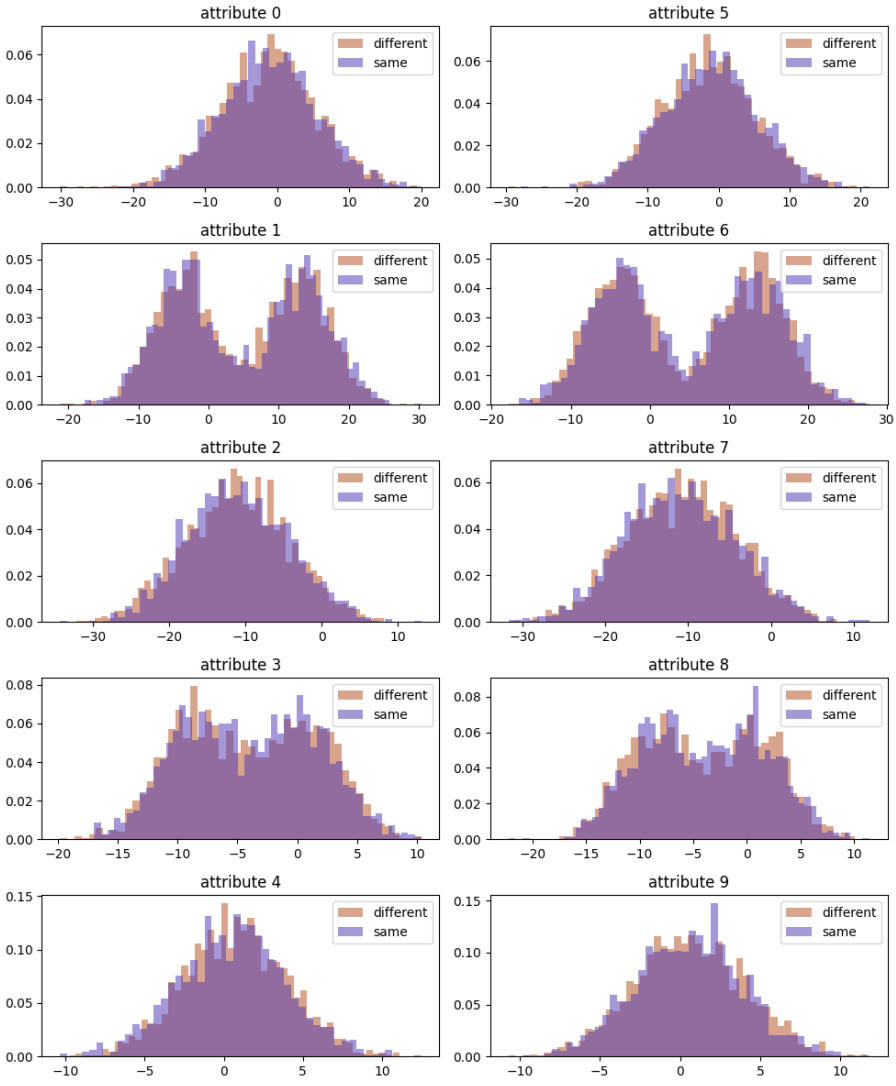


Figure 1: histograms

We can see that most of the features are approximately gaussian distributed, so we expect gaussian models to perform well. Features 1, 3, 6, 8 may be better approximated by a combination of gaussians.
Next, we report scatter plots for each pair of features and heatmaps for the entire dataset and for each class.

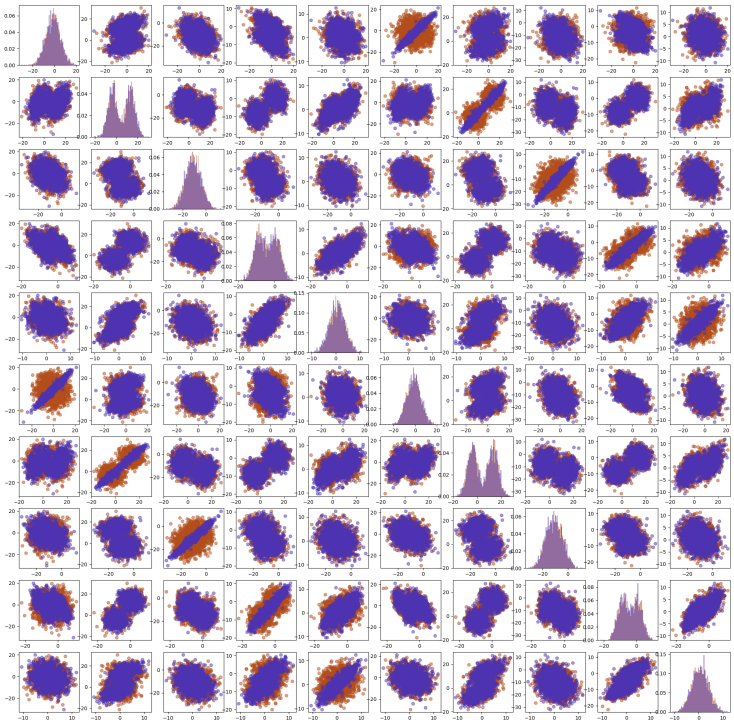
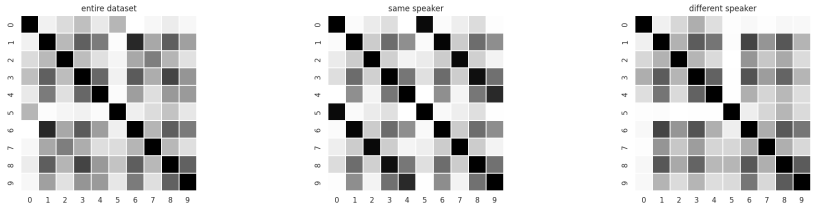


Figure 2: scatter plots & histograms



(a) entire dataset (b) same spaker (c) different speaker

Figure 3: heatmaps

From the plots we understand that linear models will probabily perform very bad, because classes don't seem to be linearly separable. So, for example, the

Tied-MVG model, linear LR and linear SVM won't be good.
Moreover, we can notice that there is a certain degree of correlation between features, so we may try to use PCA to reduce the dimensionality of the dataset and to be able to use a Naive approach. Also, correlation is different depending on the class, in fact for class 1 there are more correlations (which expected, given the problem description).
From the following plot we can see that it's possible to remove up to 4 features retaining more than 95% of the dataset variance.

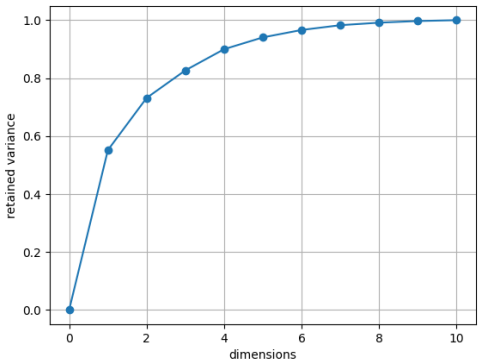


Figure 4: retained variance vs. dimensions

Finally, based on the scatter plots, we don't think that linear model will perform well, since classes don't seem to be linearly separable. We have also verified this assumption by plotting the samples along the LDA direction. However we will also verify this assumption later in the "Training & Validation" phase.

