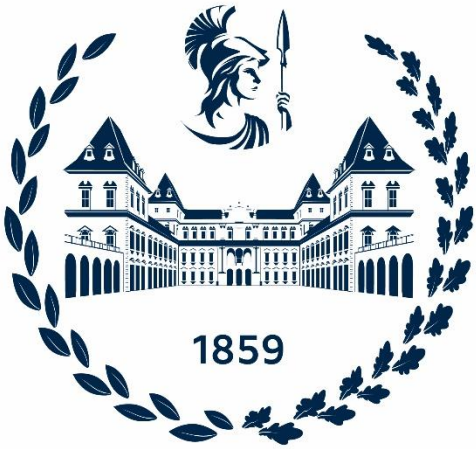


Politecnico di Torino

Stiven Hidri s315147

Gender identification

September 2023

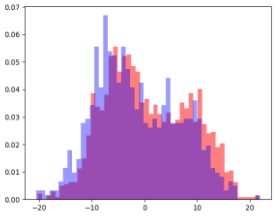
Index

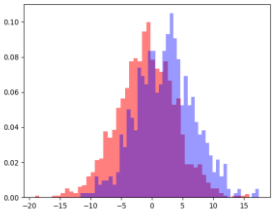
Overview

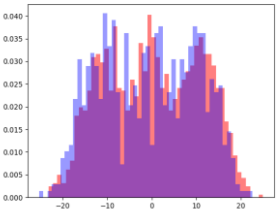
The goal of the project is the development of a classifier able to identify, from features representing face images, the correct gender.
The dataset consists of low-dimensional records obtained by mapping face images to a common, low-dimensional manifold. For tractability and privacy reasons the data is synthetic, containing a much lower dimensionality with respect to real case scenarios. Each record is 12-dimensional belonging to either the male (label 0) or the female (label 1) class. Train and test sets are imbalanced. The former one contains 2400 samples, 720 male samples and 1680 female samples whereas the latter contains 6000 total samples where 4200 are male samples and the remaining 1800 are female samples.
Each record belongs to one of three different age groups, even though such information is not available.

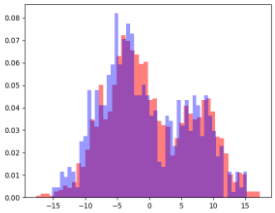
Features Analysis

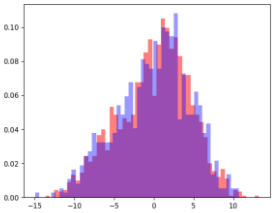
The following pictures show the distributions of all features for both classes.

