Machine Learning and Pattern Recognition

Final project – Gender detection

Academic Year 2022/2023

Michele Mogavero – s308835
Raffaele Viola – s309063

Introduction

The task of the project is the development of a binary classifier able to distinguish male ($class\ label\ =\ 0$) and female ($class\ label\ =\ 1$) samples in a dataset composed by image embeddings.
Each sample of the dataset corresponds to a 12-dimensional continuous-valued vector, whose features has no physical interpretation.
Each sample may belong to one out of three different age groups and each age group may have different distributions of features; although, the age information is not given.
The project relies on a training set and an evaluation set: the first is used to build the classification models, while the latter is used to test how well the models perform.
Both datasets have unbalanced class distributions, but in different ways: the training set is composed by 1680 samples belonging to the female class and 720 samples belonging to the male class, while the evaluation set is composed by 4200 samples belonging to the male class and 1800 samples belonging to the female class.
Thus, we can observe that the training set is unbalanced towards the female class and the evaluation set is unbalanced towards the male class.

Dataset analysis

Graphical representation may be useful to show and comment some interesting aspects of the dataset.
The following heatmaps show the Pearson correlation of each feature in the training dataset, raw and centred.
An important clarification must be made: the Pearson coefficient is a number between -1 and 1 which is an index of the correlation between features; for $0 < r < 1$ we have positive correlation and for $-1 < r < 0$ we have negative correlation.
Thus, since we only have interest in showing how strongly two features are correlated (positively or negatively), in the following heatmaps the Pearson coefficient is taken with its absolute value.

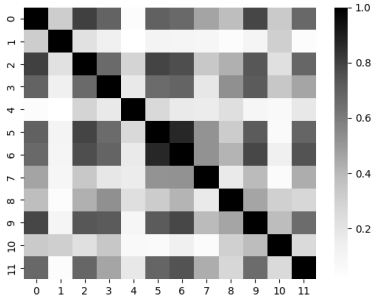

Figure 1 - Heatmap of the whole dataset

It is possible to observe that most features are slightly correlated, with some exceptions (respectively, the darker and brighter ones are the most and less correlated).
We now consider the heatmaps regarding the single classes.

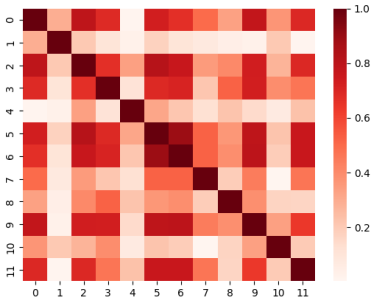

Figure 2 - Heatmap of the Male class

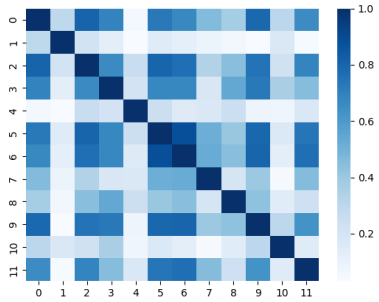

Figure 3 - Heatmap of the Female class

Once again, for the heatmaps of the single classes we can repeat the same considerations made on the similarity of the heatmap of the whole training dataset.
From this first analysis we can assume that, because of the correlation between the features, the Naïve Bayes model may perform poorly.
In addition, the MVG and Naïve Bayes models may have similar performances to their respective tied versions because the heatmaps of the classes look very similar.

