Machine Learning and Pattern Recognition
Gender Identification

Marco Cirone
s314878

Paola Matassa
s317632

Contents

1. Application description

The goal of this project is to develop a classifier able to identify the gender from high-level features of face images. The training dataset is composed of 2400 samples and it's highly unbalanced since 70% of samples belong to the female class. Each sample is composed by 12 continuous dimensions, which do not have a physical interpretation. Samples belong to 3 different age groups, each of which may be characterized by a different distribution. However, we don't have access to the age information. The test dataset, on the other hand, has 6000 samples and it's also highly unbalanced, but in this case it's the male class the bigger number of samples.

2. Dataset analysis

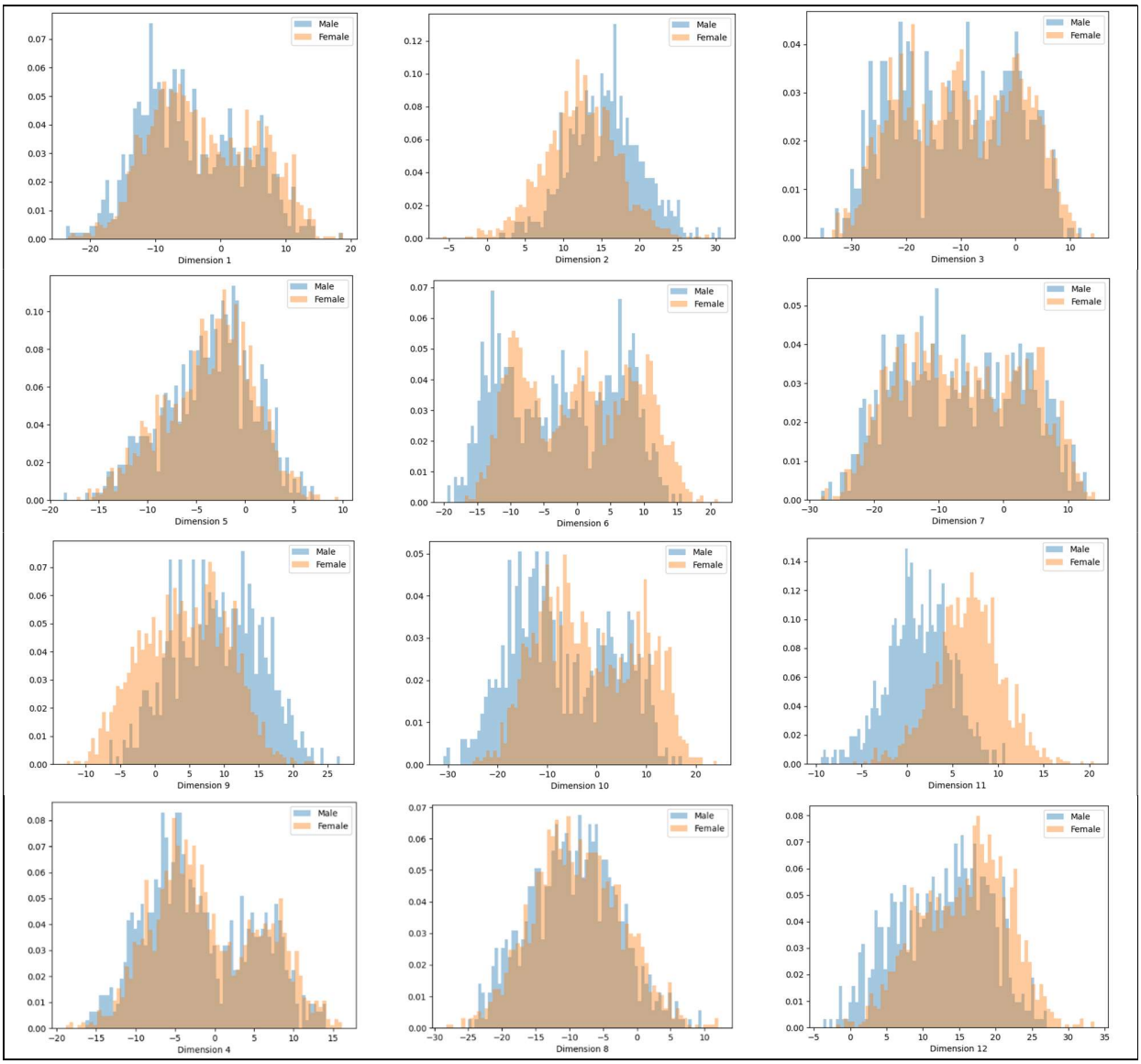


Figure 1: Feature Dataset

Looking at these histograms, we can see how the different dimensions seem to have gaussian distributions or, in the case of dimension 3 for example, they seem to be made of a 3-component gaussian, probably depending on the 3 age groups the dataset is divided into. All the dimensions have a similar distribution for both classes. The only exception is the 11th one, which could act as a discriminant dimension. To see if classes can be linearly separated, we apply the LDA transformation to our dataset. In Figure 2 we can see the separation between the two classes is apparent outside of a small overlapping area.

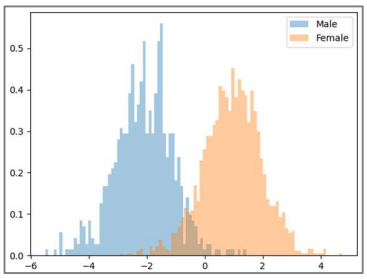


Figure 2: LDA

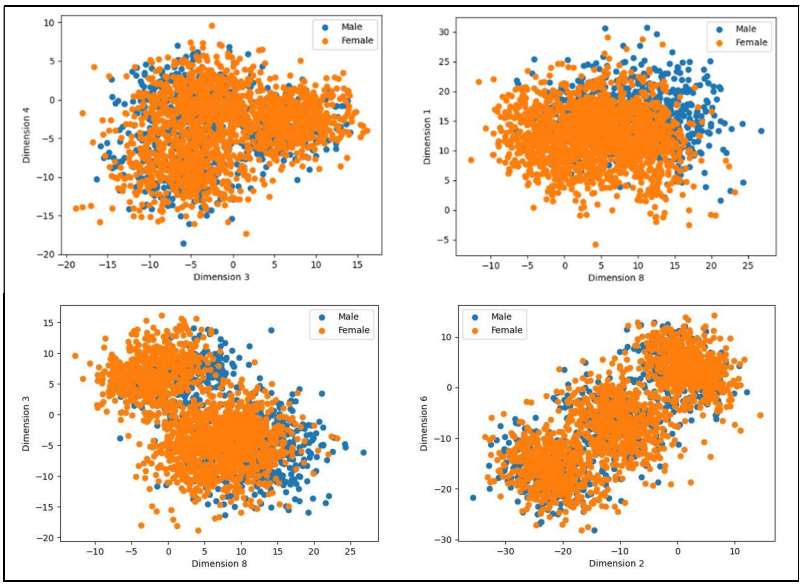


Figure 3: Scatter Plot

These scatter plots also indicate that the distributions are gaussians and they seem to be made of different clusters or components, which again is probably because of the 3 different age groups present in our dataset.

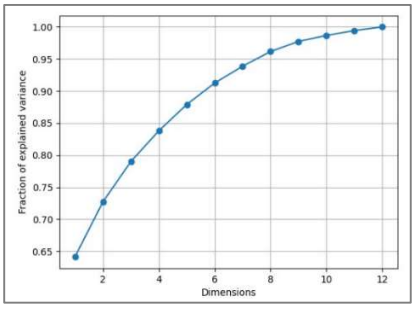


Figure 4: PCA

The picture above shows the amount of variance we lose when reducing the dimensionality through PCA. As we can see, we can afford to remove 1 or 2 dimensions and keep a good amount of information.

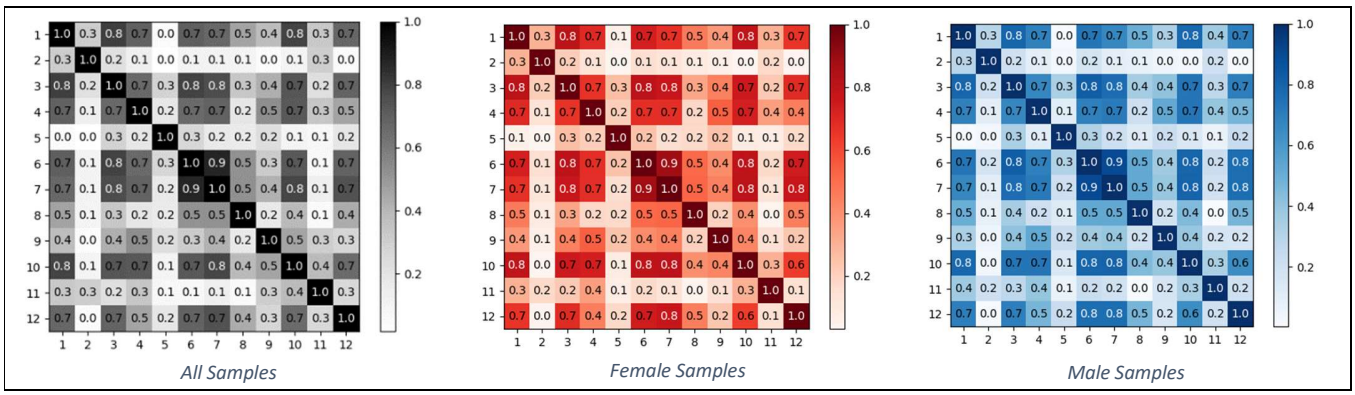


Figure 5: Correlation matrix

Figure 5 shows the different heatmaps. The first one represents the correlation between the various dimensions for the whole dataset. There are dimensions that seem strongly correlated like 1 and 10, and other that seem slightly correlated like 3 and 4. Combining this with the information we got from Figure 4 leads to the conclusion that losing a few dimensions can be beneficial to reduce the number of parameters to estimate the model. Let's now turn our attention to the class specific correlation matrices. They look very similar, which means that the Multivariate Gaussian Model and the Tied Covariance Gaussian Model have similar performance.

3. Validation Method

To estimate the performance of the models, we implemented a K-Fold cross validation approach with K=5. We also applied different preprocessing techniques to our data, such as PCA and Z-score. Our main application is defined by the triplet (π_T = 0.5, C_{fn} = 1, C_{fp} = 1), but we want to test our

models for different applications as well. We decided to use the following 2: (π_T = 0.9, C_{fn} = 1, C_{fp} = 1) and (π_T = 0.1, C_{fn} = 1, C_{fp} = 1). To measure how good a model is, we employ the normalized minimum detection cost metric, which is used to evaluate the performance of the model considering the cost associated with different kinds of errors.

3.1. Gaussian Models

The first class of models we evaluated are the Gaussian models. As previously stated, we also tried different preprocessing techniques to see if the result provided by the model can be better than the ones obtained with raw data. Results of different combinations of preprocessing methodologies are reported below.