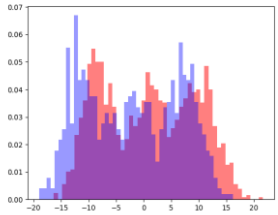

Feature 1

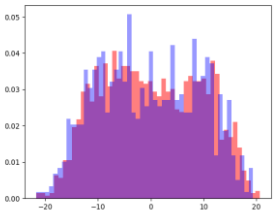

Feature 2

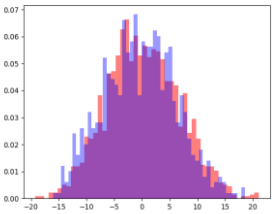

Feature 3

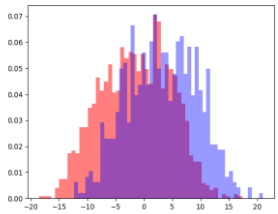

Feature 4

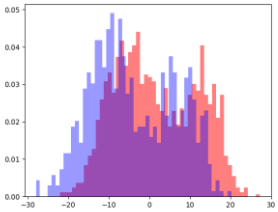

Feature 5

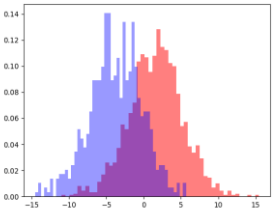

Feature 6

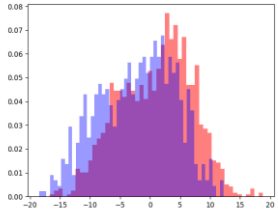

Feature 7


Feature 8


Feature 9

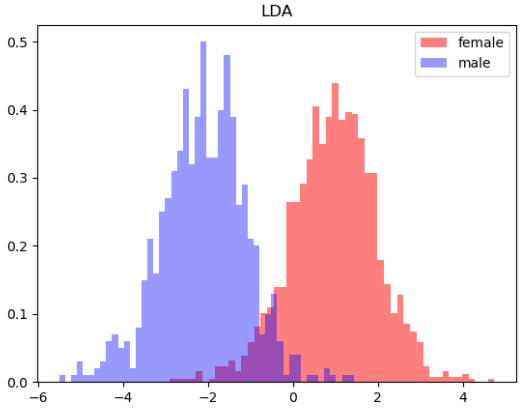

Feature 10


Feature 11

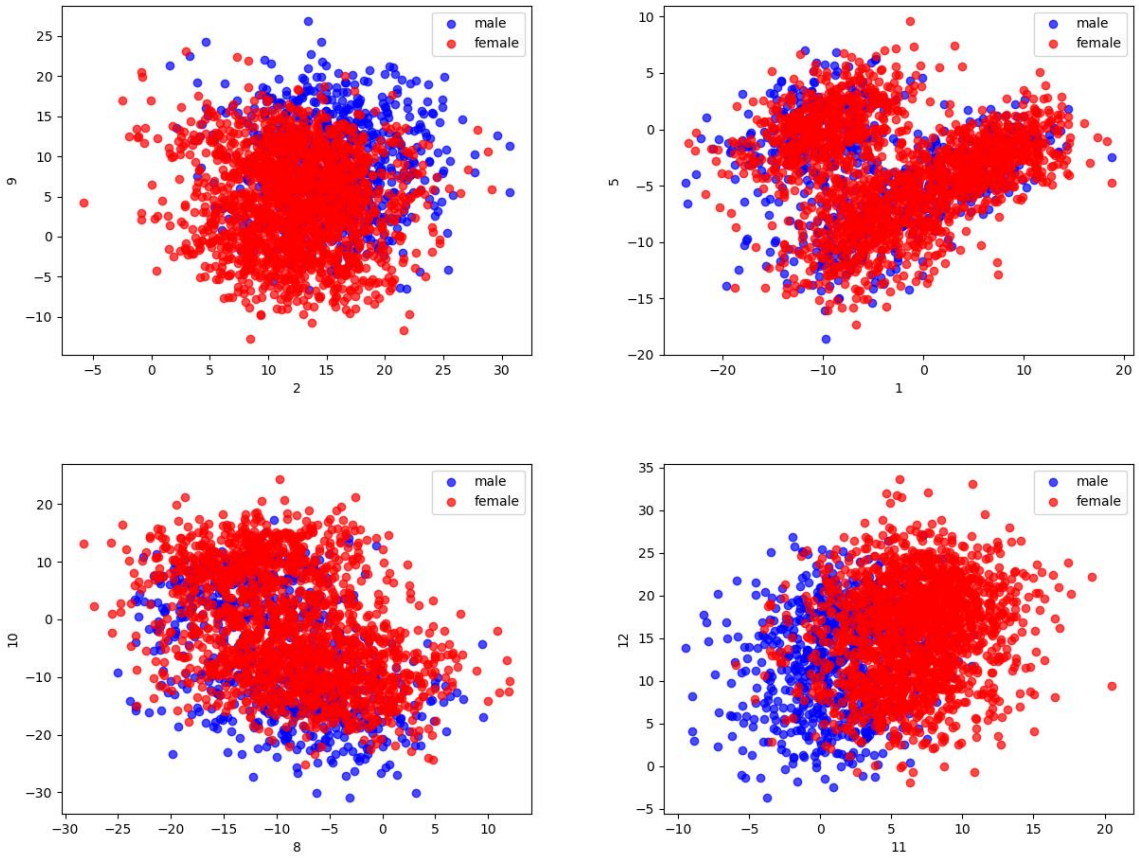

Feature 12

We can see that feature 11 is the most discriminant one. Feature 3 and 6 clearly show a gaussian distribution with 3 components probably caused by the 3 different age groups from which the samples were taken from.