We now consider the histograms showing the distribution of each feature:

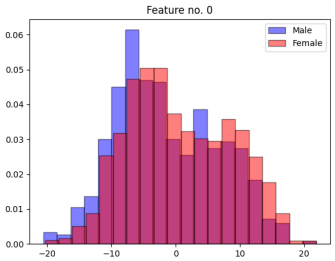

Figure 4 – Distribution of Feature 0

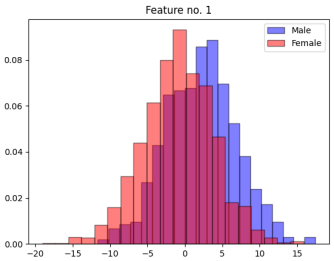

Figure 5 – Distribution of Feature 1

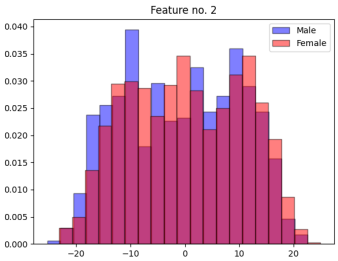

Figure 6 – Distribution of Feature 2

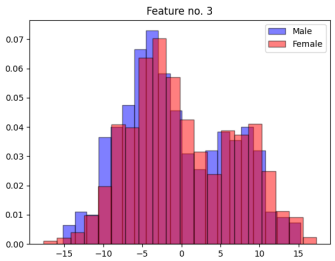

Figure 7 – Distribution of Feature 3

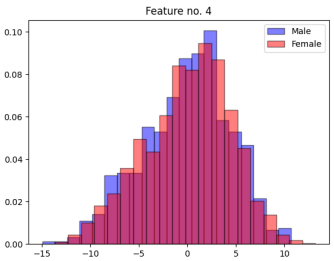

Figure 8 – Distribution of Feature 4

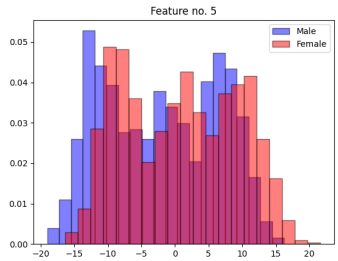

Figure 9 – Distribution of Feature 5

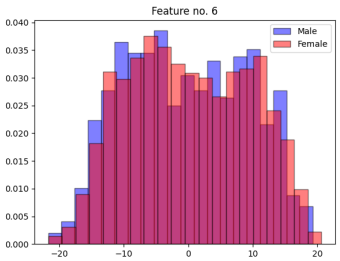

Figure 10 – Distribution of Feature 6

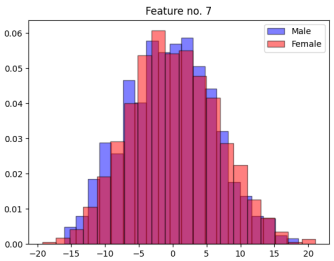

Figure 11 – Distribution of Feature 7

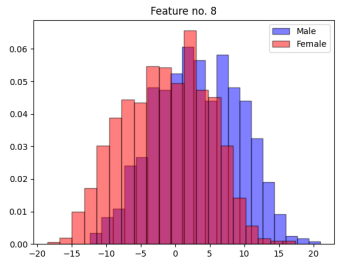

Figure 12 – Distribution of Feature 8

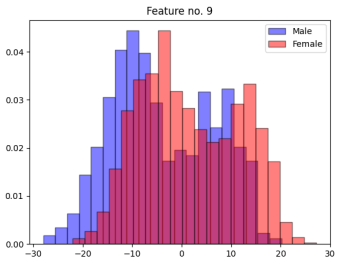

Figure 13 – Distribution of Feature 9

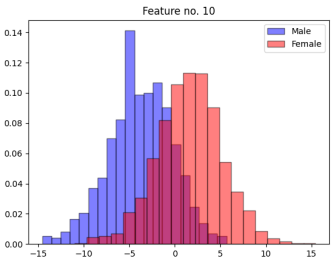

Figure 14 – Distribution of Feature 10

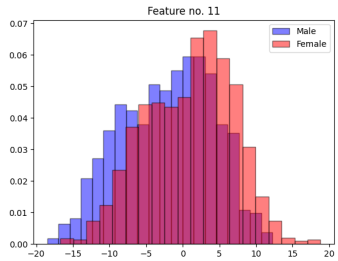

Figure 15 – Distribution of Feature 11

We can see that most of the features follows a Gaussian trend, while some of them may be seen as the composition of two (feature 3, 9) or even three Gaussian models (feature 6).
It is also possible to notice that all the distributions overlap or almost overlap for both classes except for feature 10, which is found to be the most discriminant one.