	No PCA			PCA 10			PCA 11			PCA 12		
	π= 0.1	***π= 0.5***	***π= 0.9***	***π= 0.1***	***π= 0.5***	***π= 0.9***	***π= 0.1***	***π= 0.5***	***π= 0.9***	***π= 0.1***	***π= 0.5***	***π= 0.9***
MVG	0.308	0.121	0.347	0.423	0.169	0.479	0.309	0.122	0.347	0.308	0.121	0.347
TMVG	0.302	**0.114**	0.336	0.394	0.169	0.47	0.303	0.123	0.351	0.302	**0.114**	0.336
NB	0.784	0.463	0.789	0.434	0.172	0.479	0.314	0.133	0.367	0.323	0.126	0.349
TNB	0.781	0.462	0.8	0.383	0.173	0.465	0.304	0.124	0.369	0.308	0.122	0.352

Table 1: Gaussian models without z-score with and without PCA

Z-Score												
	No PCA			PCA 10			PCA 11			PCA 12		
	π= 0.1	***π= 0.5***	***π= 0.9***	***π= 0.1***	***π= 0.5***	***π= 0.9***	***π= 0.1***	***π= 0.5***	***π= 0.9***	***π= 0.1***	***π= 0.5***	***π= 0.9***
MVG	0.308	0.121	0.347	0.41	0.192	0.538	0.313	0.124	0.35	0.308	0.121	0.347
TMVG	0.302	**0.114**	0.336	0.427	0.186	0.531	0.299	0.121	0.355	0.302	**0.114**	0.336
NB	0.784	0.463	0.789	0.418	0.186	0.543	0.317	0.126	0.336	0.32	0.122	0.343
TNB	0.781	0.462	0.8	0.404	0.182	0.535	0.29	0.125	0.356	0.287	0.119	0.333

Table 2: Gaussian models with z-score with and without PCA

As we can see, results are in line with our initial assumptions. The Naïve Bayes model's performance is poor compared to the other models. The heatmaps were in fact showing that a few dimensions were moderately or strongly correlated. PCA improves the model a lot because it gives orthogonal directions to project our data, and orthogonality is strictly correlated to independence. Also, as we predicted from the class specific correlation matrices, we can notice the similarity in performance between the multivariate models and the tied ones.

3.2. Logistic Regression

We then turn our attention to the Logistic Regression model. This model tries to estimate the posterior distribution by computing the scores in the form:

$$s(x) = w^T x + b$$

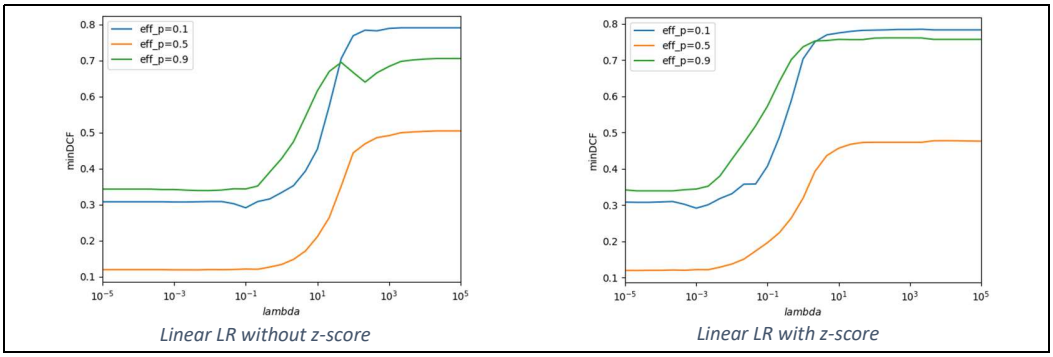
Where w is the vector orthogonal to the hyperplane we defined, and b represents the bias term.
To estimate the correct parameters for the model, we need to minimize the following objective function:

$$J(\boldsymbol{w}, b) = \frac{\lambda}{2}||\boldsymbol{w}||^2 + \frac{\pi_T}{n_T}\sum_{i=1|c_i=1}^{n} \log(1 + e^{-z_i(w^T x_i + b)}) + \frac{1-\pi_T}{n_F}\sum_{i=1|c_i=0}^{n} \log(1 + e^{-z_i(w^T x_i + b)})$$

Where:

- z is the vector representing the class of the samples (z_i=-1 if sample's class is 0, z_i=1 if sample's class is 1),
- n_T is the number of samples belonging to class 1,
- n_F is the number of samples belonging to class 0,
- π_T is the prior we use to generalize the model.

First, we need to choose the hyper-parameters of the model. In this case the only hyper-parameter to set is the regularization term λ. To estimate it, we resort to cross-validation.



Linear LR without z-score *Linear LR with z-score*

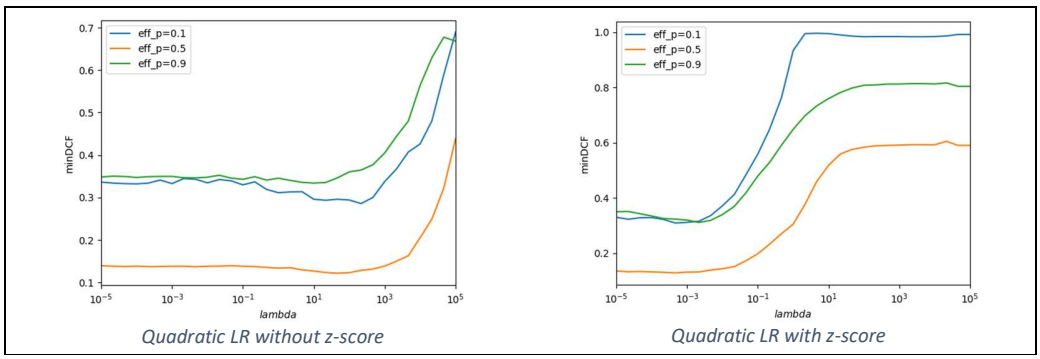
Figure 6: Linear LR

As we can see from Figure 6, the regularization term that gives us the minimum DCF is 10^{-5}, so we are going to use that value while we train the model. The tables below report the minimum DCF for different applications and different values of π_T.

	No Z-Score			Z-Score		
	π= 0.1	***π= 0.5***	***π= 0.9***	***π= 0.1***	***π= 0.5***	***π= 0.9***
LR(π_T=0.1)	0.290	0.121	0.370	0.291	0.121	0.371
LR(π_T=0.5)	0.308	**0.119**	0.343	0.308	**0.119**	0.341
LR(π_T=0.9)	0.338	0.116	0.321	0.337	0.118	0.322

Table 3: Linear Logistic Regression with $\lambda=10^{-5}$ with and without z-score

As we can see, the difference between presence and absence of z-score is hardly noticeable.
Now we focus on the Quadratic Logistic Regression Model. Just like for the previous model, we need to estimate the regularization term through cross-validation.



Quadratic LR without z-score *Quadratic LR with z-score*

Figure 7: Quadratic LR

This time we can see z-score producing a difference. For the raw dataset we are considering the regularization term equal to 10^2, but we will choose the value 10^{-5} if we are using the z-score. This difference is given by the fact that the Quadratic Logistic Regression performs a feature expansion of the dataset, making the model not invariant to data transformation.

	No Z-Score ($\lambda=10^2$)			Z-Score ($\lambda=10^{-5}$)		
	π= 0.1	***π= 0.5***	***π= 0.9***	***π= 0.1***	***π= 0.5***	***π= 0.9***
QLR(π_T=0.1)	0.282	0.124	0.407	0.351	0.140	0.368
QLR(π_T=0.5)	0.294	**0.123**	0.360	0.330	**0.136**	0.350
QLR(π_T=0.9)	0.382	0.130	0.362	0.347	0.144	0.367

Table 4: Quadratic Logistic Regression with and without z-score

In this case we can notice how z-score is making the performance of the model worse, except for when we set the prior to 0.9. Like we anticipated, the linear version of the Logistic Regression model seems to have a better performance. Finally, we decide to keep the best performing model, which in this case is the linear one with π_T=0.9.

3.3. Support Vector Machine

Support Vector Machines are linear classifiers that look for the maximum margin separation hyperplane. The (primal) SVM objective consists in minimizing:

$$J(\boldsymbol{w}, b) = \frac{1}{2}||\boldsymbol{w}||^2 + \sum_{i=1}^{n} C_i \max(0, 1 - z_i(w^T x_i + b))$$

to solve the SVM problem we can maximize the dual formulation:

$$J_D(\alpha) = -\frac{1}{2}\alpha^T \boldsymbol{H}\alpha + \alpha^T \mathbf{1} \qquad s.t. 0 \le \alpha_i \le C_i, \forall i \in \{1 \dots n\}$$

Where:

- $C_i = \begin{cases} C\frac{\pi_T}{\pi_T^{emp}} & if\ x_i\ belong\ to\ class\ 1 \\ C\frac{\pi_F}{\pi_F^{emp}} & if\ x_i\ belong\ to\ class\ 0 \end{cases}$
- $z_i = \begin{cases} +1 & if\ x_i\ belong\ to\ class\ 1 \\ -1 & if\ x_i\ belong\ to\ class\ 0 \end{cases}$
- H: matrix whose elements are $H_{ij} = z_i z_j x_i^T x_j$ for the linear problem.

Now we try to find the hyper-parameters. For the linear problem we consider C and K values. First, we confront 5 k values (0.01, 0.1, 10, 100)

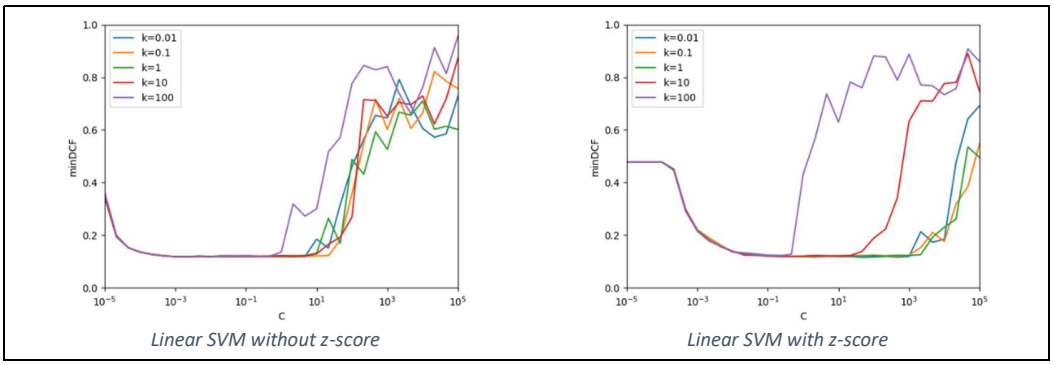


Figure 8: Linear SVM considering K= 0.01, 0.1, 1, 10, 100

In the charts presented earlier, we observe that in the model without z-score normalization, the min DCF consistently exhibits the same lower value across all values of K, except for when K equals 100. In contrast, in the model with z-score normalization, this pattern remains consistent, except for K values of 10 and 100. Consequently, it is reasonable to consider K=1 for both models. Now we consider all possible C values across the range of [10^{-5}, 10^{5}].