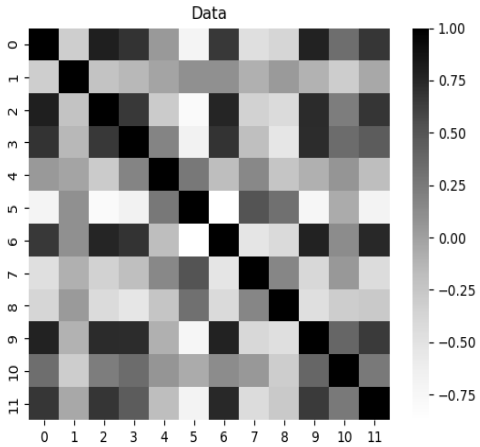
After applying LDA transformation we can see that the 2 classes are likely to be linearly separable, even though we can spot a little overlapping area that could lead to errors. This means that linear separation rules may perform well but may not be the best choices.



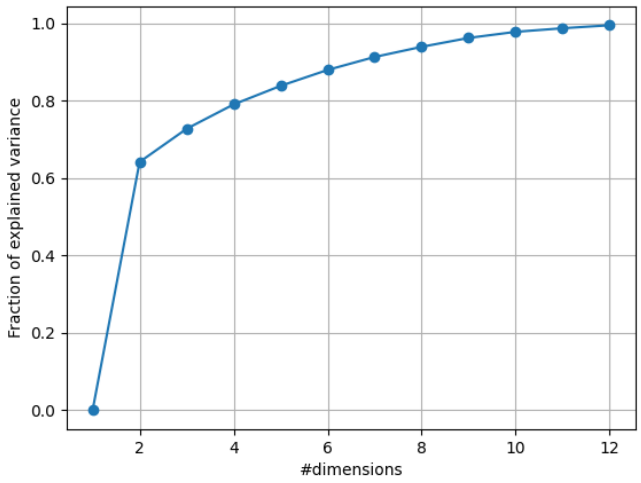
Here are reported some scatter plots of different attributes pairs for both classes. Globular shapes indicate that a Gaussian Distribution could describe well the data. Still we can see different clusters, again the assumption is that they belong to the 3 different age groups. Moreover the within class spread seems to be similar, so models that assume such condition may perform well (tied models).



In the following is reported the heatmap showing the correlation between features regarding the whole dataset. We can see both strong and weak correlations between features. We can then state that models that assume independency between features will not perform well (Naïve Bayes assumption)



Looking at the following graph we can see that if we reduce the dimensionality of the dataset to 8 we still retain at least 90% of the variance. Given this result we could consider lowering the dimensions to 8 through PCA even though, considering the already low number of initial features, lowering it could lead, almost immediately, to a meaningful loss of useful information.



Validation

This report contains the result of the following models:

- Multivariate Gaussian Classifier
  - Full Covariance
  - Naïve Bayes Assumption
  - Tied Covariance
- Linear Logistic Regression
- Support Vector Machine
  - Linear
  - Polynomial Kernel
  - Radial Basis Function Kernel
- Gaussian Mixture Model

Each model has been trained using the K-fold technique with K=10 since it allows to simulate more data for training and validation purposes.
The target application considers a balanced use-case, with a working point defined by the triplet (π = 0.5,Cfn = 1,Cfp = 1). Will be taken into considerations also working points defined by the two triplets (πT = 0.1,Cfn = 1,Cfp = 1) and (πT = 0.9,Cfn = 1,Cfp = 1).
We will use the validation phase in order to train the different models with various settings in order to understand which configuration works better. The previously done features analysis together with the assumptions and working strategies of the different models will help in trying to predict their behaviors.