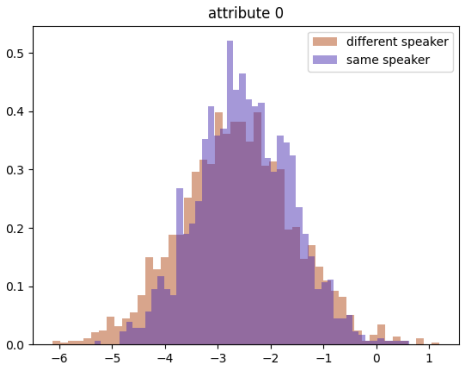


Figure 5: LDA direction

2 Training & Validation

2.1 Introduction

In order to write the report, the collected results have been taken as the output of the following classifying models' list:

- Gaussian Classifiers
  - Multivariate Gaussian Classifier (MVG)
  - MVG + Naive assumptions
  - MVG + Tied Covariance
- Logistic Regression
- Support Vector Machine
- GMM Classifiers

For what concerns validation, it is necessary to make the following clarifications:

- To search the best hyperparamters configuration for each model and to asses the effect of PCA, we use K-fold cross-validation with K=5.
- Prior-weighted models have been trained with the effective prior $\pi_T = 0.1$. In fact, our working point is $(\pi_T, C_{fn}, C_{fp}) = (0.1, 1, 1)$.
- While choosing the best model for our application, we will also consider how the model performs with respect to a different one $(\pi_T, C_{fn}, C_{fp}) = (0.5, 1, 1)$.
- We will refer to the application with $\tilde{\pi} = 0.1$ as *Target Application* or *App A*, and to the application with $\tilde{\pi} = 0.5$ as *App B*.

2.2 Multivariate Guassian Classifier

2.2.1 Expectations

Since our features are in some way correlated, we expect that MVG with naive assumptions will not be good in the case we do not use PCA. As we already said, we expect linear models to perform bad because of non-linearly separable classes. For this reason we won't try the Tied+Naive model but we will try the Tied one for demonstration purposes.
On the other hand, we expect the standard MVG to perform well although not all the features follow a Guassian distribution.

2.2.2 Results

The best results (shown in Table 1) were brought from MVG with Naive assumption and PCA(8), therefore, we choose it as the best model for gaussian classifiers. The obtained minDCF is 0.260 for the target application. The same model gives a minDCF of 0.082 for App B, which is sub-optimal since the full-covariance model with PCA(8) gives a minDCF of 0.081.

Table 1: Gaussian Classifiers

Gaussian classifiers MVG (minDCF)						
	$\tilde{\pi} = 0.1$			$\tilde{\pi} = 0.5$		
PCA	**Full Cov.**	**Naive**	**Tied**	**Full Cov.**	**Naive**	**Tied**
-	0.267	1.0	1.0	0.083	0.976	0.946
10	0.267	0.269	1.0	0.083	0.086	0.945
9	0.270	0.269	1.0	0.083	0.086	0.939
8	0.272	**0.260**	1.0	**0.081**	0.082	0.937
7	0.268	0.270	1.0	0.084	0.086	0.907
6	0.292	0.291	1.0	0.087	0.090	0.848

We can see that Naive models perform well only when PCA is applied. This is due to the fact that PCA make features uncorrelated. We also tried MVG Tied in order to verify that linear models perform very bad.
So, as already said, the best gaussian model is

- **MVG Naive + PCA(8)**: this model gave a minDCF of 0.260 for the target application and 0.082 for *App B*, which is close to optimal.

2.3 Logistic Regression

2.3.1 Expectations

We don't expect the standard Logistic Regression model to perform well, since it provides linear separation rules. However, if we apply the feature expansion

$$\phi(\mathbf{x}) = \begin{bmatrix} \text{vec}(\mathbf{x}\mathbf{x}^T) \\ \mathbf{x} \end{bmatrix}$$

we can obtain a quadratic separation surface, i.e. better results, since we already saw that the Full Covariance MVG model performs well.
Since the logistic regression model tends to learn the empirical prior of the training dataset which in our case is very different from the effective prior of the target application, we have trained the model with a prior-weighted version of the objective function

$$R(\mathbf{w}, b) = \frac{\lambda}{2}||\mathbf{w}||^2 + \frac{\pi_T}{N_T} \sum_{i|z_i=1} l(z_i s_i) + \frac{1-\pi_T}{N_F} \sum_{i|z_i=-1} l(z_i s_i)$$

2.3.2 Results

We have verified the assumption that linear LR doesn't work well, in fact we have always obtained a minDCF of 1.0 (or close to 1.0).

By applying features expansion, instead, we obtain the results in Table 2.

Table 2: LR with feature expansion

QLR (minDCF)						
	PCA					
λ	**RAW**	**10**	**9**	**8**	**7**	**6**
$\mathbf{10^{-5}}$	0.325	0.319	0.319	0.31	0.346	0.321
$\mathbf{10^{-4}}$	0.326	0.33	0.318	0.312	0.329	0.316
$\mathbf{10^{-3}}$	0.33	0.324	0.312	0.313	0.346	0.328
0.01	0.328	0.333	0.317	0.31	0.332	0.309
0.1	0.322	0.32	0.318	**0.306**	0.337	0.32
1	0.351	0.352	0.344	0.343	0.337	0.332
10	0.546	0.553	0.538	0.544	0.535	0.516

We can see that, as expected, quadratic surfaces are more suitable for separating the data. The table reports the best value of λ for each PCA, and the overall best value is obtained again with PCA(8). The optimal value of λ in this case is 0.1.
About this model, we can say that given the relatively high value of lambda we expect good generalization on the evaluation set.
However, the minDCF value corresponding to the best model (0.306) is a bit worse than the one found with gaussian models.

We have also tried to apply z-normalization to the features (after PCA), and obtained slightly better results (Table 3).

Table 3: LR with z-norm + feature expansion

QLR z-norm $\tilde{\pi} = 0.1$ (minDCF)						
	PCA					
λ	**RAW**	**10**	**9**	**8**	**7**	**6**
$\mathbf{10^{-5}}$	0.341	0.331	0.312	0.306	0.293	0.316
$\mathbf{10^{-4}}$	0.326	0.327	0.319	0.302	**0.291**	0.318
$\mathbf{10^{-3}}$	0.399	0.349	0.346	0.334	0.298	0.334
0.01	0.634	0.465	0.448	0.494	0.446	0.397
0.1	0.832	0.594	0.592	0.658	0.663	0.671
1	0.949	0.61	0.623	0.694	0.713	0.729
10	0.952	0.573	0.583	0.664	0.688	0.682

In this case the optimal minDCF value has been found with PCA(7) and λ equal

to 0.0001. The corresponding minDCF value is still higher than the best MVG model. Moreover, given the smaller value of λ with respect to the non normalized version of the model, we expect worse generalization for the evaluation set. So, regarding LR, the best models are

- **QLR with $\lambda = 0.1$ + PCA(8)**: this model gave us a minDCF of 0.306 for the target application with working point (0.1, 1, 1). The minDCF for the working point (0.5, 1, 1) is instead 0.085, which is also the lowest minDCF value among those given by all the models in Table 2.
- **QLR with $\lambda = 0.0001$ + PCA(7) + z-norm**: this model gave us a minDCF of 0.291 for the target application. The minDCF for *App B* is instead 0.091, which is 0.005 higher than the best minDCF, obtained by applying PCA(8) instead of PCA(7). However, since the difference is small, we can say that the selected model works well for both applications.