We can also consider scatter plots of pairs of features, of which we report some significant examples:

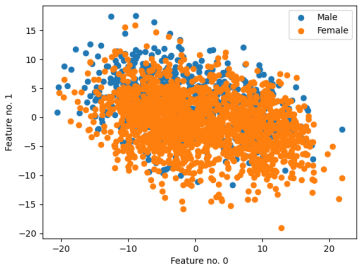

Figure 16 – Scatter plot, feature 0 vs feature 1

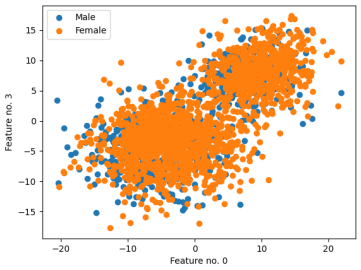

Figure 17 – Scatter plot, feature 0 vs feature 3

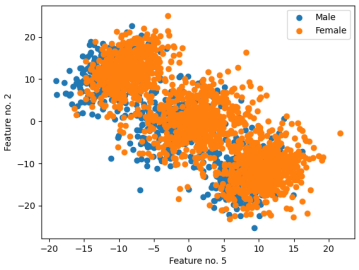

Figure 18 – Scatter plot, feature 5 vs feature 2

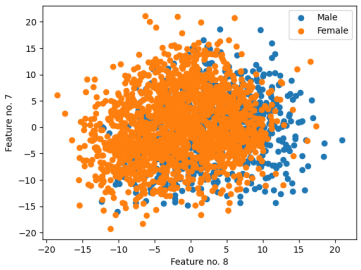

Figure 19 – Scatter plot, feature 8 vs feature 7

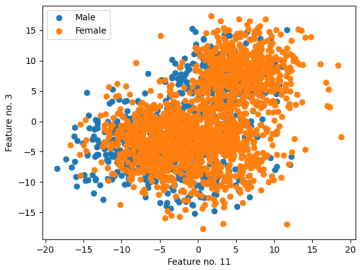

Figure 20 – Scatter plot, feature 11 vs feature 3

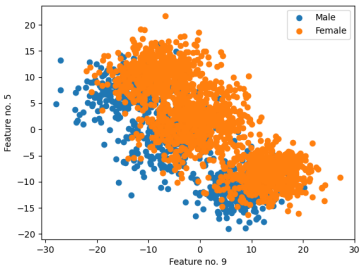

Figure 21 – Scatter plot, feature 9 vs feature 5

Even though in some cases we can observe a single-component attitude of the features (figure 16, figure 19), we can also find trends with two (figure 17, figure 20) or even three (figure 18, figure 21) clusters: this probably happens because samples of the dataset are actually divided into three age groups, but it cannot be confirmed since we have no information about them.
We finally analyse the possible usage of some dimensionality reduction techniques.
Having each sample 12 dimensions, it could be useful to map the dataset onto a m-dimensional subspace, with $m < 12$.
The following graph (figure 25) shows the trend of the fraction of the explained variance (which is an indicator of much information from the original dataset is preserved in the projected one) in function of all the possible m factors exploitable by the **PCA** (Principal Component Analysis) algorithm.