Multivariate Gaussian Classifier

MVG is a generative model which means that it models the class distribution of the data and then it exploits the Bayes Rule in order to compute the class posterior probabilities.
It has been shown that a gaussian distributions can be a good fit in order to describe the data of both classes. From the heatmap we have spotted features which are highly correlated so we can expect that the Naïve Bayes assumption will not hold and so the MVG Naïve Bayes model will perform poorly with respect to the others. Moreover since the spread within the classes seems similar it can ben forecasted that the tied model will be performing well.

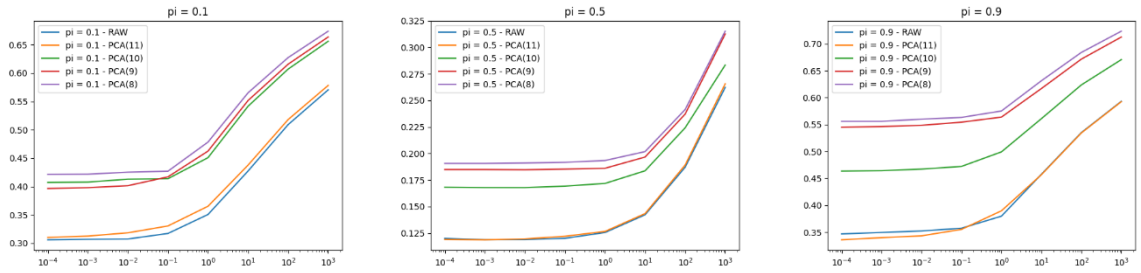
Model	DATA	π = 0.1	π = 0.5	π = 0.9
Full covariance	RAW	0.303	0.114	0.354
	PCA(11)	0.298	0.119	0.356
	PCA(10)	0.423	0.168	0.488
	PCA(9)	0.418	0.187	0.554
	PCA(8)	0.437	0.192	0.560
Naive-Bayes	RAW	0.772	0.464	0.778
	PCA(11)	0.305	0.131	0.362
	PCA(10)	0.446	0.171	0.48
	PCA(9)	0.453	0.191	0.546
	PCA(8)	0.460	0.195	0.551

Tied	**RAW**	0.292	0.112	0.338
	PCA(11)	0.293	0.120	0.345
	PCA(10)	0.395	0.166	0.469
	PCA(9)	0.395	0.180	0.539
	PCA(8)	0.405	0.187	0.557

Let's analyze the result of each model:
The **full covariance** one performs better when keeping the dimensionality close to 12.
The **Naïve-Bayes** model performs poorly as we expected. The performance is increased when we apply PCA since it projects the data over the directions that maintain the highest variance.
In the **tied** case the best performance is reached without applying PCA. This model performs slightly better than the other 2, probably because the distributions of the features for both classes are actually similar.
Both Full Covariance and Tied model highlight a significant loss of performance if we try to lower the number of features below 11. Applying PCA(11) gives results which are in line with the RAW data case.

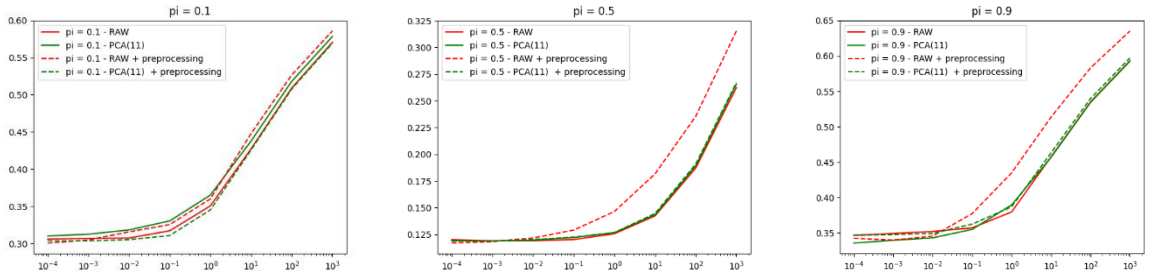
Logistic Regression

Logistic regression is a discriminative model, this means that rather than modelling the distribution of observed samples it directly models the class posterior one.
In particular the regularized version of the model has been implemented, where a norm penalty has been added to the objective function introducing the hyperparameter lambda. It has to be chosen in order to increase the performances of the classifier and the only way to do it is running different configurations of the model and checking which is the best one.
Applying LDA to the dataset has shown that the classes seem to be linearly separable (even though a little overlapping area was present) so we expect performances to be not too bad with respect to the previously described models.