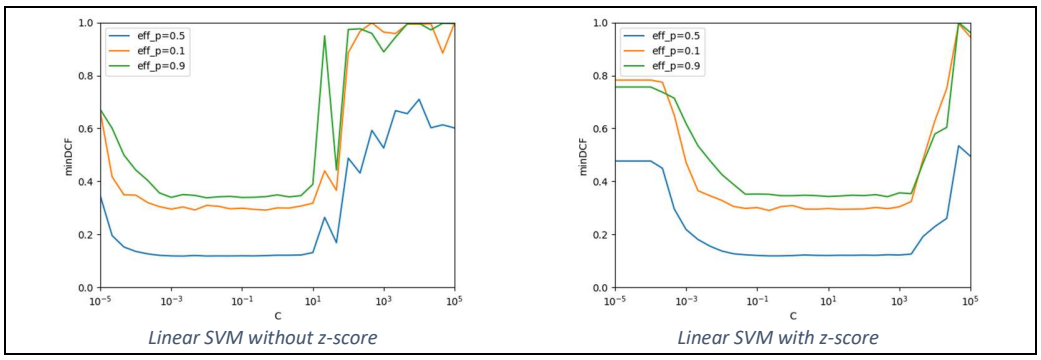


Figure 9: Linear SVM considering K= 1

As we can see, in this case we can choose C=1 for both models to obtain the best min DCF values. In the Table 5 we have the results obtained with our hyper-parameters.

	No Z-Score			**Z-Score**		
	π= 0.1	***π= 0.5***	***π= 0.9***	***π= 0.1***	***π= 0.5***	***π= 0.9***
Linear SVM(π_T=0.1)	0.293	0.125	0.365	0.31	0.127	0.374
Linear SVM(π_T=0.5)	0.3	0.121	0.349	0.309	0.12	0.346
Linear SVM(π_T=0.9)	0.347	**0.118**	0.347	0.31	0.127	0.374

Table 5: Linear SVM with K=1 and C=1

Now we analyze the non-linear problem. The dual formulation discussed before, allows for non-linear separation through the kernel trick. In this case we have $H_{ij} = z_i z_j k(x_i, x_j)$ where $k(x_i, x_j)$ is the kernel function. We consider two types of kernels. The first type is the polynomial kernel with $k(x_i, x_j) = (x_i^T x_j + c)^d$. In this case we must find the best values for constant c, C and K. First, we confront 3 K values (0.1, 1, 10) across all c and C values.

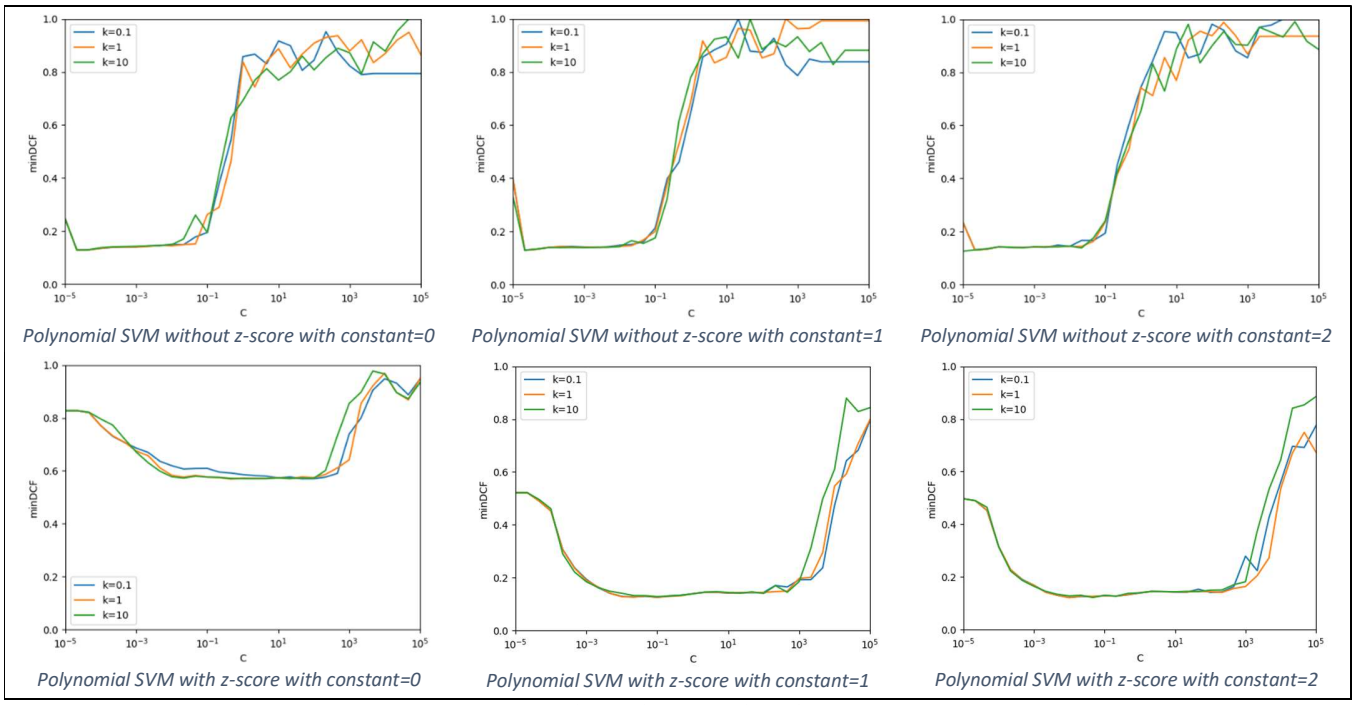


Figure 10: Polynomial SVM considering k=0.1, 1, 10

As we can see in the Figure 10, the graphs have a similar behavior for different K values. So, we can choose K=1 again. Now we analyze the 3 different values for constant c (0, 1, 2).

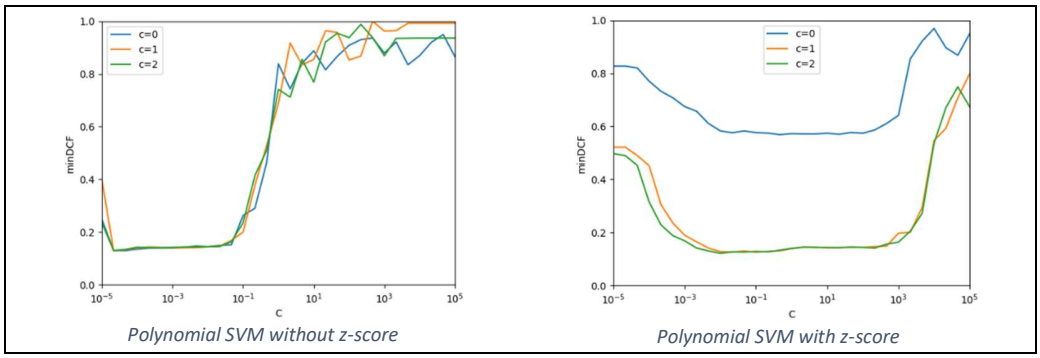


Figure 11: Polynomial SVM considering constant=0, 1, 2 and k=1

For model without z-score we have a similar behavior for all c specially for low min DCF values. In the case with z-score we have good min DCF values only for c=1 and c=2. We can choose c=1 for both models. Now we consider all C values across the range of $[10^{-5}, 10^{5}]$.

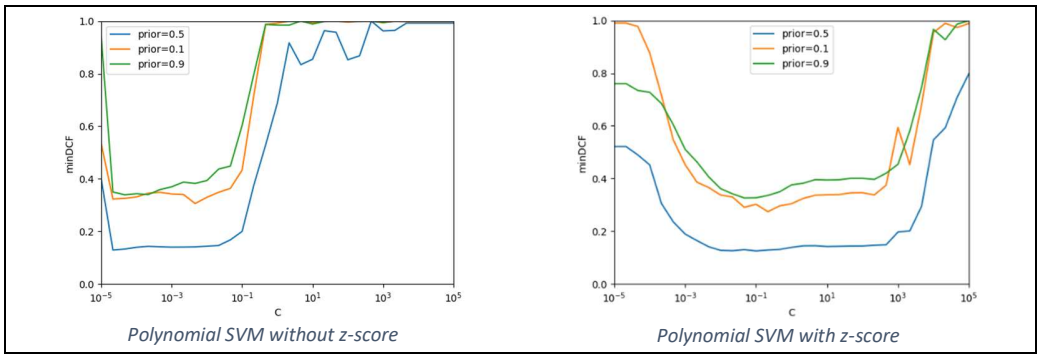


Figure 12: Polynomial SVM considering constant=1 and k=1

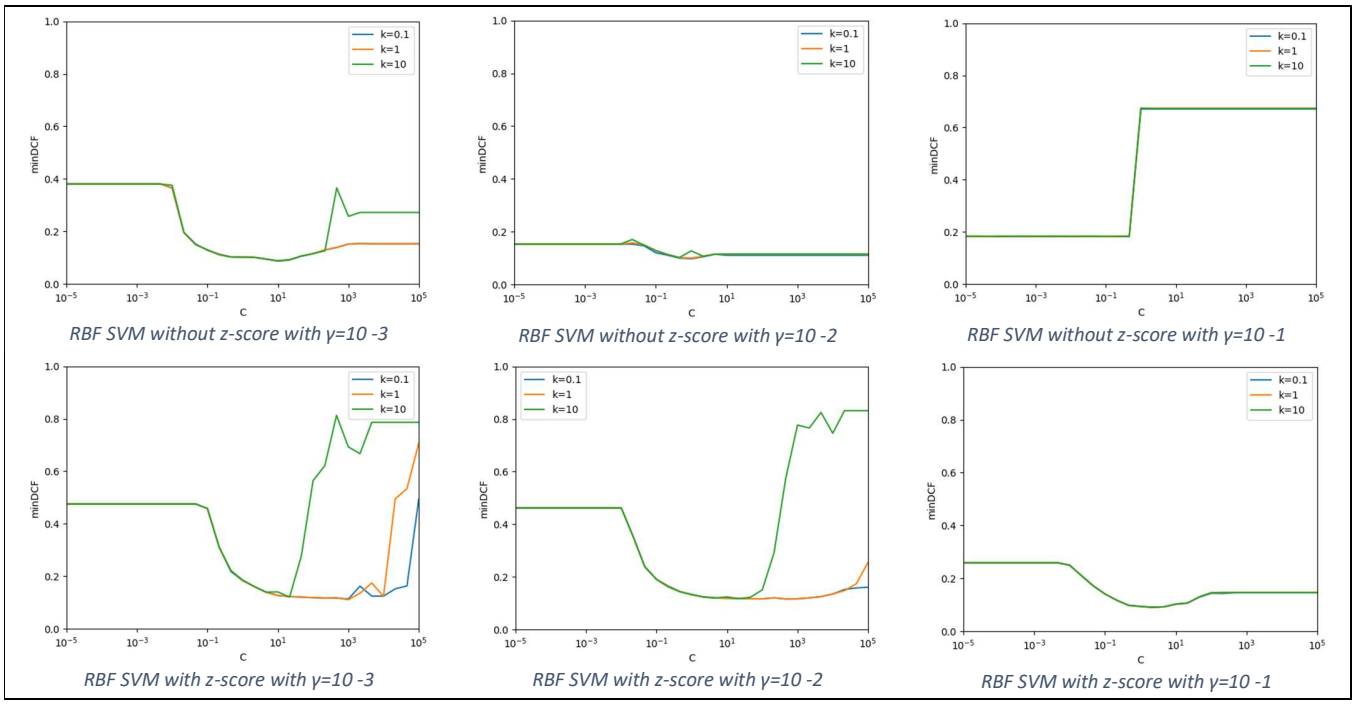
In Figure 12 we can see that we have the best values for min DCF considering $C=10^{-3}$ for the model without z-score and $C=10^{-1}$ for the model with z-score. In the table below we have the results considering all the choices done before.