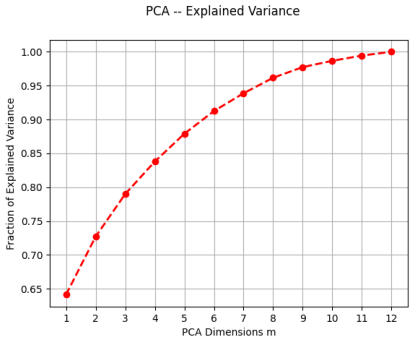

Figure 22 – Fraction of the explained variance fraction in function of the number of the PCA subspace

We can therefore notice that the smallest m that preserves at least the 95% of the original information is 8.
Given a generic dataset with C classes, the **LDA** (Linear Discriminant Analysis) algorithm can find at most $C - 1$ discriminant directions (with largest separation between classes and small spread inside each class).
Thus, being this a binary problem, LDA can find one and only one discriminant direction, which is an indicator of how much the two classes are linearly separable.

The following histogram reports the result obtained after the application of the LDA algorithm.

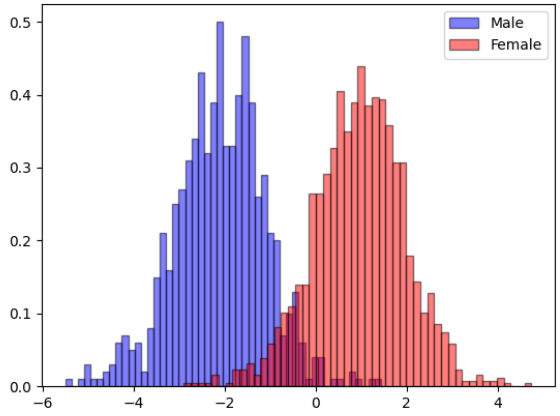

Figure 23 – Binary classification with LDA

We can see that the two classes are quite well linearly separable, even though a not negligible number of samples may be misclassified: this may suggest that a non-linear and/or Gaussian models may perform better than linear ones.

Model analysis

In this section we will analyse various models, considering the performance of the various classifiers and the impact on the application of some pre-processing techniques.
We will use a **K-fold cross validation** (i.e., a process that splits the training set in K folds, using $K-1$ of them as training data and the remaining one for the evaluation; it allows to exploit the given training set more efficiently and to have a more accurate performance estimation) with $K = 5$.
We will also take into account the application of the **z-score** pre-processing technique, which consists in centring the dataset and dividing it by its standard deviation.
Various applications will be analysed; they are characterized by the following working points:
$(\pi_T = 0.5,\ C_{ft} = 1, C_{fn} = 1)$, $(\pi_T = 0.2,\ C_{ft} = 1, C_{fn} = 1)$, $(\pi_T = 0.8,\ C_{ft} = 1, C_{fn} = 1)$.
However, our target application is $(\pi_T = 0.5,\ C_{ft} = 1, C_{fn} = 1)$
The metric used for the performance evaluation is the $\boldsymbol{DCF_{min}}$ (minimum Detection Cost Function), which corresponds to the cost we would pay if the optimal threshold for the prediction was known.

Gaussian classifiers

The first objects of our analysis are Gaussian classifiers, which are MVG, Naïve Bayes and their corresponding tied versions (i.e., the same classifiers but with the assumption that all classes share the same covariance matrix).
In the following tables the values of each DCFmin are reported; they have been computed for each working point, both on the raw data and on data normalized with the z-score, with and without the application of PCA with the m value in the range 8-12.
The performed measurements are reported below.

	PCA NOT APPLIED					
	RAW			**ZSCORE**		
Model	$\boldsymbol{\pi = 0.5}$	$\boldsymbol{\pi = 0.2}$	$\boldsymbol{\pi = 0.8}$	$\boldsymbol{\pi = 0.5}$	$\boldsymbol{\pi = 0.2}$	$\boldsymbol{\pi = 0.8}$
MVG	0.117	0.217	0.233	0.117	0.217	0.233
TMVG	0.116	0.195	0.223	0.116	0.195	0.222
NB	0.465	0.747	0.668	0.465	0.747	0.668
TNB	0.461	0.749	0.678	0.461	0.749	0.678