Here it is shown how the minimum DCF varies with respect to lambda in the 3 different working points. Raw data and PCA(11) are the most performant cases where PCA(11) overcomes the Raw data case only when the prior is 0.9. Further decreasing of the dimensionality of the dataset worsens the results significantly. The best value for lambda is 10^{-4} which corresponds to the lowest value of minimum DCF.

When applying Logistic Regression it often helps to adopt some preprocessing strategies. In this case it has been tried to standardize the deviations and to whiten both the Raw data case and the PCA(11) one since they performed better.



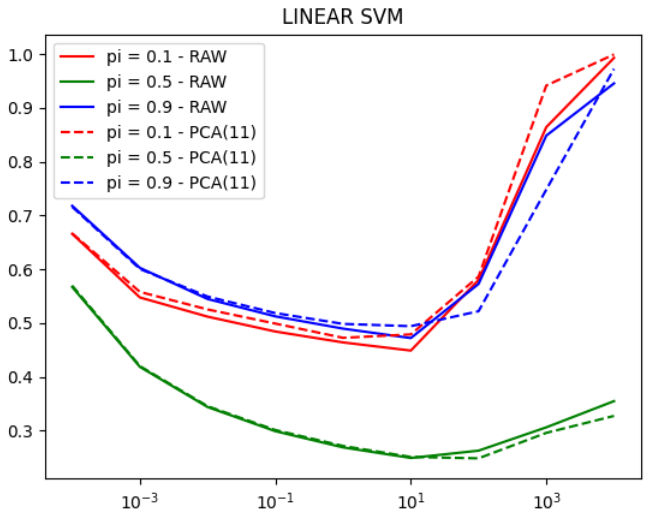
Here we can see that this preprocessing technique actually improves the performance of the model always when applied on Raw data. When applied on PCA(11) we have always an improvement except for the case where the prior is 0.9.
Overall the model is more performant on RAW data and still PCA(11) results are in line with the RAW data case. Again when trying to lower more the number of dimensions a significant loss of performance is shown.
Here in the table are reported the results with lambda equal to 10^{-4} which has given the best results:

Model	Data	π = 0.1	π = 0.5	π = 0.9
Logistic Regression $\lambda=10^{-4}$	**RAW**	0.306	0.120	0.347
	RAW*	0.301	0.117	0.342
	PCA(11)	0.310	0.119	0.336
	PCA(11)*	0.304	0.119	0.347
	PCA(10)	0.407	0.168	0.463
	PCA(9)	0.396	0.185	0.546
	PCA(8)	0.421	0.190	0.556

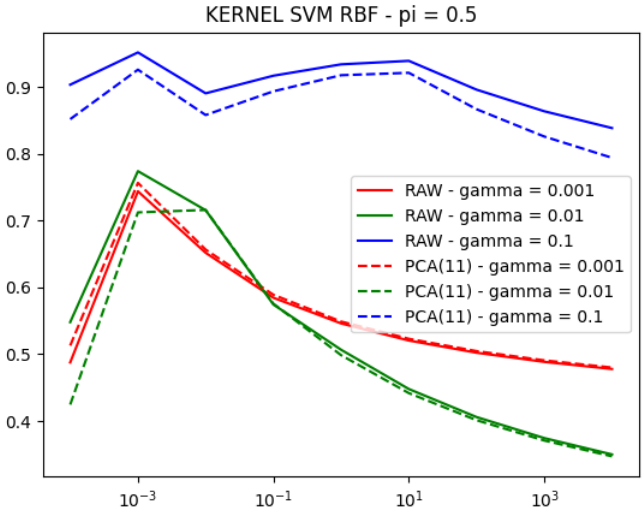
* indicates that preprocessing (standardize deviations + whitening) has been applied