2.4 Support Vector Machine - SVM

2.4.1 Expectations

As for the LR model, we don't expect the linear SVM model to perform well. However, we should get good results at least with the polynomial kernel of degree 2. We will also try polynomial kernel of degree 3 and RBF kernel to see how they perform.
The best hyperparameters are selected by means of grid search.

2.4.2 Results

Again, linear SVM gives us a minDCF of 1.0 (or close to 1.0), as expected, so we will report data only for SVM with kernels.

Table 4 shows how the poly kernel with d=2 performs for the target application.

Table 4: Polynomial SVM d=2

Polynomial SVM $\tilde{\pi} = 0.1$ (minDCF)							
		PCA					
C	**K**	**-**	**10**	**9**	**8**	**7**	**6**
0.0001	**0.1**	0.356	0.339	0.34	0.38	0.468	0.566
	1	0.338	0.327	0.329	0.355	0.46	0.58
	10	0.324	0.335	0.316	0.322	0.326	0.354
	100	0.297	0.296	0.313	0.299	**0.288**	0.315
0.001	**0.1**	0.536	0.515	0.591	0.576	0.613	0.6
	1	0.551	0.525	0.566	0.46	0.665	0.645
	10	0.512	0.671	0.534	0.574	0.412	0.526
	100	0.665	0.529	0.722	0.548	0.46	0.462

The best configuration of hyperparameters for this model seems to be $(C, K) = (10^{-4}, 100)$, which gives the best results when combined with PCA(7).
For comparison purposes, we have also seen that for *App B* this model gives a minDCF of 0.089, which is also obtained with same hyperparameters and PCA(6) or PCA(9). By reducing the value of K we get the same minDCF also with other configurations, but we prefer those with higher K because of a lower regularization on the bias term. So, we can say that the model with $(C, K) = (10^{-4}, 100)$ and PCA(7) is the best performer for both the target application and the *App B* ($\tilde{\pi} = 0.5$).

From Table 5 we can see that z-normalization is able to improve the performances for the SVM model with polynomial Kernel (d=2).

Table 5: Polynomial SVM with z-norm d=2

Polynomial SVM with z-norm $\tilde{\pi} = 0.1$ (minDCF)							
		PCA					
C	K	RAW	10	9	8	7	6
0.001	**0.1**	0.8	0.565	0.558	0.607	0.633	0.681
	1	0.802	0.549	0.548	0.584	0.605	0.625
	10	0.8	0.437	0.43	0.444	0.346	0.328
	100	0.801	0.425	0.425	0.437	0.327	0.324
0.01	**0.1**	0.393	0.344	0.344	0.311	0.3	0.323
	1	0.363	0.321	0.321	0.287	0.276	0.321
	10	0.35	0.297	0.298	**0.269**	0.284	0.309
	100	0.862	0.47	0.588	0.605	0.634	0.457
0.1	**0.1**	0.316	0.303	0.313	0.288	0.288	0.299
	1	0.318	0.308	0.314	0.293	0.288	0.301
	10	0.314	0.302	0.305	0.301	0.28	0.295
	100	0.918	0.799	0.881	0.779	0.884	0.88
1	**0.1**	0.335	0.326	0.303	0.295	0.274	0.305
	1	0.35	0.336	0.299	0.298	0.279	0.295
	10	0.384	0.338	0.313	0.315	0.324	0.314
	100	0.978	0.928	0.875	0.973	0.917	0.928

In this case the best result (minDCF=0.269) is obtained with $(C, K) = (0.01, 10)$ and PCA(8). This application gives a minDCF=0.097 for the other application, which is not the best result. In fact it's possible to obtain a minDCF=0.086 with $(C, K) = (0.1, 1)$ and no PCA for the application with $\tilde{\pi} = 0.5$.

The polynomial kernel with d=3 gives very high values of minDCF (0.9 - 1.0) for the target application, so we discard it.

The RBF kernel without z-normalization performs a bit better, giving a minDCF of 0.409 for the target application, and 0.113 for the application with $\tilde{\pi} = 0.5$. This result has been obtained with hyperparameters $(C,\ K,\ \gamma) = (1,\ 0.1,\ 0.01)$

and PCA(7). However, this is still a bad result compared to those obtained until now.
A "3D" grid search has been done over hyperparameters (C, K, γ) with different values of PCA. The considered hyperparameters values are:

- $C = 10^{-5},\ 10^{-4},\ 10^{-3},\ 10^{-2},\ 10^{-1},\ 1$
- $K = 0.1,\ 1,\ 10,\ 100,\ 1000$
- $\gamma = 10^{-3},\ 10^{-2},\ 10^{-1},\ 1$

From the grid search it also turns out that for the application $\tilde{\pi} = 0.5$ the best configuration is always $(C,\ K,\ \gamma) = (1,\ 0.1,\ 0.01)$ but with PCA(6), giving a minDCF of 0.106 instead of 0.113.

At this point, let's see if z-normalization can improve the performances. Since SVM training is slow and we didn't obtain very promising results with RBF kernel, we will just consider PCA(7) and PCA(8), since they gave the best results until now.
Z-normalization is able to improve the performances of the SVM + RBF kernel classifier, making it comparable with the other optimal models we have found up to now. The best minDCF for the target application is 0.289 and it is obtained with hyperparameters $(C,\ K,\ \gamma) = (1,\ 0.1,\ 0.16)$ + PCA(7). The hyperparameters configuration has been found by means of a grid search with refining around the most promising point.
This model performs well also for *App B* ($\tilde{\pi} = 0.5$) giving a minDCF of 0.103, however a lower minDCF (0.099) is obtained with $\gamma = 0.1$.

So, regarding SVM, the best models are

- **SVM poly** $d = 2$**,** $(C, K) = (0.01, 10)$ **+ PCA(8) + z-norm**: this model gave us a minDCF of 0.269 for the target application and 0.097 for *App B*. The result for *App B* is suboptimal, since it's possible to obtain a minDCF=0.086 with $(C, K) = (0.1, 1)$ and no PCA.
- **SVM RBF** $(C, K, \gamma) = (1, 0.1, 0.16)$ **+ PCA(7) + z-norm**: this model gave us a minDCF of 0.289 for the target application, which is slightly worse than the previous model. For *App B* the obtained minDCF is 0.103, which is a bit worse than the lowest minDCF obtained with SVM RBF + z-norm (0.099).

2.5 Gaussian Mixture Models - GMM

2.5.1 Expectations

In this case, since some features are distributed as sums of gaussians, we expect good performances for this model, at least as good as for the MVG classifier.

2.5.2 Results

We obtained that the best GMM model (apart from cases with only one component which are equivalent to MVG models) is the Tied GMM with no PCA in Figure 7. It reaches a minDCF of 0.267 for the target application by modelling the *same speaker class* with 2 components and the *different speaker class* with 2 components. We choose the one with less components in order to estimate less parameters and try to avoid overfitting. In addition, we should get more generalization for unseen data.
The fact that the Tied model performs well makes sense, since from the scatter plots of features we can see that clusters are similarly distributed around their mean.
Concerning *App B*, the minDCF for the just cited model is 0.083. We can also observe that the best model, with no PCA, for *App B* is the Tied one using 16 components for *same speaker class* and 4 components for the other class. It has a minDCF of 0.078.