	PCA(12)					
	RAW			ZSCORE		
Model	$\pi = 0.5$	$\pi = 0.2$	$\pi = 0.8$	$\pi = 0.5$	$\pi = 0.2$	$\pi = 0.8$
MVG	0.117	0.217	0.233	0.117	0.217	0.233
TMVG	0.116	0.195	0.223	0.116	0.195	0.223
NB	0.129	0.224	0.239	0.120	0.201	0.230
TNB	0.120	0.205	0.238	0.118	0.203	0.232

	PCA(11)					
	RAW			ZSCORE		
Model	$\pi = 0.5$	$\pi = 0.2$	$\pi = 0.8$	$\pi = 0.5$	$\pi = 0.2$	$\pi = 0.8$
MVG	0.123	0.207	0.232	0.125	0.203	0.232
TMVG	0.121	0.209	0.241	0.122	0.205	0.237
NB	0.138	0.229	0.241	0.128	0.216	0.234
TNB	0.124	0.204	0.252	0.123	0.206	0.240

	PCA(10)					
	RAW			ZSCORE		
Model	$\pi = 0.5$	$\pi = 0.2$	$\pi = 0.8$	$\pi = 0.5$	$\pi = 0.2$	$\pi = 0.8$
MVG	0.167	0.308	0.325	0.187	0.352	0.365
TMVG	0.169	0.304	0.313	0.183	0.325	0.360
NB	0.175	0.311	0.321	0.181	0.355	0.375
TNB	0.173	0.306	0.314	0.181	0.326	0.366

	PCA(9)					
	RAW			ZSCORE		
Model	$\pi = 0.5$	$\pi = 0.2$	$\pi = 0.8$	$\pi = 0.5$	$\pi = 0.2$	$\pi = 0.8$
MVG	0.187	0.332	0.369	0.220	0.388	0.423
TMVG	0.185	0.314	0.369	0.212	0.362	0.411
NB	0.192	0.342	0.371	0.212	0.384	0.438
TNB	0.188	0.352	0.375	0.210	0.369	0.433

	PCA(8)					
	RAW			ZSCORE		
Model	$\pi = 0.5$	$\pi = 0.2$	$\pi = 0.8$	$\pi = 0.5$	$\pi = 0.2$	$\pi = 0.8$
MVG	0.195	0.348	0.379	0.265	0.458	0.516
TMVG	0.190	0.325	0.373	0.269	0.452	0.494
NB	0.196	0.345	0.379	0.266	0.450	0.511
TNB	0.193	0.336	0.390	0.259	0.441	0.509

The first thing that we can observe is that, even though we have seen that for $m \geq 8$ the PCA algorithm preserves more than 95% of the information of the original training set, it is not convenient to use values of m lower than 11 because they lead to a not negligible increasing of the DCF_{min}.
Also, we can notice that for such values the DCF_{min} of raw and normalized data are very close, or even the same in the case we do not apply PCA.

We can also observe that the previous hypothesis were correct: we now can confirm that Gaussian models perform better than the Naïve Bayes ones (which assume that there is no correlation between features), as predicted by the slight correlation between features seen in the heatmaps.
This is also confirmed by the fact that the Gaussian classifiers suffer less from the loss of information due to the decreasing of m because the features of the dataset follow Gaussian trends and PCA project the samples on the directions which preserve the highest variance.
On the other hand, we can also notice how PCA significantly improves the performances of both Naïve Bayes models: this happens because the algorithm projects the samples on orthogonal directions, which implies independence of features, that is perfectly in line with the Naïve Bayes assumptions.
As regards the working points, ($\pi_T = 0.5,\ C_{ft} = 1, C_{fn} = 1$) is the one that ensures the best results, while the other have comparable but slightly worst performances.
In the end, we select as best model the **Tied MVG with z-score**.

Logistic Regression

We now shift the object of our analysis on discriminative models.
We focus on the prior-weighted version of the Logistic Regression.
We will analyze both Linear and Quadratic models, trying to estimate the parameters that make them working better through cross validation.
Starting with Linear Logistic Regression, we will estimate the most suitable value for the regularization term λ.
In the graph reported below, we can study the variation of DCF_{min} in function λ.