Support Vector Machine

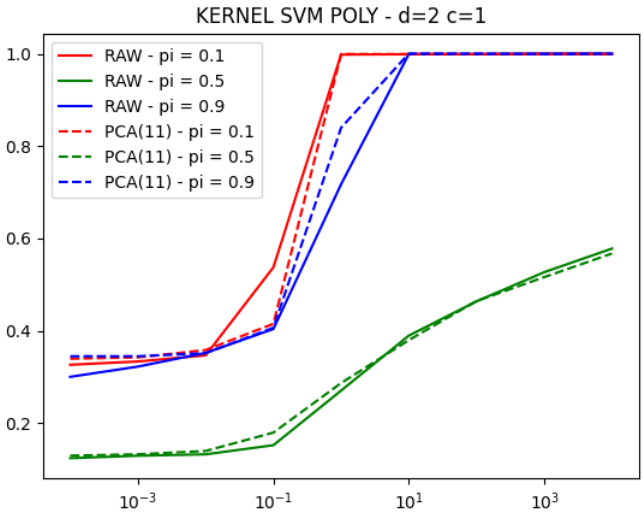
SVM is a discriminative non probabilistic model. This means that it constructs a function that maps feature vectors to a set of scores, without probabilistic interpretation. It tries to find the hyperplane that divide in the best way the classes while trying to maximize the margin i.e. the distance from the separation surface and the closest point of both the 2 classes.
The objective function contains an hyperparameter C. It basically tunes the number of points allowed inside the margin. We again evaluate it running different configurations of the model and picking the best one.
From now on only RAW data and PCA(11) are taken into consideration due to the intensive computations required in order to train the following models. Moreover it has been show until now the further lowering the number of features significantly worsen the performances.
The first model taken into consideration is the linear one. Generally the best performances are reached with the Raw dataset and C = 10



Now we take into consideration models that exploit the concept of kernel. Basically if we have a function called k(x1,x2) that computes efficiently the dot product inside an expanded space it can be used to compute scores in the training phase. There is no rule that tell us how to construct one but fortunately there are well known ones that we can use. We will exploit the Gaussian Radial Basis Function Kernel and the Polynomial Kernel of degree 2.
SVM RBF Kernel performs the best with C=10^4 and lambda=0.01 applied on PCA(11) (still very similar result with respect to RAW data)



SVM POLY Kernel performs the best with C = 10^{-4} and RAW data.



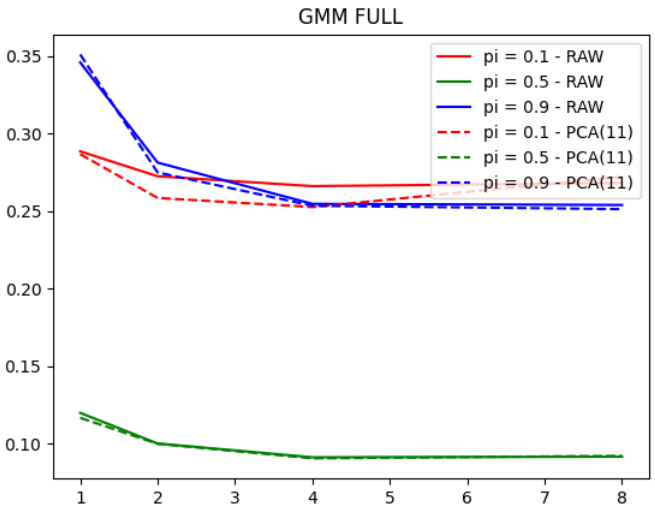
In the following table are reported the summary of the results:

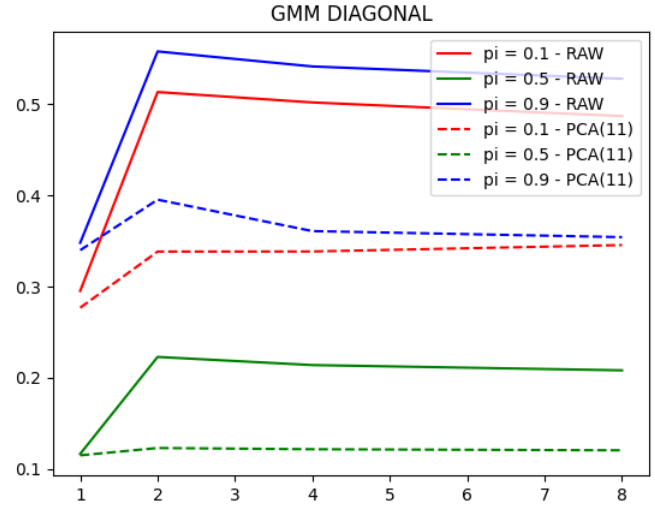
Model	Data	π = 0.1	π = 0.5	π = 0.9
Linear SVM	**RAW** c=10	0.449	0.249	0.490
	PCA(11) c=10	0.479	0.250	0.494
Kernel SVM RBF	**RAW** γ=0.001 c=10^4	0.864	0.478	0.821
	RAW γ=0.01 c=10^4	0.710	0.350	0.699
	RAW γ=0.1 c=10^4	0.995	0.838	0.995
	PCA(11) γ=0.001 c=10^4	0.864	0.480	0.828

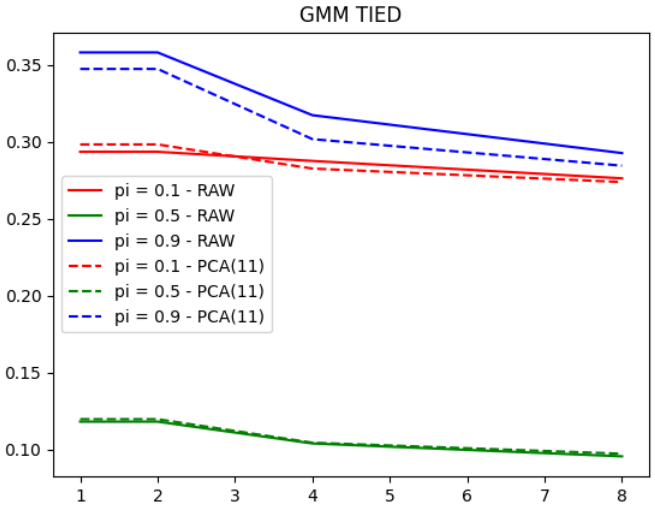
	PCA(11) γ=0.01 $c=10^4$	0.703	0.347	0.694
	PCA(11) γ=0.1 $c=10^4$	0.996	0.794	0.993
Kernel SVM POLY	**RAW** $c=10^{-4}$	0.326	0.124	0.300
	PCA(11) $c=10^{-4}$	0.339	0.129	0.344

Gaussian Mixture Model

GMM is a model then can be interpreted as a weighted sum of Gaussian distributions. It allows us to estimate any sufficiently regular distribution up to a certain degree. Looking at our dataset we have spotted 3-compononents gaussians that probably describe the three age groups. Since that LBG algorithm has been implemented we are allowed to have only power of 2 components for our gaussian. Said so it can be forecasted that 2 or 4 components might be best to describe the data.
The 3 following graphs show how the minimum DCF varies with respect to the number of components chosen in the 3 versions of the model: Full, Diagonal and Tied.







Here it is shown the table summarizing all the results.