As we expected, the two diagonal models do not produce good results (without PCA) because we have moderate correlation among features as we have seen previously in the heatmap, in fact they always gave us a minDCF above 0.9 for the target application. Because of this, we won't report any graph for these diagonal models.

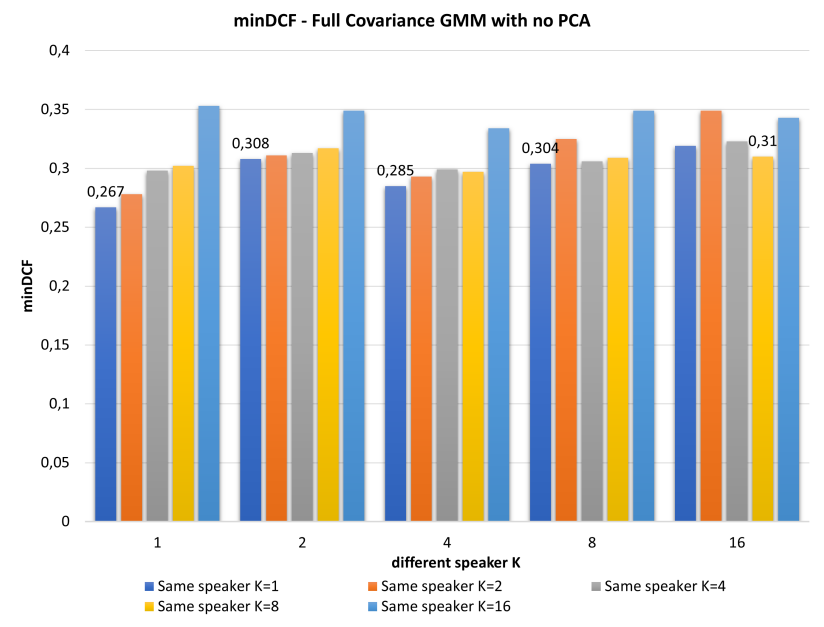


Figure 6: minDCF - Full Covariance GMM with no PCA

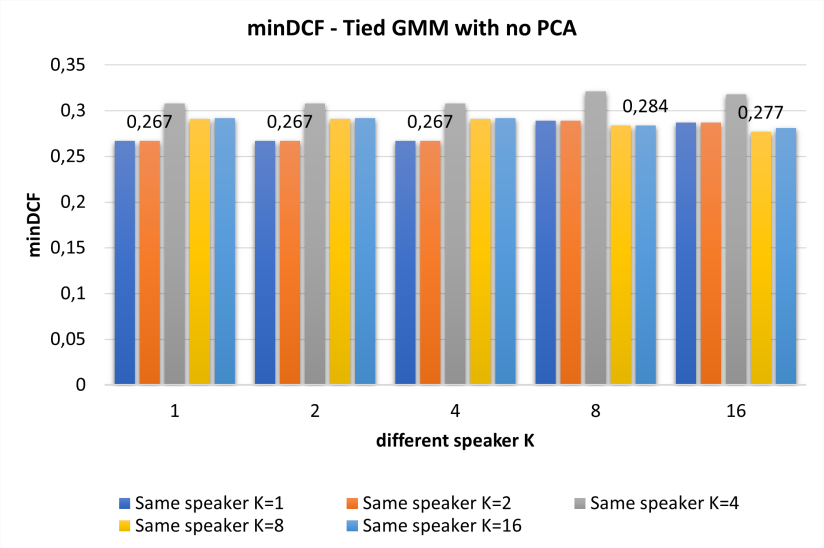


Figure 7: minDCF - Tied GMM with no PCA

We also tried to apply PCA to see if we could get better performances. Different values of PCA were used, but the best results have been obtained with PCA(8), for which we reported the following figures.

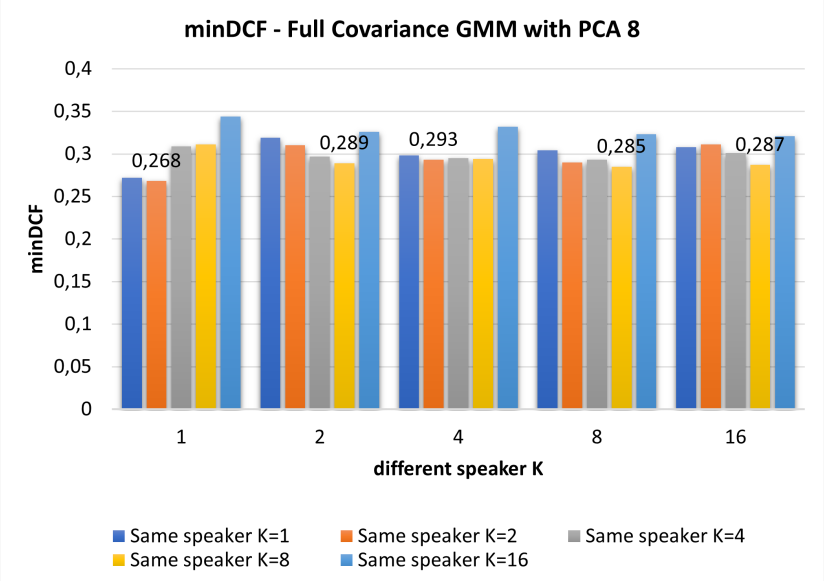


Figure 8: minDCF - Full Covariance GMM with PCA(8)

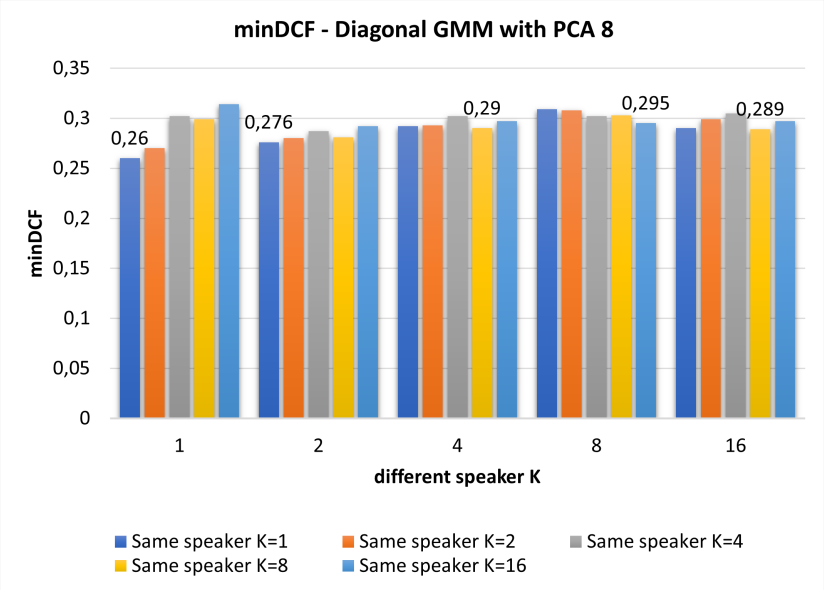


Figure 9: minDCF - Diagonal GMM with PCA(8)