	No Z-Score (C=10^{-3})			Z-Score (C=10^{-1})		
	π= 0.1	***π= 0.5***	***π= 0.9***	***π= 0.1***	***π= 0.5***	***π= 0.9***
Polynomial SVM(π_T=0.1)	0.378	0.158	0.448	0.33	0.148	0.44
Polynomial SVM(π_T=0.5)	0.342	0.140	0.37	0.302	0.126	0.327
Polynomial SVM(π_T=0.9)	0.369	0.153	0.349	0.382	0.135	0.298

Table 6: Polynomial SVM with K=1, constant=1 and d=2

Considering the results obtained in Table 6, we can see that the SVM with the polynomial kernel perform worse than the linear SVM.

The second type of kernel we want to analyze is the RBF kernel where $k(x_i, x_j) = e^{-\gamma\|x_i - x_j\|^2}$, so, we must find the best values for K, γ and C. We start confronting again 3 k values (0.1, 1, 10).

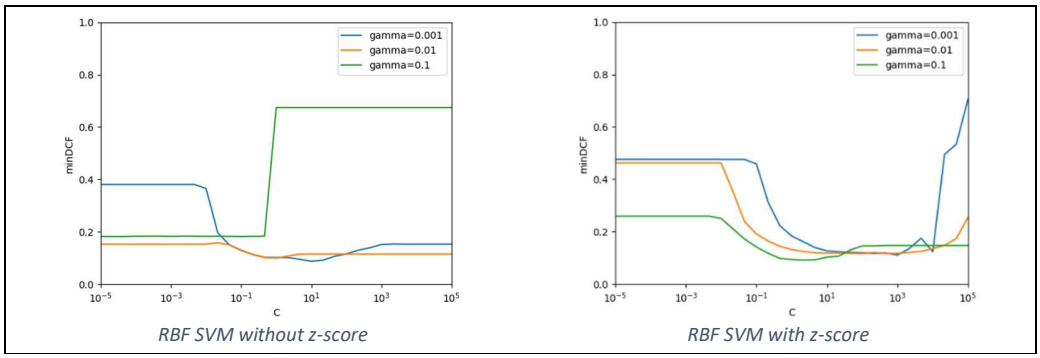


RBF SVM without z-score with γ=10 -3 | *RBF SVM without z-score with γ=10 -2* | *RBF SVM without z-score with γ=10 -1*

RBF SVM with z-score with γ=10 -3 | *RBF SVM with z-score with γ=10 -2* | *RBF SVM with z-score with γ=10 -1*

Figure 13: RBF SVM considering k=0.1, 1, 10

In Figure 13 we can see that we have a similar behavior for almost all the graphs. In few cases where the charts differ for the k values, the worst value is K=10. We can choose again K=1. Now we analyze 3 different values of γ (10^{-1}, 10^{-2}, 10^{-3}).



RBF SVM without z-score | *RBF SVM with z-score*

Figure 14: RBF SVM considering γ=10^{-1}, 10^{-2}, 10^{-3} and k=1

In Figure 14, it is evident that the min DCF value is lowest for γ=10^{-3} in the model without z-score normalization, and for γ=10^{-1} in the model with z-score normalization. Towards the end, we examine the same range of C values as in the previous analyses.

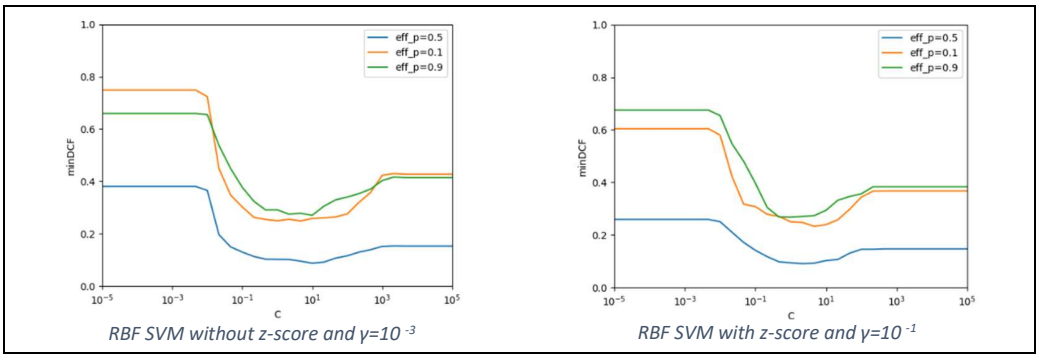


Figure 15: RBF SVM considering all prior and k=1

As we can see in Figure 15 the best values are C=10 for SVM without z-score and C=1 for SVM with z-score.

	No Z-Score (C=10, $\gamma=10^{-3}$)			**Z-Score (C=1, $\gamma=10^{-1}$)**		
	π= 0.1	***π= 0.5***	***π= 0.9***	***π= 0.1***	***π= 0.5***	***π= 0.9***
RBF SVM(π_T=0.1)	0.244	0.107	0.374	0.236	0.12	0.379
RBF SVM(π_T=0.5)	0.258	**0.087**	0.27	0.25	0.094	0.268
RBF SVM(π_T=0.9)	0.34	0.104	0.257	0.325	0.109	0.272

Table 7: RBF SVM with K=1

In Table 7 we have the results obtained considering our choices. We can see that in this case the models perform significantly better than the linear SVM.
We decide to keep the linear model with K=1, C=1 and π_T=0.9 and the RBF SVM with K=1, C=10, $\gamma=10^{-3}$ and π_T=0.5, both without z-score.

3.4. Gaussian Mixture Models

Last model we are going to analyze is the Gaussian Mixture Model. Considering what we saw in the scatter plots and on the histograms about the dataset, we expect this model to have a better performance with a number of components between 2 and 4 (remembering that the dataset is composed of 3 age groups which we can't access). Also, for the same reasons already mentioned while analyzing the Gaussian Model, we expect the normal GMM model and the Tied Covariance GMM Model to have similar performances.

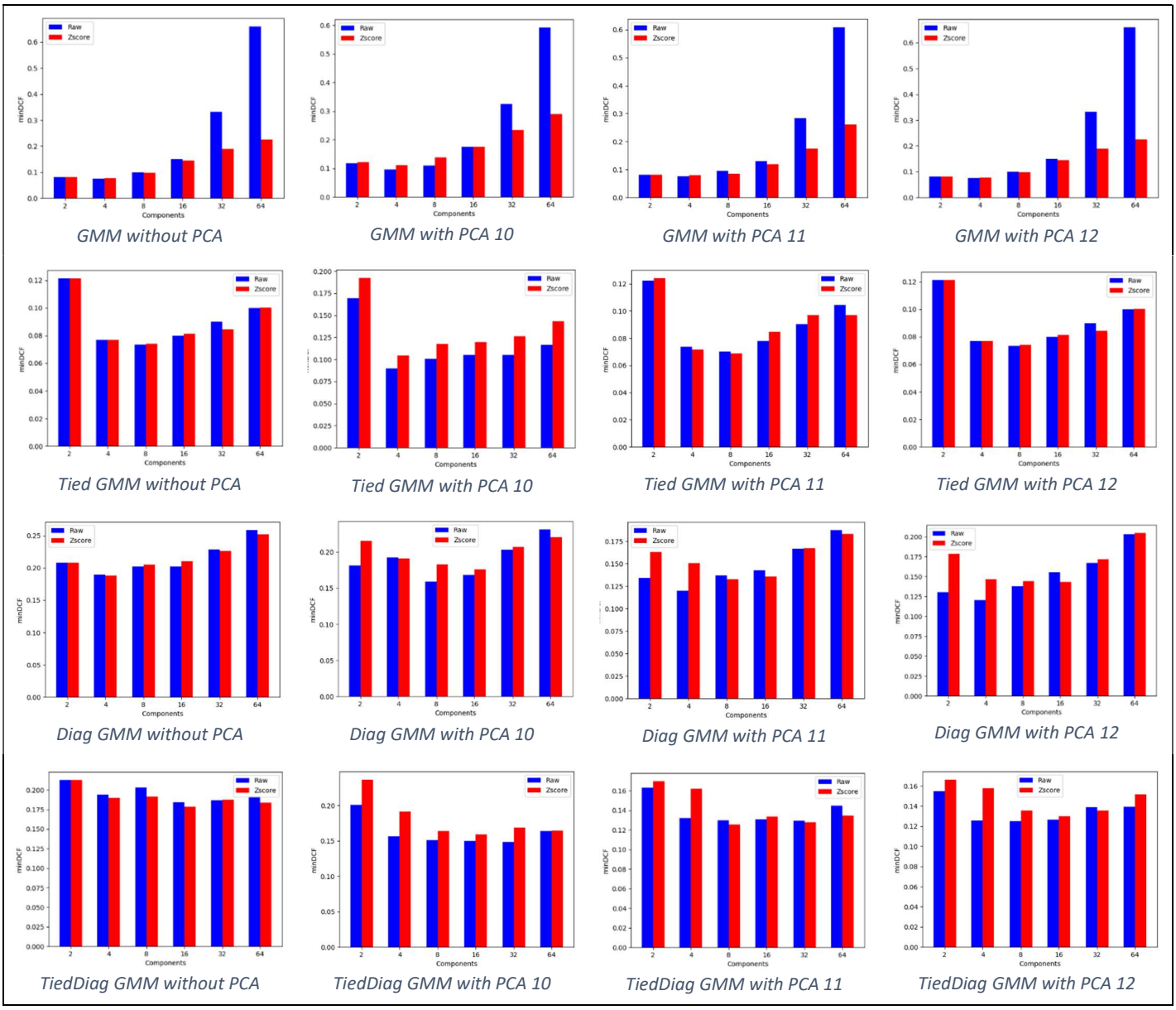


Figure 16: GMM models with and without z-score

Figure 16 shows the DCF for different types of GMM with different kinds of preprocessing methodologies. For the normal GMM model, the optimal number of components seems to be 4. For the Tied Covariance GMM, the optimal number of components is 8. For the Diagonal GMM it's 4. For the Diagonal Tied Covariance, it's 16.