Model	DATA	π = 0.1	π = 0.5	π = 0.9
GMM FULL	**RAW** d=4	0.266	0.091	0.255
	PCA(11) d=4	0.253	0.090	0.254
GMM TIED	**RAW** d=8	0.276	0.096	0.292
	PCA(11) d=8	0.273	0.097	0.284
GMM DIAGONAL	**RAW** d=8	0.487	0.208	0.528
	PCA(11) d=8	0.346	0.120	0.354

At the end GMM Full covariance with 4 components and PCA(11) performed the best. Follows the Tied model with 8 components and lastly the Diagonal one which performs much better applying PCA(11). The first 2 actually have similar performances on both RAW data and PCA(11)

Models comparison

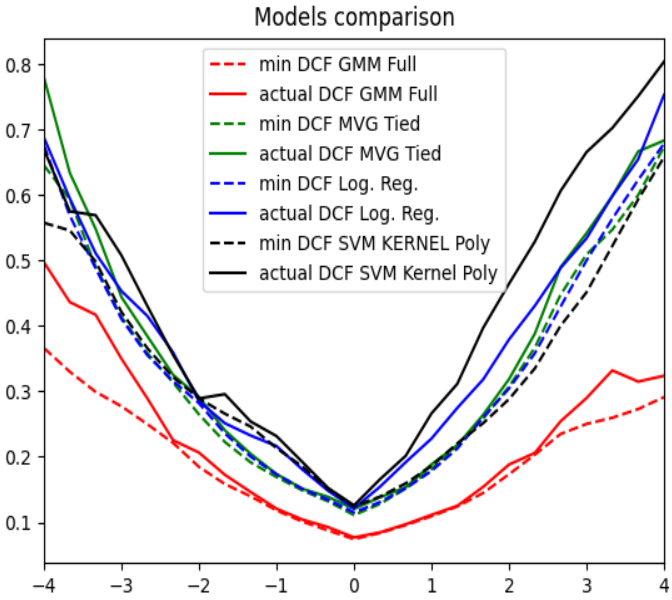
Here are reported the results of the models (the best one for each family) that have better performed:

#	Model	PCA	π = 0.1	π = 0.5	π = 0.9
1	**GMM FULL** d=4	**11**	0.253	0.090	0.254
2	**MVG Tied**	**RAW**	0.292	0.112	0.338
3	**Logistic Regression** $\lambda=10^{-4}$	**RAW***	0.301	0.117	0.342
4	**Kernel SVM POLY** $c=10^{-4}$	**RAW**	0.326	0.124	0.300

* indicates that preprocessing (standardize deviations + whitening) has been applied

Since the gap from GMM Full covariance and all the others is considerable it has been chosen as best possible solution.

Here is shown the Bayes Error Plot for all the above mentioned models. We can see that MVG Tied is already well calibrated: the actual DCF is always very close to the minimum one. All the other would benefit from calibration. Our best performing model, GMM with full covariance matrix, is well calibrated for threshold values that have |t|<2.5.



Evaluation

We will perform the evaluation step using the proposed solution and the other most promising models mentioned above. This will be done training the model on the whole training set without folds and then evaluating it on the evaluation set.

Model	PCA	π = 0.1	π = 0.5	π = 0.9
GMM FULL Evaluation d = 4	**11**	0.156	0.061	0.156
GMM FULL Training d = 4	**11**	0.253	0.090	0.254
MVG Tied Evaluation	**RAW**	0.301	0.116	0.308
MVG Tied Training	**RAW**	0.292	0.112	0.338
Logistic Regression Evaluation $\lambda = 10^{-4}$	**RAW***	0.314	0.122	0.314
Logistic Regression Training $\lambda = 10^{-4}$	**RAW***	0.301	0.117	0.342
Kernel SVM POLY Evaluation $c=10^{-4}$	**RAW**	0.310	0.111	0.277
Kernel SVM POLY Training $c=10^{-4}$	**RAW**	0.326	0.124	0.300

* indicates that preprocessing (standardize deviations + whitening) has been applied

Conclusions

The choices that have been taken during the training phase have been proved effective. It can be underlined the strong similarity between the results obtained during validation and evaluation phases which indicates that the evaluation population is similarly distributed with respect to the training one. Using GMM with full covariance matrix we have been able to achieve a minimum DCF of 0.061 for the target application (π=0.5) and for the other two application taken into consideration (π=0.1 and π=0.5) we achieved 0.156 respectively.