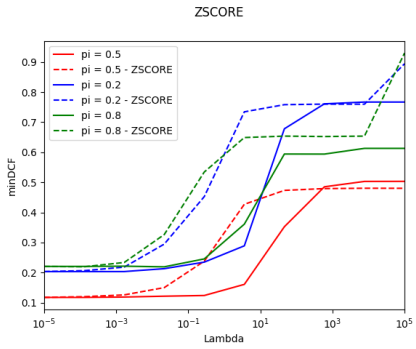

Figure 24 – LLR, Variation od DCF_{min} in function of λ

We can observe that z-score pre-processing seems to be ineffective.
$\lambda = 10^{-3}$ seems to be a good choice, indeed choosing a too low value (e.g., 10^{-5}) may lead to overfitting and a too high value would simplify the model giving too much weight to the regularization term.
We also analyze the effect of PCA on Linear Logistic Regression.

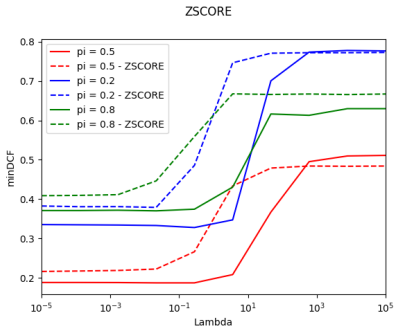

Figure 25 – LLR, Variation of DCF_{min} in function of λ + PCA(9)

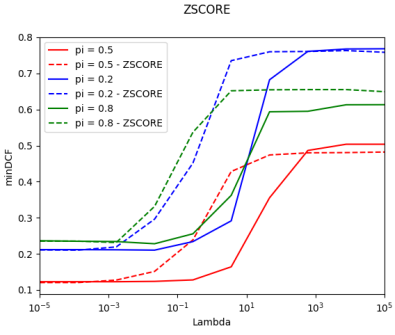

Figure 26 – LLR, Variation of DCF_{min} in function of λ + PCA(11)

Applying PCA does not lead to better performances.
Given λ, we can analyze the impact of the training prior on which we train the model.

The performed measurements are reported below.

		RAW($\lambda = 10^{-3}$)			ZSCORE($\lambda = 10^{-3}$)		
Model	$\boldsymbol{\pi_T}$	$\boldsymbol{\pi = 0.5}$	$\boldsymbol{\pi = 0.2}$	$\boldsymbol{\pi = 0.8}$	$\boldsymbol{\pi = 0.5}$	$\boldsymbol{\pi = 0.2}$	$\boldsymbol{\pi = 0.8}$
LogReg	0.2	0.124	0.203	0.235	0.125	0.219	0.244
LogReg	0.3	0.123	0.206	0.228	0.125	0.212	0.237
LogReg	0.4	0.121	0.209	0.227	0.125	0.213	0.234
LogReg	0.5	0.118	0.211	0.221	0.122	0.210	0.232
LogReg	0.6	0.117	0.209	0.222	0.121	0.209	0.228
LogReg	0.7	0.117	0.212	0.220	0.121	0.209	0.229
LogReg	0.8	0.116	0.217	0.220	0.121	0.212	0.231

We can see that changing the training prior does not deeply affect the performance of the model.
The best value for the training prior is 0.8, probably because the training dataset is unbalanced in favor of the female class (class 1), but we keep as best model the one with prior 0.7 to mitigate this issue.
We can now focus on the quadratic version of Logistic Regression.
As before, we try to estimate the most suitable value for λ, studying the plot reported below, that represents the variation of DCF_{min} in function of λ.