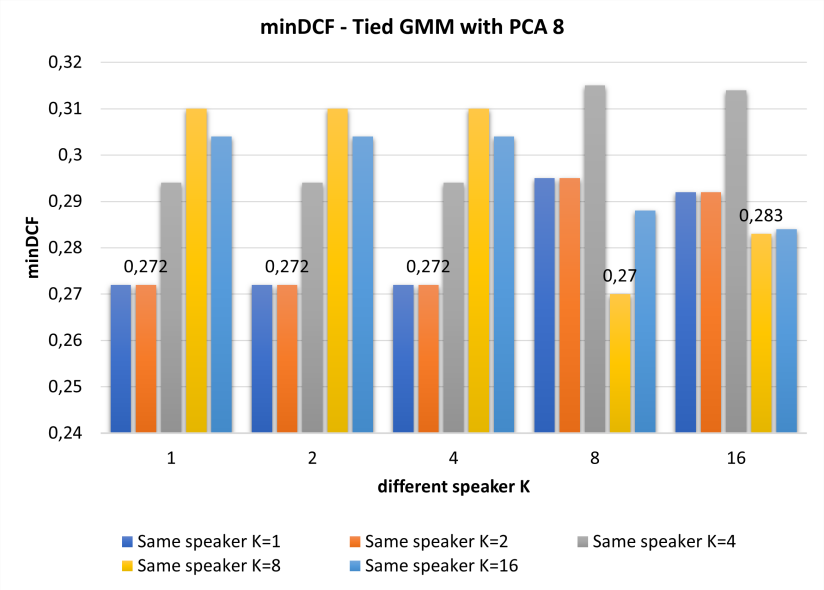


Figure 10: minDCF - Tied GMM with PCA(8)

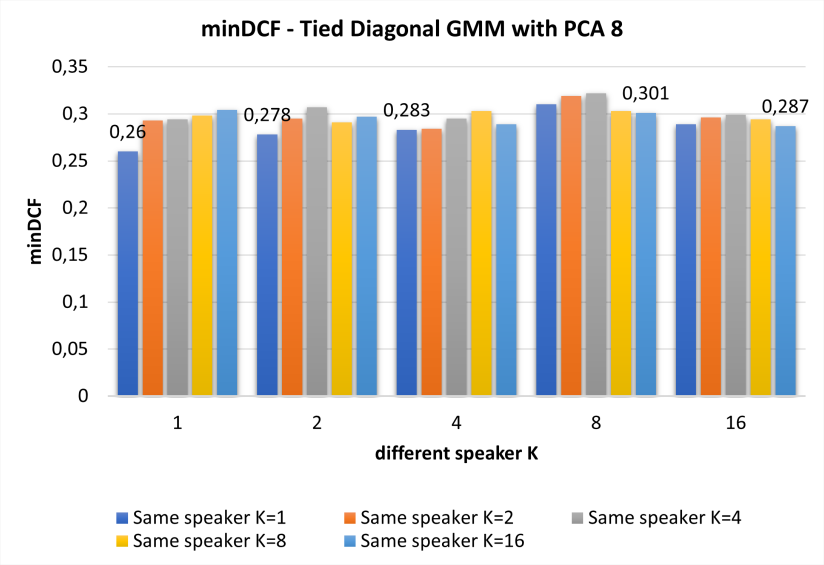


Figure 11: minDCF - Tied Diagonal GMM with PCA(8)

We can see in Figure 10 that also Tied GMM with PCA(8) using 8 components for *same speaker class* and 8 for the *different speaker class* has good results ($minDCF = 0.27$), which is slightly worse with respect to the best model with no PCA.
App B has, instead, a minDCF of 0.088 for the just cited model.
The best model with PCA(8) for *App B* is the Full Covariance one using 2 components for *same speaker class* and 1 component for the other class. It has a minDCF of 0.076.

2.5.3 More considerations about GMM models

For some reason the results obtained with the GMM classifiers aren't better than those obtained with the MVG classifiers, even thought the features analysis would suggest us differently. At least, this holds in terms of minDCF: we will see later what happens when considering actual DCF.

Because of this, we have also tried an other approach: we have uncorrelated the features of the training set by applying PCA (first PCA(10), then also 9, 8, ...). We have seen that in any case the distribution of the first PCA feature could be described very well by a 2-components GMM, while all the other features were distributed as simple Gaussians.
Since in this case we were working with uncorrelated features, we have tried to build an ad-hoc Naive-Bayes GMM model by estimating different parameters for each class and for each feature separately. In particular we have estimated

a 2-components GMM distribution for the first feature, and a single-component GMM distribution for the others.
However, this new model didn't give better results w.r.t. those obtained until now, so we discarded it.

2.6 More considerations about linear models

By taking into account the fact that our samples represent pairs, we notice that corresponding features (0-5, 1-6, ..., 4-9) have a very similar distribution. Also, given the problem description, we think that to understand if the pair embeddings refer to the same speaker or to different speaker, we can just consider differences between corresponding features.
Thanks to these observations, we have tried to preprocess our data to obtain a new dataset with just 5 features per sample, corresponding to the difference of corresponding features in the original dataset.
At this point, from the scatter plots of the transformed data, we have noticed that the data was distributed as a "sphere", which is why linear classify didn't perform well. In particular, the samples corresponding to the *same speaker* class are distributed nearer to the sphere center, while the samples of *different speaker* class are spread farther from the center. This means that we might be able to perform classification by looking at the distance of a point from the center of the data.

After all these considerations, we have tried to further preprocess our 5D samples transforming them from cartesian to 5D polar coordinates (i.e. radius + 4 angles). We could have computed just the radius, but we didn't want to lose too much information, since this is already an approximation which may not lead to good results.
Finally, we run linear classifiers on the transformed dataset collecting the following best results:

- **MVG Tied+Naive, no PCA**: minDCF = 0.315 for the target application and 0.111 for *App B*. Best value for *App B* is 0.110 obtained with PCA(4) (remember that now the dataset only has 5 features).
- **LR** $\lambda = 0.1$**, no PCA** (prior-weighted): minDCF = 0.321 for target application and 0.110 for *App B*. Best value for *App B* is 0.108 obtained with $\lambda = 10^{-5}$ or $\lambda = 10^{-4}$.
- **SVM** $(C, K) = (10^{-3}, 10)$**, no PCA**: minDCF = 0.318 for the target application and 0.110 for *App B*. Best value for *App B* is 0.109 obtained with $(C, K) = (10^{-4}, 100)$ or $(C, K) = (10^{-3}, 100)$.

We can see that **MVG Tied+Naive** with raw features is the best linear model (even thought others are very close). However, the minDCF of this model (0.315) is higher with respect to those obtained previously, which were always below 0.3. This may be because by removing 5 features computing just the

difference between corresponding ones is an approximation and we are probabily losing some information during this process.

2.7 Best models recap

Table 6 shows a recap of the best models we found up to now.