	No PCA			PCA 10			PCA 11			PCA 12		
	π=0.1	***π=0.5***	***π=0.9***	***π=0.1***	***π=0.5***	***π=0.9***	***π=0.1***	***π=0.5***	***π=0.9***	***π=0.1***	***π=0.5***	***π=0.9***
GMM (4)	0.229	0.076	0.205	0.278	0.097	0.246	**0.206**	**0.076**	**0.208**	0.228	0.076	0.205
Tied GMM (8)	0.249	0.073	0.193	0.283	0.101	0.243	**0.226**	**0.070**	**0.198**	0.249	0.073	0.193
Diag GMM (4)	0.460	0.189	0.482	0.465	0.193	0.422	0.302	0.120	0.309	0.321	0.120	0.321
TiedDiag GMM (16)	0.486	0.184	0.470	0.427	0.150	0.411	0.371	0.131	0.349	0.379	0.126	0.325

Table 8: GMM models without z-score with and without PCA

Z-Score												
	No PCA			PCA 10			PCA 11			PCA 12		
	π=0.1	***π=0.5***	***π=0.9***	***π=0.1***	***π=0.5***	***π=0.9***	***π=0.1***	***π=0.5***	***π=0.9***	***π=0.1***	***π=0.5***	***π=0.9***
GMM (4)	0.229	0.076	0.205	0.301	0.11	0.252	0.208	0.079	0.208	0.229	0.076	0.205
Tied GMM (8)	0.234	0.074	0.185	0.301	0.117	0.288	**0.211**	**0.069**	**0.177**	0.234	0.074	0.185
Diag GMM (4)	0.459	0.188	0.481	0.449	0.191	0.490	0.370	0.151	0.349	0.372	0.147	0.346
TiedDiag GMM (16)	0.525	0.179	0.472	0.413	0.159	0.449	0.382	0.133	0.344	0.426	0.130	0.355

Table 9: GMM models with z-score with and without PCA

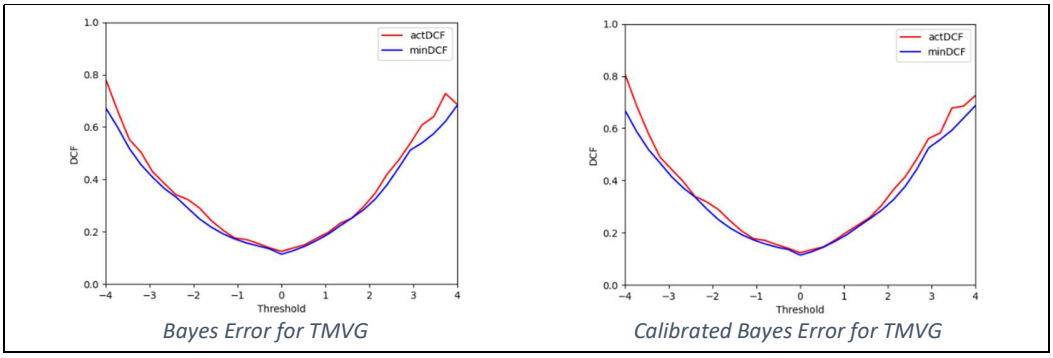
As we can see, the best model seems to be the Tied GMM with 8 components in combination with Z-score and PCA (11). But we were anticipating the optimal number of components to be 2 or 4. Let's in fact remember that both the training and test dataset are composed of 3 age groups. Thus we decide to bring to the evaluation phase a second GMM model, corresponding to the regular GMM with 4 components and PCA equal to 11, expecting a difference in performance when testing the model on the evaluation set.

Best models so far

TMVG no PCA, no Z-score
LR $\lambda=10^{-5}$, π_T: 0.5
Linear SVM, π_T: 0.9, no Z-score, K=1, C=1
RBF SVM, π_T: 0.5, no Z-score, C: 10, γ: 10^{-3}, K: 1
GMM (4), PCA: 11,
Tied GMM (8), Z-score, PCA (11)

4. Calibration

Now we know the best models. The problem is that we only looked for the ones that get the minimum DCF, but it's not guaranteed that their scores are good for the application we are trying to optimize. To overcome this issue, we can calibrate the scores of these models. In this case, we chose to use the prior weighted Logistic Regression in combination with a K-fold approach. We use the training scores of the model we want to calibrate as training data and we train a prior weighted Logistic Regression that will give us the calibrated scores at the end. To understand if the model needs to have its scores calibrated or not, we use the Bayes Error Plots to understand the difference between the actual DCF and the minimum DCF given different thresholds. We start with the Tied Gaussian model.



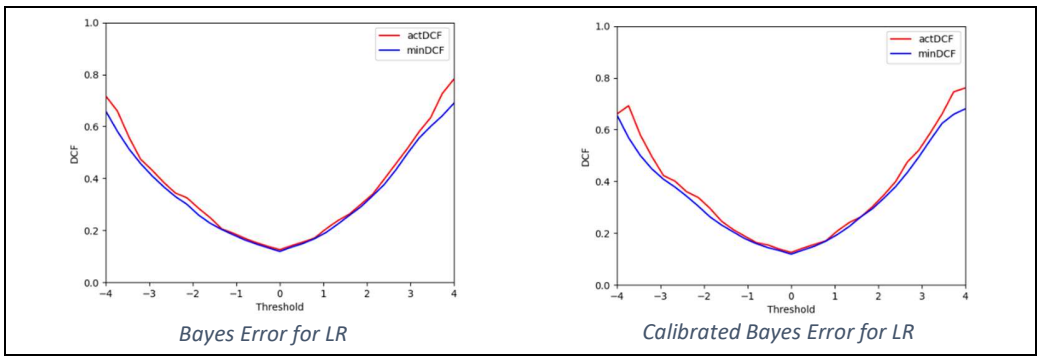
Bayes Error for TMVG *Calibrated Bayes Error for TMVG*

Figure 17:Bayes Error Plot for TMVG with and without calibration

Actual DCF	***π= 0.1***	***π= 0.5***	***π= 0.9***
Uncalibrated	0.324	0.125	0.360
Calibrated	0.332	0.123	0.380

Table 10: Actual DCF for TMVG

As we can see, calibration doesn't provide major improvements, especially for the application we are interested in. Thus, we decide to not calibrate the scores for this model.
Let's now analyze the Logistic Regression model.



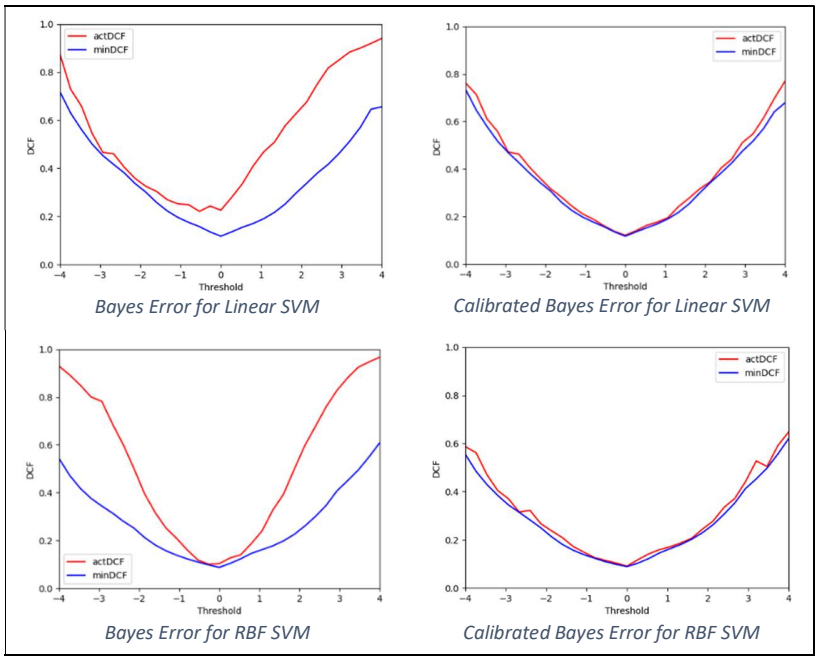
Bayes Error for LR *Calibrated Bayes Error for LR*

Figure 19: Bayes Error Plot for Logistic Regression with and without Calibration

Actual DCF	$\pi = 0.1$	$\pi = 0.5$	$\pi = 0.9$
Uncalibrated	0.341	0.126	0.356
Calibrated	0.340	0.126	0.360

Table 11: Actual DCF for Logistic Regression

Just like the gaussian model, there are no major improvements, so calibration is not needed.
We know analyze the SVM models.



Bayes Error for Linear SVM *Calibrated Bayes Error for Linear SVM*

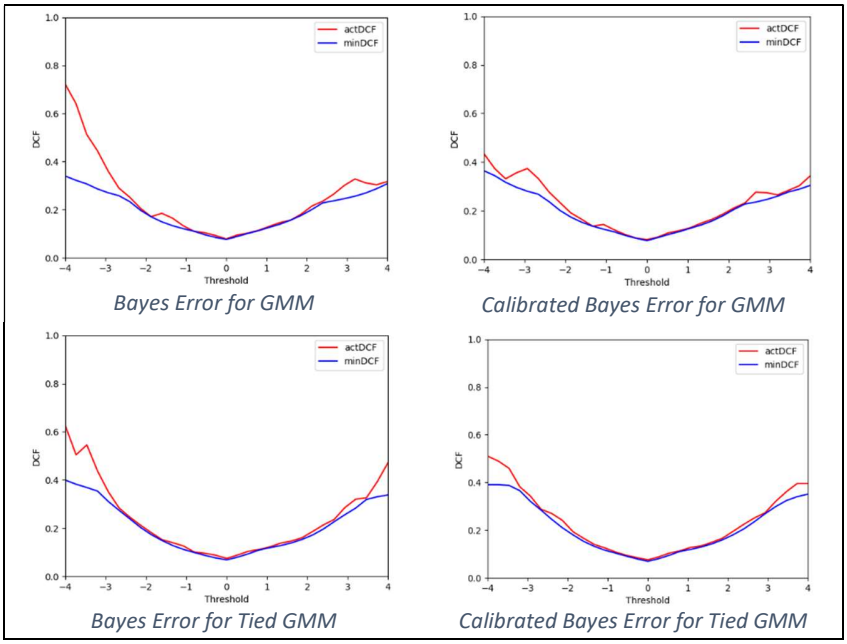
Bayes Error for RBF SVM *Calibrated Bayes Error for RBF SVM*

Figure 18: Bayes Error Plot for Linear SVM and RBF SVM with and without calibration

	Linear SVM			RBF SVM		
Actual DCF	$\pi = 0.1$	$\pi = 0.5$	$\pi = 0.9$	$\pi = 0.1$	$\pi = 0.5$	$\pi = 0.9$
Uncalibrated	0.362	0.226	0.689	0.525	0.103	0.615
Calibrated	0.366	0.121	0.364	0.279	0.091	0.290

Table 12: Actual DCF for Linear SVM and RBF SVM

We can see how the calibration provides a very big improvement for the linear SVM model, which is not even good for the application we want to optimize without having its scores calibrated. This was expected because as we know, SVM scores follow a geometric interpretation rather than a probabilistic one, and calibration is often used in combination with this kind of model to resolve this problem. The RBF SVM instead seems good for our target application but performs poorly for other applications. Calibration gives a significant improvement, so we decide to calibrate these scores as well. Now, we analyze the GMM models.