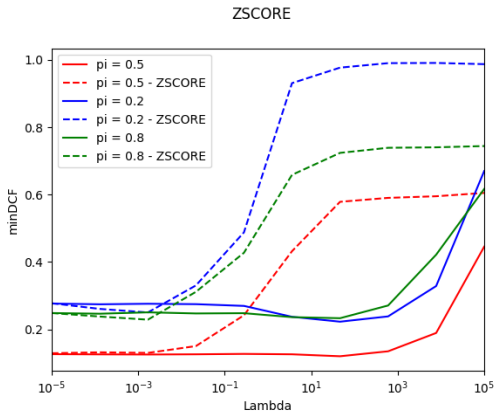

Figure 27 – QLR, Variation of DCF_{min} in function of λ

We can observe that the model with z-score performs equally to the one applied on raw data for the target application and slightly better for the other working points until $\lambda = 10^{-3}$ and then the performance get worse for all the applications.
The quadratic version of the model is more sensitive to the pre-processing step.
Since without preprocessing the model performs better and this model has a higher computational cost, we will not analyze the z-score version in the next validation step.
We will now consider PCA dimensionality reduction to estimate the best parameters for the model.

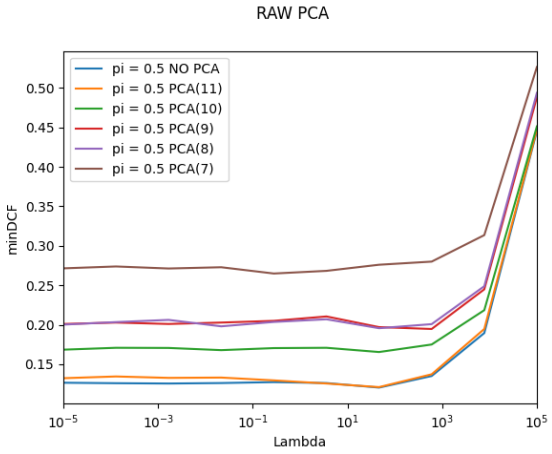

Figure 28 – QLR, Variation of DCF_{min} in function of λ + PCA

From the graph reported above, we can observe that PCA does not improve the performances of the model.
In particular, the best PCA m value is 11 but the performances are the same or slightly worse respect to raw data.
Thus, we select as best model Quadratic Logistic Regression with no PCA and $\lambda = 10^2$.
We will now try different training priors as we have done for the linear version.
The models on raw data are trained with $\lambda = 10^2$ and the models with z-score preprocess step are trained with $\lambda = 10^{-3}$, as we can see from the results reported below.

		RAW($\lambda = 10^2$)			**ZSCORE**($\lambda = 10^{-3}$)		
Model	$\boldsymbol{\pi_T}$	$\boldsymbol{\pi = 0.5}$	$\boldsymbol{\pi = 0.2}$	$\boldsymbol{\pi = 0.8}$	$\boldsymbol{\pi = 0.5}$	$\boldsymbol{\pi = 0.2}$	$\boldsymbol{\pi = 0.8}$
QLogReg	0.2	0.123	0.214	0.248	0.132	0.253	0.239
QLogReg	0.3	0.125	0.224	0.244	0.131	0.255	0.232
QLogReg	0.4	0.123	0.230	0.237	0.130	0.257	0.232
QLogReg	0.5	0.121	0.233	0.235	0.129	0.259	0.229
QLogReg	0.6	0.120	0.240	0.234	0.127	0.249	0.230
QLogReg	0.7	0.119	0.254	0.231	0.126	0.249	0.223
QLogReg	0.8	0.121	0.273	0.232	0.127	0.243	0.224

The performances are similar if compared to Linear Logistic Regression and we can therefore draw the same conclusions.
Among the Logistic Regression classifiers, we choose the Linear Logistic Regression, because the performances are slightly better than the Quadratic version and the cost is obviously lower.
We also observe that, among all the considered working points, the cost for ($\pi_T = 0.5,\ C_{ft} = 1, C_{fn} = 1$) is the lowest, but the model chosen is the one that best performs in this phase among the other evaluated.
We can therefore conclude that the best choice is **Linear Logistic Regression ($\boldsymbol{\lambda = 10^{-3}}$, $\boldsymbol{\pi_T}$= 0.7)**.