Table 6: Best models on evaluation set

Model	**minDCF**	
	$\tilde{\pi}$=0.1	$\tilde{\pi}$=0.5
MVG Naive + PCA(8)	**0.260**	0.081
QLR (l=0.0001) + PCA(7) + z-norm	0.291	0.091
poly SVM d=2, C=0.01, K=10 + PCA(8) + z-norm	0.269	0.097
RBF SVM C=1, K=0.1, $\gamma = 0.16$ + PCA(7) + z-norm	0.289	0.103
GMM Tied + no PCA + K=2 components per class	0.267	0.083

We can see that for the target application, the best model remains MVG Naive with PCA(8), so we will keep it as the best candidate model for now. We can also notice that this model is also the best for *App B*.
However we need to check if the scores given by the model are well calibrated. We will also check if the scores of other models are well calibrated, and we will eventually calibrate them. We will also try to exploit model fusion to try to improve the performances of our final model.

From now on we will always refer to these models with these hyperparameters. Also, since QLR and poly SVM provide similar separation surfaces, we decide to ignore QLR because it got worse performances on the validation data.

2.8 Score calibration

Up to now, we have considered only the minDCF metric to evaluate different models, however, the issue is that the cost that we pay depends on a threshold. The theoretical trashold is $-\log \frac{\tilde{\pi}}{1-\tilde{\pi}}$, but some models produce scores for which the optical treshold is different from the theoretical one. To overcome this issue we apply score calibration with a prior-weighted Logistic Regression model, in such a way that the *actual DCF* (actDCF) becomes as similar as possible to the minDCF.

We followed a K-fold approach also for calibration. After the training-step K-fold, we pool the scores of each fold to obtain a score set. Then we apply K-fold over the obtained scores using a different randomized shuffle in order to evaluate the model performances over the pooled scores.
While doing this we also train two additional models: one over the entire training set (used to obtain the scores) and one over the entire score set (used for

calibration). These two models will be used for inference in the evaluation phase.

We start by analizing the overall best model, i.e. MVG Naive + PCA(8).

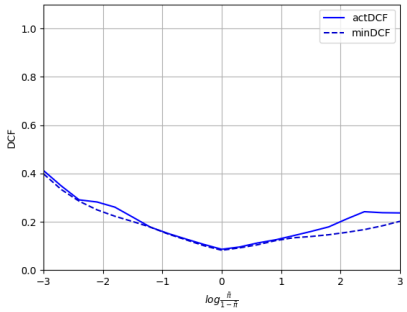


Figure 12: MVG Naive - Uncalibrated

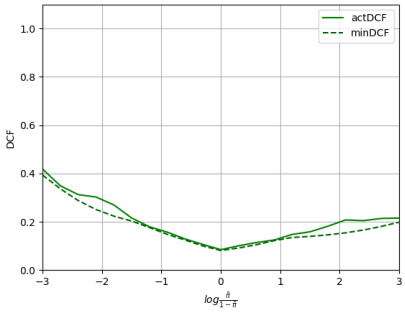


Figure 13: MVG Naive - Calibrated

We can see from Figures 12 and 13 that in this case calibration is not very usefull. Actually, it makes actual DCF a bit worse for the target application (all actDCF values can be found in the table 7 at the end of this section). Because of this, we will use the uncalibrated model in the evaluation phase.

Let's also see how calibration affects the other models.

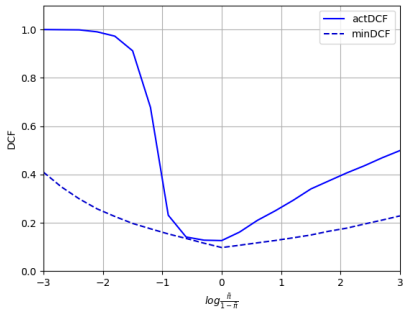


Figure 14: poly SVM - Uncalibrated

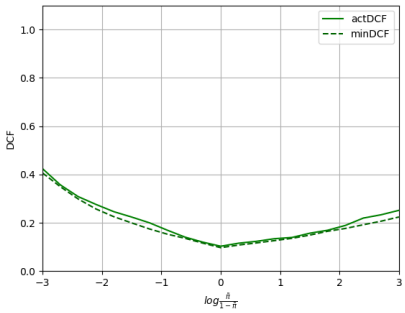


Figure 15: poly SVM - Calibrated

We can see from Figures 14 and 15 that the poly SVM scores are highly uncalibrated, in particular for the target application (for which $\log \frac{\tilde{\pi}}{1-\tilde{\pi}} = -2.2$). By applying calibration as said before, the actDCF improves a lot, becoming very similar to the minDCF.

Figures 16 and 17 show a similar result regarding RBF SVM: uncalibrated scores give a very bad actDCF, but calibration solves the problem for all the considered tresholds.

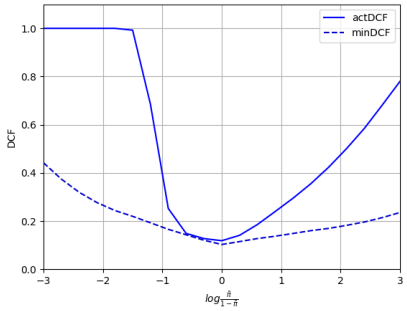


Figure 16: RBF SVM - Uncalibrated

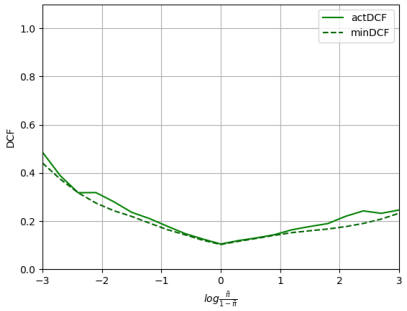


Figure 17: RBF SVM - Calibrated

Finally, let's see what happens for the GMM Tied model.

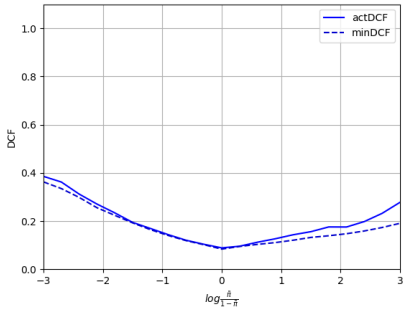


Figure 18: GMM Tied - Uncalibrated

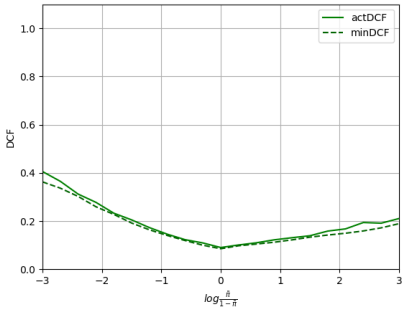


Figure 19: GMM Tied - Calibrated

From Figures 18 and 19 we notice that the scores were already well calibrated for the target application, but calibration improves the performances for tasks with an effective prior closer to 1 (rightmost part of the graphs).

Table 7 shows a recap of the obtained actDCFs.

Table 7: actDCF for selected models

actDCF	**Uncalibrated**		**Calibrated**	
Model	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.5$
(1) **MVG Naive + PCA(8)**	0.282	0.086	0.301	**0.085**
(2) **GMM Tied**	**0.268**	0.088	0.292	0.090
(3) **poly SVM + PCA(8)**	0.995	0.126	0.291	0.102
(4) **RBF SVM + PCA(7)**	1.0	0.118	0.33	0.104

The table highlights the best actDCF for each model (for the target application). We can see that the best performances are given by the GMM Tied model with 2 components per class and no PCA (with uncalibrated scores), so unless we get better results with models fusion we will choose this model as the final one to deliver.
At the same time, we notice that the best model for *App B* is the calibrated MVG Naive. However, the uncalibrated GMM Tied performances on *App B* are not significantly worse.