Bayes Error for GMM *Calibrated Bayes Error for GMM*

Bayes Error for Tied GMM *Calibrated Bayes Error for Tied GMM*

Figure 19: Bayes Error Plot for GMM and Tied GMM with and without Calibration

	GMM			Tied GMM		
	π= 0.1	**π= 0.5**	**π= 0.9**	**π= 0.1**	**π= 0.5**	**π= 0.9**
Uncalibrated	0.215	0.079	0.231	0.222	0.075	0.191
Calibrated	0.248	0.081	0.222	0.253	0.075	0.197

Table 13: Actual DCF for GMM and Tied GMM

Regular GMM model improves in a moderate way after calibration but it's for applications that are very different from the ones we tested, so we decided to not use the calibration. The Tied GMM model gets some improvements but also in this case it's for applications we are not really interested in and the improvements are not very significant anyway, so we decide to not calibrate this model as well.

To conclude, our best models at the end of the Validation phase are:

- TMVG no PCA, no Z-score
- LR $\lambda=10^{-5}$, π_T: 0.5
- Linear SVM, π_T: 0.9, no Z-score, K=1, C=1, with calibration
- RBF SVM, π_T: 0.5, no Z-score, C: 10, γ: 10^{-3}, K: 1, with calibration
- GMM (4), PCA: 11
- Tied GMM (8), Z-score, PCA (11)

In the image below there are the ROC curve and DET for the best models that achieve the best results.

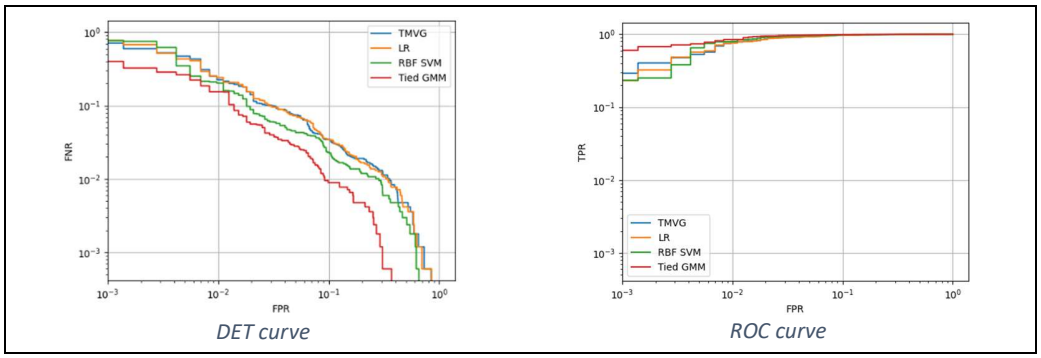


Figure 10: DET curve and ROC curve for Tied MVG, Logistic Regression, RBF SVM and Tied GMM

5. Fusion

Now we want to know if combining the scores of different models can give us better results. To merge two or more models into a single one we apply the same method we used for the calibration, only that this time the model will be trained on the training scores obtained from all the models that we want to merge. The table below is showing the minDCF of various combinations of models.

	π= 0.1	***π= 0.5***	***π= 0.9***
Tied GMM + RBF SVM + LR	**0.204**	**0.070**	**0.177**
GMM + RBF SVM + LR	0.211	0.076	0.196
Tied GMM + Linear SVM + LR	0.208	0.070	0.182
GMM + Linear SVM + LR	0.212	0.076	0.197
Linear SVM + LR	0.315	0.121	0.341
RBF_SVM + LR	0.236	0.087	0.281
Tied GMM + Linear SVM	**0.212**	**0.069**	**0.178**
Tied GMM + RBF SVM	0.212	0.072	0.183
GMM + Linear SVM	0.213	0.076	0.192
GMM + RBF SVM	0.203	0.074	0.207

Table 14: MinDCF values for the models obtained with the fusion

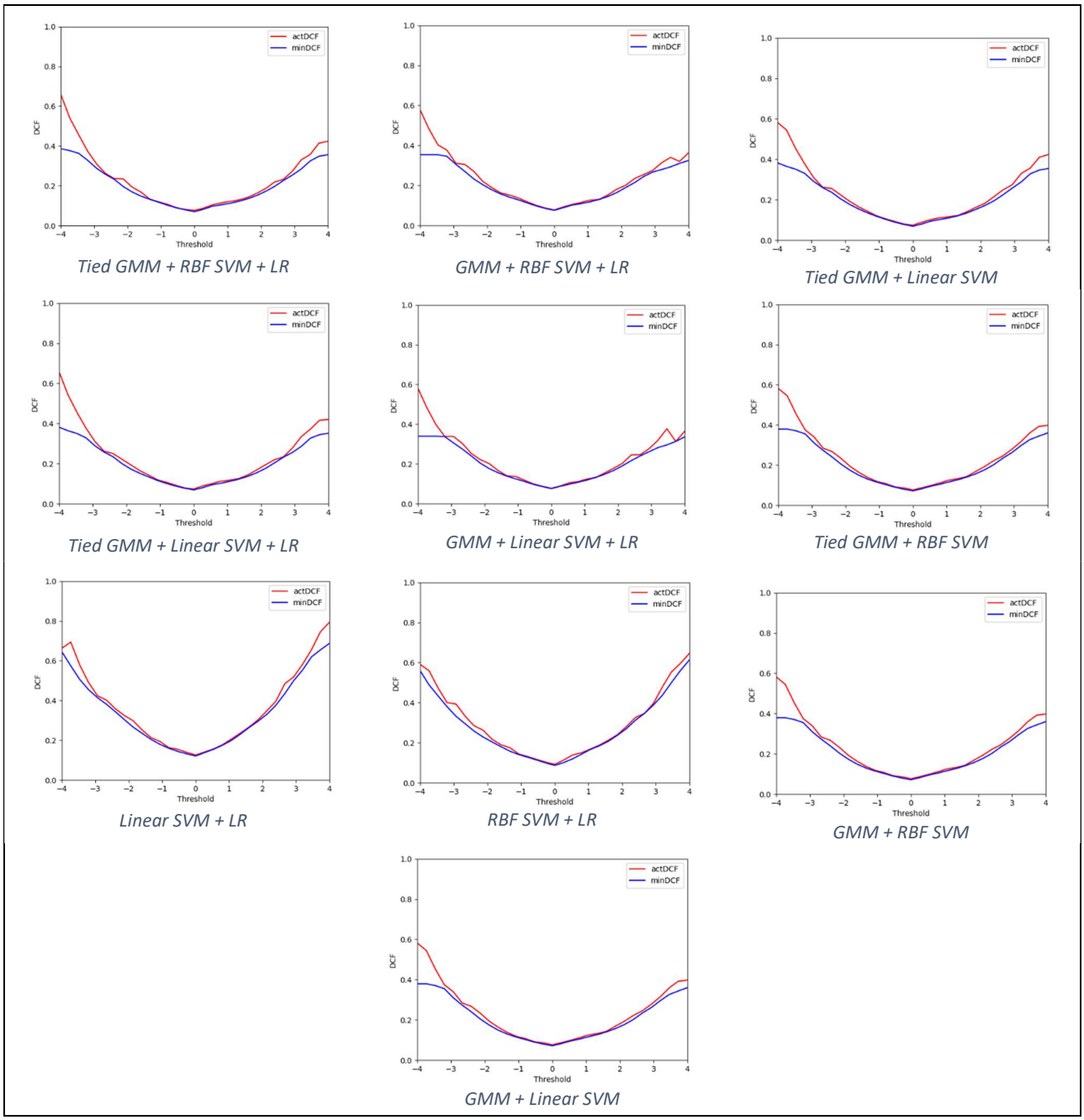


Figure 11: Bayes Error Plot for the models obtained with the fusion

We can see how both the GMM models, and the RBF SVM model gives a significant boost to the other models. But none of the following fusions seems to surpass the Tied GMM model in performance. Notice how fusing Linear SVM and LR provides us a worse result than the 2 models alone can give. In the pictures above the bayes error plots for all the fusion models are represented.
All the fusion scores are well calibrated already, so calibration is not required.
So, now we move to the evaluation part, where we will see if the chosen made so far are correct.

6. Evaluation

We now analyze the behavior of the models which obtained the best results in the previous phase on the test dataset. The table below is showing us both the minimum DCF and actual DCF for the same priors we used for the training phase. Remember that LR corresponds to LR($\lambda=10^{-5}$, π_T: 0.5), Linear SVM to Linear SVM(π_T: 0.9, no Z-score, K=1, C=1), RBF SVM to RBF SVM(π_T=0.5, C=10, $\gamma=10^{-3}$, K=1), GMM to GMM(PCA=11, components=4) and Tied GMM to Tied GMM(PCA=11, components=8, Z-score). We begin by analyzing the performance of the models we chose by looking at their minimum and actual DCF on the test dataset and on their Bayes Error plots to see if they require calibration or not. Let's remember that during the validation phase only the SVM models needed it.

	minDCF			actDCF		
	π= 0.1	***π= 0.5***	***π= 0.9***	***π= 0.1***	***π= 0.5***	***π= 0.9***
TMVG	0.301	0.116	0.308	0.309	0.117	0.323
LR	0.301	0.121	0.305	0.308	0.121	0.319
Linear SVM	0.360	0.132	0.285	0.370	0.211	0.691
RBF SVM	0.221	0.081	0.215	0.464	0.086	0.540
GMM	0.146	0.062	0.144	0.152	0.063	0.159
Tied GMM	0.174	0.066	0.150	0.175	0.070	0.157

Table 15: minDCF and Actual DCF values for the best models

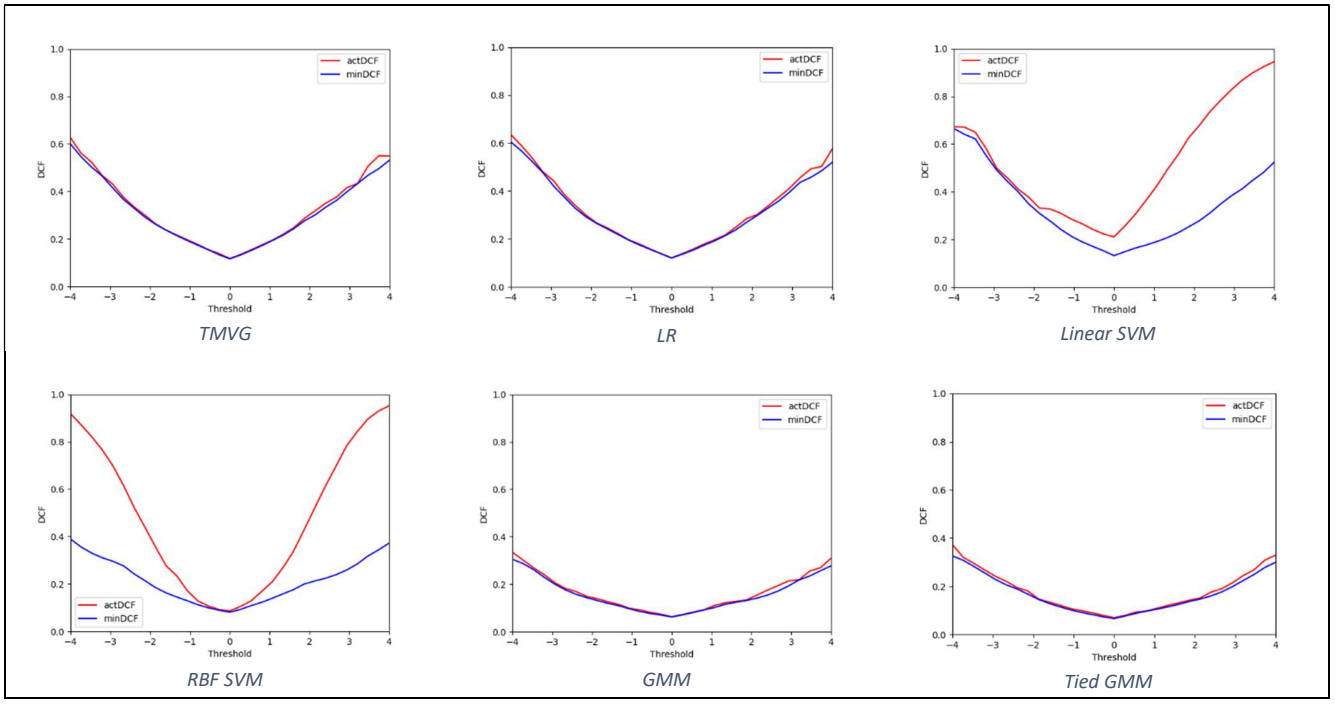


Figure 12: Bayes Error Plot with Evaluation dataset for Logistic Regression, Tied MVG, Linear SVM, RBF SVM, GMM, Tied GMM

As we can see from the differences between actual and minimum DCF and from the bayes error plots, both of the Support Vector Machine models need to be calibrated in this case just like they did during the training phase. The pictures below are showing their Bayes Error plots after the calibration.

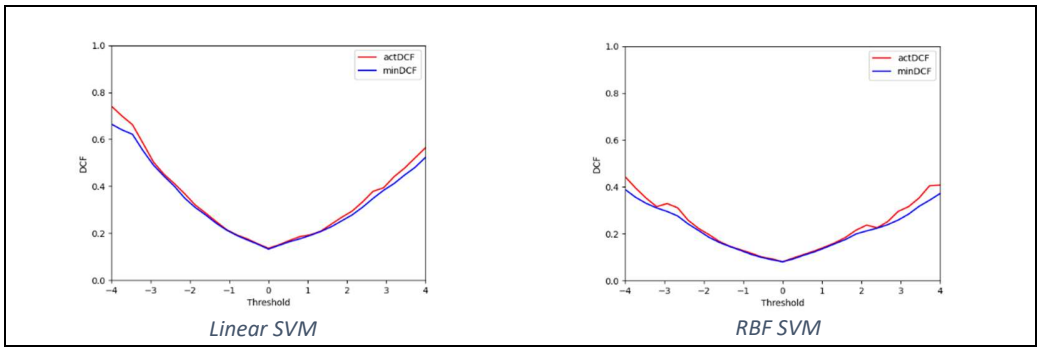


Figure 24: Bayes Error plot after calibration for Linear SVM and RBF SVM

The pictures below are instead showing the DET and ROC curve for our models (with calibration if needed).

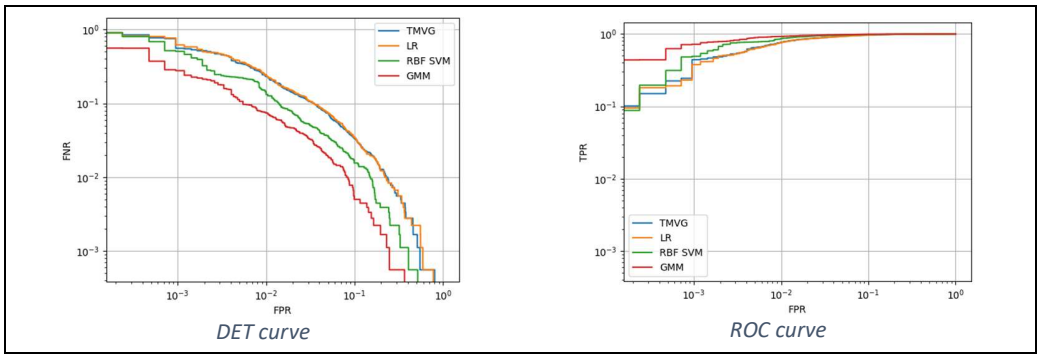


Figure 25 13: DET curve and ROC curve for Tied MVG, Logistic Regression, RBF SVM and GMM

Now we analyze the fusion models we previously trained. Just like before, we are going to show both the minimum DCF and the actual DCF.

	minDCF			***actDCF***		
	π= 0.1	***π= 0.5***	***π= 0.9***	***π= 0.1***	***π= 0.5***	***π= 0.9***
Tied GMM + RBF SVM + LR	0.167	0.063	0.154	0.182	0.064	0.165
GMM + RBF SVM + LR	**0.143**	**0.061**	**0.144**	0.157	0.063	0.162
Tied GMM + Linear SVM + LR	0.167	0.064	0.156	0.183	0.065	0.161
GMM + Linear SVM + LR	0.142	0.062	0.149	0.158	0.063	0.159
Linear SVM + LR	0.298	0.12	0.306	0.315	0.120	0.123
RBF_SVM + LR	0.228	0.083	0.225	0.224	0.084	0.229
Tied GMM + Linear SVM	0.169	0.067	0.155	0.185	0.068	0.161
Tied GMM + RBF SVM	0.171	0.066	0.148	0.183	0.067	0.172
GMM + Linear SVM	0.143	0.061	0.145	0.158	0.063	0.159
GMM + RBF SVM	0.145	0.062	0.144	**0.161**	**0.062**	**0.166**

Table 16: Min DCF and Actual DCF for model obtained with the fusion

We can notice that GMM and SVM models in general tend to improve the performance of the other models.

The best results are obtained by fusing the GMM model with RBF SVM and Linear Logistic Regression. Looking at Table 16 and also at figure 26, which is showing the Bayes Error plots for all the fusion models we tried, we can notice that all these models don't require calibration just like they didn't need it on the Validation set.

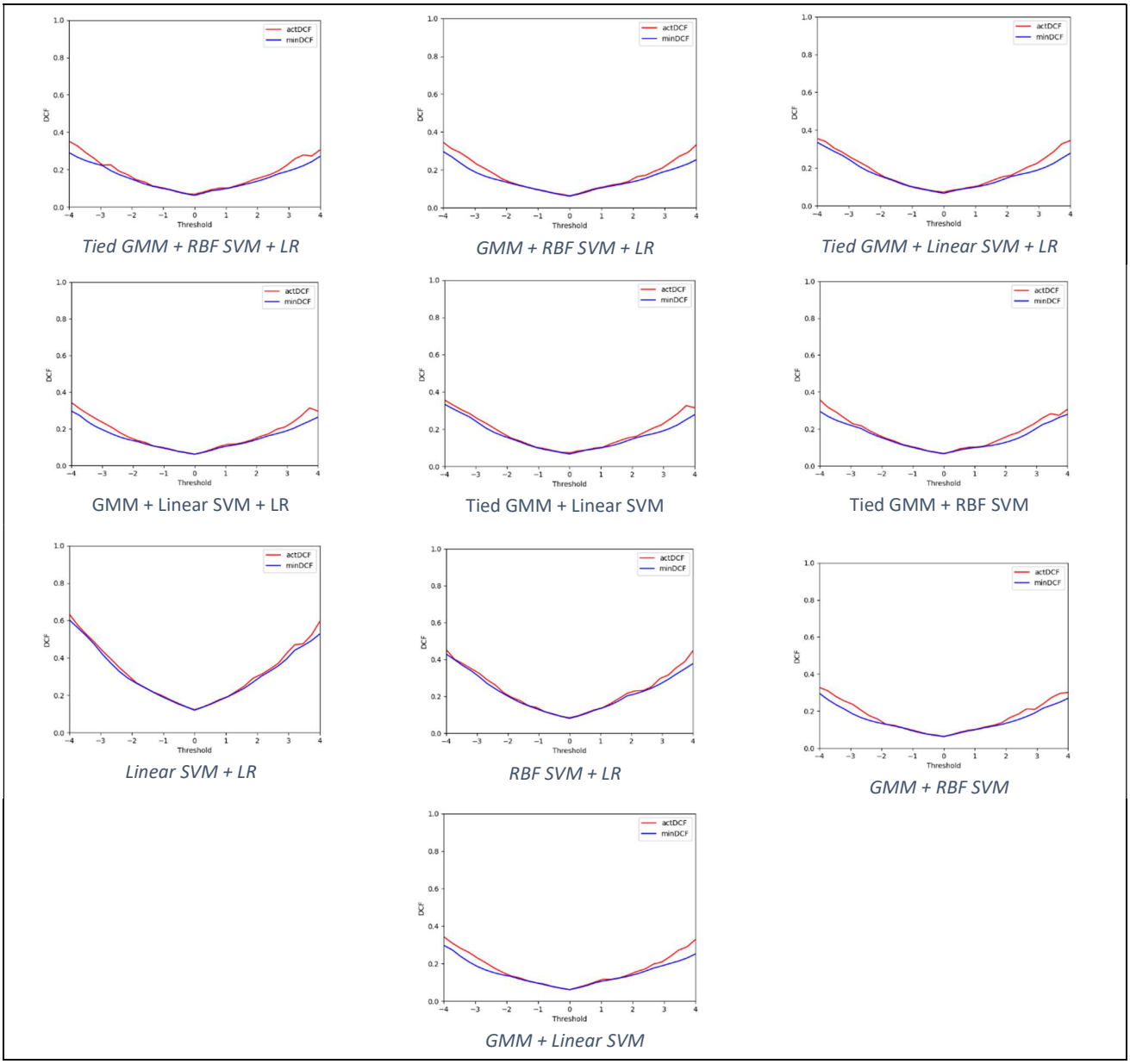


Figure 26: Bayes Error Plots for the fusion models on Evaluation set

We now analyze the behavior of our best models (fusion excluded) on the test dataset. To do it we will consider their hyper parameters on both validation and evaluation datasets and confront the results obtained.

Let's start with Logistic Regression (without Z-score). As we did during the validation phase, we will consider the lambda values from 10^{-5} to 10^{5}.

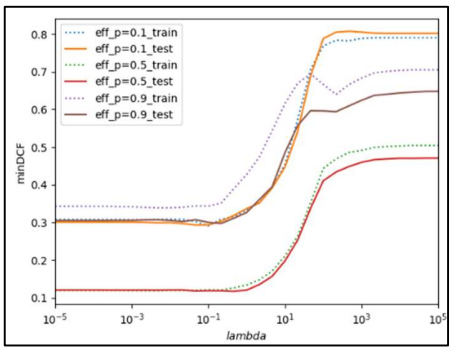


Figure 27: Logistic Regression with π_T=0.5 with validation set and evaluation set

The model follows similar patterns on the 2 datasets with very similar performances (except for π=0.9 that performs a little better with test dataset than it does with train dataset), so we can also confirm the best value for the hyper parameter λ, which was 10^{-5}.