Table 8 shows, instead, the minDCF for the selected models. Models number refer to the models in Table 7.

Table 8: minDCF for selected models

minDCF	**Uncalibrated**		**Calibrated**	
Model	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.5$
(1)	**0.260**	0.082	0.261	**0.081**
(2)	0.277	0.083	0.272	0.085
(3)	0.270	0.097	0.269	0.097
(4)	0.289	0.103	0.287	0.103

2.9 Model Fusion

As a final step for this phase, let's see if we can combine different models at score level to improve the performances of our classifiers. In the following table we will identify different models with the number associated in Table 7.

Table 9: minDCF and actDCF for fused models

	minDCF		**actDCF**	
Models	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.5$
(1) + (2) + (3) + (4)	0.267	0.088	**0.286**	0.093
(1) + (2) + (3)	0.265	0.087	0.287	0.092
(1) + (2) + (4)	0.272	0.085	0.288	0.088
(1) + (3) + (4)	0.274	0.089	0.296	0.095
(2) + (3) + (4)	0.273	0.089	0.296	0.093
(1) + (2)	**0.261**	**0.080**	0.303	**0.084**
(1) + (3)	0.264	0.088	0.299	0.093
(1) + (4)	0.274	0.087	0.290	0.089
(2) + (3)	0.267	0.088	0.299	0.093
(2) + (4)	0.273	0.087	0.294	0.089
(3) + (4)	0.266	0.096	0.293	0.103

We can see from Table 9 that fusion doesn't give any particular benefit to the models for the target application, so the best one remains **GMM Tied**.

However, we can notice that fusion gives some improvements for *App B*, since we get an actDCF of 0.084 for MVG Naive + GMM Tied. Anyway, this is a very small improvement.

Since we didn't get any significant improvement for the target application we won't consider fused models in the evaluation phase.

2.10 Final comparison

In Figure 20 below, we consider that the two SVM models are calibreted. On the other hand, MVG and GMM are not. We can see that MVG and GMM behave in quite a similar way, in fact their DET plot is mostly overlapped. In the small region around the point (FPR, FNR) = (0.02, 0.11) SVM performs better than the other models.

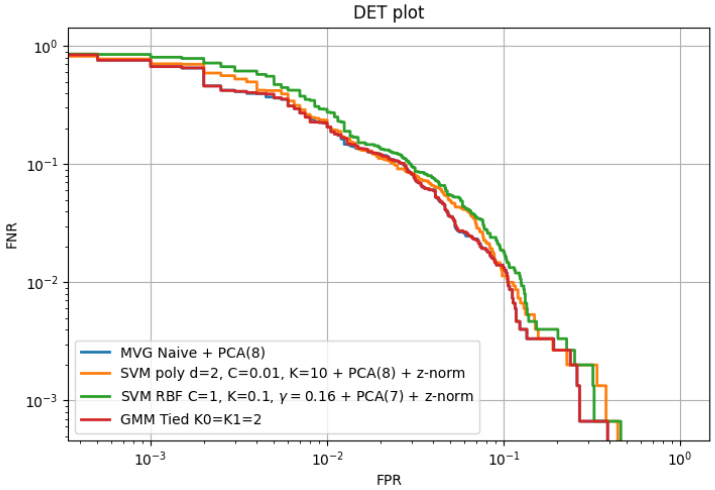


Figure 20: DET plot

Overall, the best model remains Tied GMM, together with MVG Naive. However we will select Tied GMM as the model to deliver because it has the best performances (in terms of actual cost) for the target application.

3 Experimental Results

3.1 Introduction

At this point, let's move on to the evaluation phase. We will analyze the performances on the evaluation set for the selected model, but also for other models, and we will analyze how effective our choices have been.

3.2 Best models

Table 10 shows how the "best models" selected in the training phase perform on the evaluation set.

Model	minDCF		actDCF	
	$\tilde{\pi}$=0.1	$\tilde{\pi}$=0.5	$\tilde{\pi}$=0.1	$\tilde{\pi}$=0.5
MVG Naive + PCA(8)	0.304	**0.088**	0.308	0.090
poly SVM d=2 +PCA(8) + z-norm	0.335	0.096	0.341	0.096
RBF SVM + PCA(7) + z-norm	0.336	0.099	0.344	0.100
GMM Tied + no PCA	**0.282**	0.089	0.291	0.090

Table 10: best models' minDCF on evaluation set

The models hyperparameters are intended to be the same as shown in Table 6. We can see that the chosen model (GMM Tied) is actually the best performer on the evaluation set, in terms of both minDCF and actDCF. MVG Naive obtains slightly better performances on *App B*, but the difference is extremely small.
An other thing we can observe, is that the performances of (poly and RBF) SVM got much worse with respect to those on the validation set. In particular, poly SVM got the greatest performances decline and is now comparable to RBF SVM, while before it was better.

Next, we are going to check if the hyperparameters selection has been optimal for the different models.

3.3 MVG

Starting from MVG, we are going to check if the Naive assumption is actually a good choice and if the selected number of dimensions for PCA was optimal.

From Figure 21, we can see that for the target application the standard MVG performs slightly better than the Naive MVG model. This means that the Naive hypotesis is probabily causing underfitting.
Furthermore, we can see that the best number of dimensions for PCA is 9 (instead of 8) for both Naive and standard MVG.
For what concerns *App B* instead, the difference between the standard and Naive MVG is less evident, and we can also see that the best value of PCA for

standard MVG is 9 while for Naive it's 8.

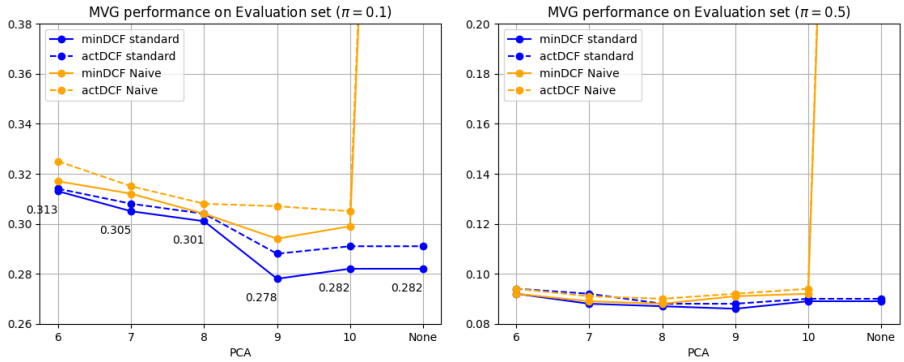


Figure 21: minDCF & actDCF on Evaluation set with MVG

Overall, we can say that our decisions were close to optimal, and in particular they were probabily better for *App B* rather than for *App A*.

3.4 Logistic Regression

Moving on to Logistic Regression, we are going to check if discarding it during the training phase was a good decision or not. Of course we are going to take into account only QLR.
The following graph shows how the minDCF varies when varying λ (we are using PCA(7)).