Let's now analyze the SVM models (without Z-score).
For all of them we will consider the same interval for C (10^{-5},10^{5}) of the validation phase.
We will begin by examining the linear SVM with π_T=0.9. As observed with the logistic regression model, the evaluation model does not significantly differ from the validation model.

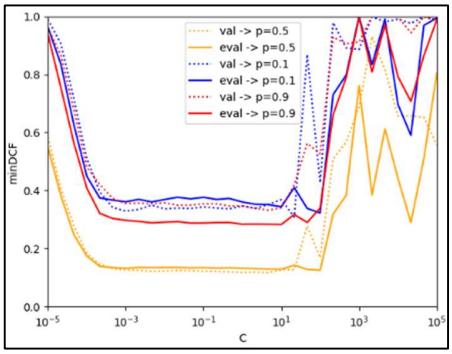
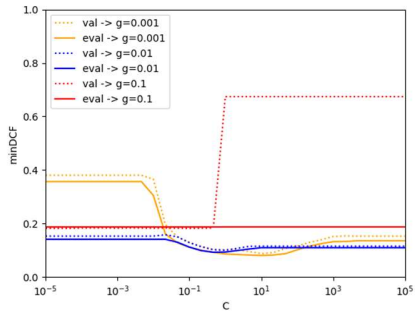


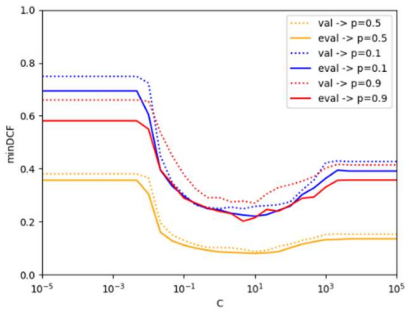
Figure 28: Linear SVM with π_T=0.9 with validation set and evaluation set

We can consider the hyperparameter C=1 as we did for the validation phase.

Now we will compare the behavior of RBF SVMs in validation and evaluation. First, let's analyze the gamma values. As we can see from Figure 29, gamma=0.01 confirms itself as the best value. Finally, considering the behavior of gamma=0.01 with different prior values, we notice once again a consistent curve pattern as observed during validation, thus confirming the value of C=10



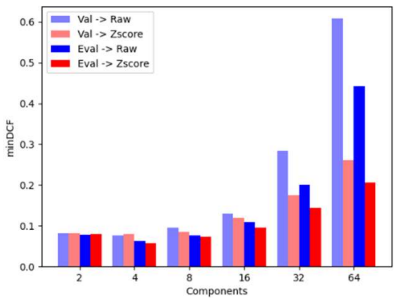
RBF SVM with π_T=0.5



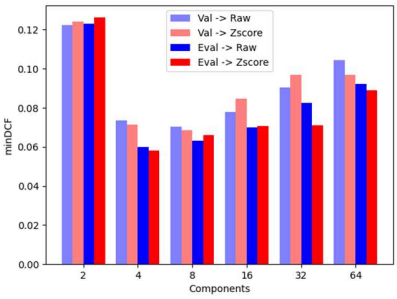
RBF SVM with π_T=0.5 and gamma=0.001

Figure 29: Linear SVM con π_T=0.9 with validation set and evaluation set

Eventually, we analyze the GMM models. Remembering the models that we decided to bring 2 models to the evaluation phase, we decided to evaluate them with PCA equal to 11 because it had the best results for both the GMM and Tied GMM models and we compare their performances with and without Z-score. The only hyper-parameter we need to consider is the number of components and the difference in their performance is shown in figure 30.



GMM with PCA 11



Tied GMM with PCA 11

Figure 30: GMM and Tied GMM with PCA 11 with validation set and evaluation set

Just like we predicted, 8 is not the optimal number of components. In fact, we can notice that in this case we get 2 new models for both GMM and Tied GMM providing the best results, which are GMM and Tied GMM with 4 components and z-score. So, we analyze them in detail showing their

minimum and actual DCF. Again, there are not significant differences in the minDCF and actDCF values, so there seems to be no need for calibration.

	minDCF			actDCF		
	π=0.1	***π= 0.5***	***π= 0.9***	***π=0.1***	***π=0.5***	***π=0.9***
Tied GMM (4 comp. Z-score)	0.19	0.058	0.163	0.20	0.06	0.176
GMM (4 comp. Z-score)	0.148	0.058	0.155	0.158	0.063	0.161

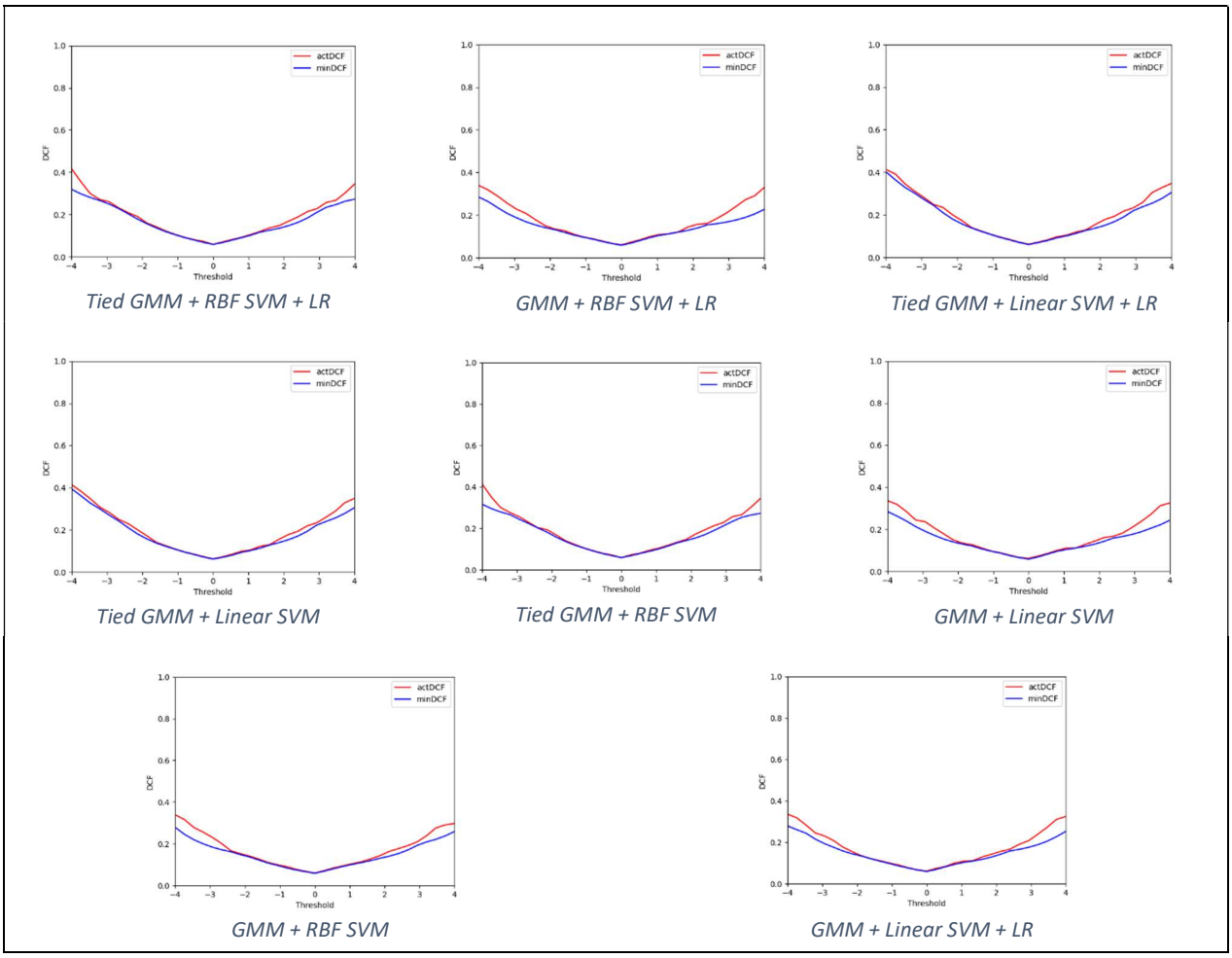
Table 17: Min DCF and Actual DCF values for new Tied GMM and GMM

In the end, our prediction on the correct number of components, which was between 2 and 4, was correct.

Since we found other better performing GMM models from the evaluation phase, we decide to recompute the scores for the fusion models including GMM and Tied GMM.

	minDCF			***actDCF***		
	π= 0.1	***π= 0.5***	***π= 0.9***	***π= 0.1***	***π= 0.5***	***π= 0.9***
Tied GMM + RBF SVM + LR	0.184	0.059	0.151	0.198	0.059	0.177
GMM + RBF SVM + LR	0.144	0.059	0.143	**0.153**	**0.060**	**0.162**
Tied GMM + Linear SVM + LR	0.187	0.061	0.155	0.210	0.063	0.182
GMM + Linear SVM + LR	0.147	0.059	0.146	0.157	0.061	0.163
Tied GMM + Linear SVM	0.187	0.061	0.157	0.208	0.062	0.181
Tied GMM + RBF SVM	0.189	0.059	0.157	**0.200**	**0.059**	**0.168**
GMM + Linear SVM	**0.143**	**0.058**	**0.146**	0.158	0.061	0.157
GMM + RBF SVM	0.151	0.059	0.143	0.160	0.060	0.170

Table 18: Min DCF and Actual DCF values for the fusion obtained with the new models



Tied GMM + RBF SVM + LR | *GMM + RBF SVM + LR* | *Tied GMM + Linear SVM + LR*

Tied GMM + Linear SVM | *Tied GMM + RBF SVM* | *GMM + Linear SVM*

GMM + RBF SVM | *GMM + Linear SVM + LR*

Table 9: Bayes Error Plots for fusion models with the new GMM models

We get better results on the fusion models with the newly computed GMM and Tied GMM scores, furtherly confirming the effectiveness of the new models.
Now the Tied GMM + RBF SVM model seems to provide the best actual DCF, whereas the GMM + Linear SVM provides the best minimum DCF.

7. Conclusions

The best actual DCF (0.059) for our target application is given by the fusion of Tied GMM (z-score, PCA=11, components=4) and RBF SVM (π_T=0.5, C=10, $\gamma=10^{-3}$, K=1), but we found a great variety of models which provide results that are better than the minimum goal of DCF=0.1. For example, we have the fusion of GMM (z-score, PCA=11, components=4), RBF SVM (π_T=0.5, C=10, $\gamma=10^{-3}$, K=1) + LR ($\lambda=10^{-5}$, π_T: 0.5) that provides a slightly worse result (actDCF=0.06) on our target application but way better results on the other 2 applications we used to test our models.

Overall, the choices made during the evaluation phase proved to be effective and the chosen models have similar performances both on the validation and on the test set, exception made for the GMM models where results obtained on the validation phase were not in line with what we got in the evaluation phase.