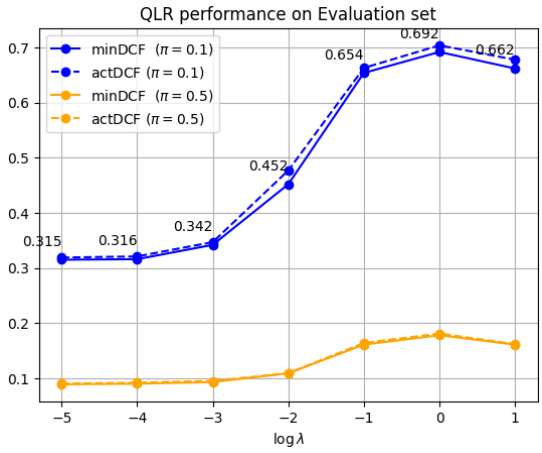


Figure 22: QLR with PCA(7) + z-norm + calibration ($\pi_T = 0.1$)

Actually, we can see that even though the QLR performances were worse than

SVM on the validation set, we are getting better results on the evaluation set (with respect to the selected model, SVM poly d=2 + PCA(8) + z-norm, C=0.01, K=10).
We can also see that $\lambda = 10^{-4}$, which proved to be the optimal value during the training phase, is just a bit worse to the best value ($\lambda = 10^{-5}$), so we found a close to optimal hyperparamter. This holds for both the target application and *App B*.

3.5 SVM

Let's now turn our attention towards SVM models.
We start analyzing poly SVM (PCA(8)+z-norm) with Table 11.

Polynomial SVM			
minDCF			
C	**K**	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.5$
0.001	**0.1**	0.575	0.137
	1	0.551	0.130
	10	0.398	0.101
	100	0.375	0.099
0.01	**0.1**	0.345	0.942
	1	0.338	0.094
	10	0.335	0.096
	100	0.330	0.096
0.1	**0.1**	0.317	0.091
	1	0.321	0.089
	10	0.321	**0.088**
	100	0.422	0.099
1	**0.1**	0.323	0.091
	1	**0.316**	0.089
	10	0.331	0.090
	100	0.788	0.211

Table 11: poly SVM d=2 + PCA(8) + z-norm + calibration ($\pi_T = 0.1$)

In this case, we can see that our hyperparameters selection was not optimal. In fact the best performances are obtained with $(C, K) = (1, 1)$ instead of $(0.01, 10)$. The performances difference is quite high (0.02 less), but the poly SVM model remains worse than the chosen model (Tied GMM K=2).
We can also observe that poly SVM actually has the same performances with respect to QLR, so the models are probabily selecting the same separation surface. This means that the in section 3.4 we thought that QLR was better than SVM despite of a wrong hyperparameters selection, and not because the model actually performs better.

Regarding RBF SVM, instead, we don't report any table because of the large number of hyperparameters. However, from our analysis, we have seen that given PCA(7) and z-norm we obtain a minDCF of 0.334 with $(C, K, \gamma) = (1, 0.1, 0.1)$, for the target application. This means that the hyperparameters selection was accurate for this model. We may find a better minDCF value by doing some refining, as we did for the training phase (where we found $\gamma = 0.16$), but this result is already satisfying.
These hyperparameters are also the best for *App B*.
RBF SVM still remains the worst model among the "best selections".

3.6 GMM

Finally, let's analyze our choices for the GMM model.
We report some graphs for the Tied GMM model and for the full covariance once, without applying PCA.

Regarding the Full Covariance model (Figure 23) we can see that the overall performances are given with one component, which corresponds to applying a MVG model.

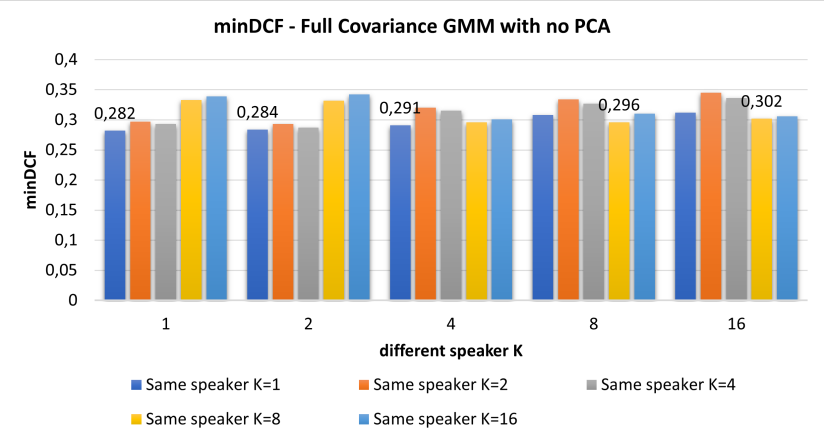


Figure 23: Full Covariance GMM with no PCA

The Tied model (Figure 24) confirms to be the best approach, with a minDCF of 0.282 obtained with different components configurations. This means that our result with K1=K2=2 can be considered optimal. However, the configuration with K1=K2=8 gives lower actual costs (but the difference is really small).

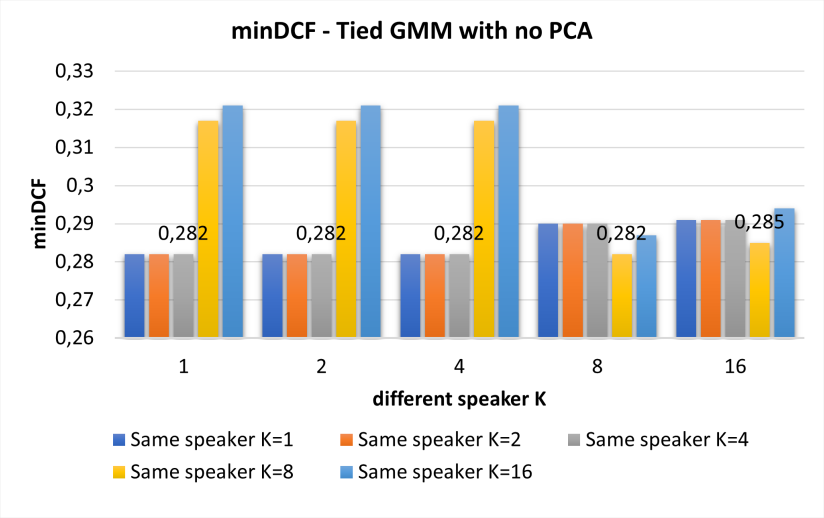


Figure 24: Tied GMM with no PCA

Regarding *App B*, the best results are obtained with the full covariance model with less components (1 to 4), but the model with K1=K2=8 also provides close to optimal minDCF values for *App B*.

Let's also observe that when treating MVG models we have obtained a lower minDCF on the target application (0.278); however the corresponding actDCF is higher than 0.290, while the actDCF of the best GMM model is much more similar to the minDCF.

4 Conclusions

In conclusion, we can say that the choices and strategies used during the training phase have proven to be effective when applied to test data.
Using the Tied Gaussian Mixture Model with 2 components per class, has demonstrated to give close to optimal min and act DCFs on both the target application and on *App B*. On the target application ($\tilde{\pi} = 0.1$) we achieved a pretty good minDCF of 0.282 on the evaluation data. Furthermore, for *App B*, we obtained a minDCF of 0.089, which is again close to optimal.
So, we can say that the choice of delivering the Tied GMM model with 2 components per class and no PCA was good.