Support Vector Machines

The **SVM** (Support Vector Machine) is a non-probabilistic algorithm, whose aim is to find the hyperplane which separates the classes with the largest margin (i.e., the distance of the closest point to the separation hyperplane).
For linear surfaces, the objective function to maximize is $max_\alpha\ \alpha^T \mathbf{1} - \frac{1}{2}\ \mathbf{H}\alpha$, subject to
$0 \leq \alpha_i \leq C_i\ , i = 1, \ldots, n$ (**dual SVM**).
This is a linear function in α, but the parameter we focus on is C, which represents the margin: we should choose the value that grants the best trade-off between margin width (large values of C) and errors on the training set (small values of C).

For the choose of the most appropriate C, we consider the corresponding variations of the $DCF_{\min}$ (related to each working point, both for raw data and z-score) over the range $[10^{-5}, 10^{5}]$, reported on the x axis of the following graph.

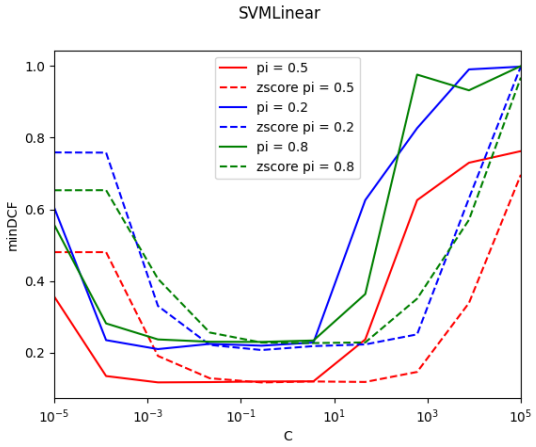

Figure 29 – Linear SVM, Variation of DCF_{min} in function of C

From the graph we select $C = 10^{-1}$ for raw data and $C = 10$ for the data pre-processed with z-score and we test the application on a set of π_T.
The performed measurements are reported below.

		RAW($C = 10^{-1}$)			**ZSCORE**($C = 10$)		
Model	$\boldsymbol{\pi_T}$	$\boldsymbol{\pi = 0.5}$	$\boldsymbol{\pi = 0.2}$	$\boldsymbol{\pi = 0.8}$	$\boldsymbol{\pi = 0.5}$	$\boldsymbol{\pi = 0.2}$	$\boldsymbol{\pi = 0.8}$
SVMLinear	0.2	0.127	0.219	0.248	0.126	0.220	0.239
SVMLinear	0.3	0.123	0.210	0.229	0.119	0.211	0.232
SVMLinear	0.4	0.119	0.217	0.225	0.120	0.212	0.228
SVMLinear	0.5	0.118	0.223	0.229	0.118	0.222	0.227
SVMLinear	0.6	0.118	0.222	0.228	0.117	0.224	0.226
SVMLinear	0.7	0.118	0.219	0.233	0.116	0.224	0.220
SVMLinear	0.8	0.117	0.235	0.225	0.116	0.219	0.224

We can notice that in general we have good results for all the applications, but the cost is inversely proportional respect to the training prior.
In addition, we can observe that in most of the cases the application of the z-score transformation leads to an improvement in the performances of the algorithm.
The best results are obtained with $\pi_T = 0.8$ for most of the applications, with some exceptions like $\pi = 0.2$ (no pre-processing) where we have the best results for $\pi_T = 0.7$.

Another possible approach is to use the polynomial SVM algorithm, which allows to find non-linear separation surfaces through the optimization of a polynomial of degree d exploiting the concept of kernel.
Here we report the same analysis on $DCF_{\min}$ in function of C for $d = 2$ and $d = 3$.

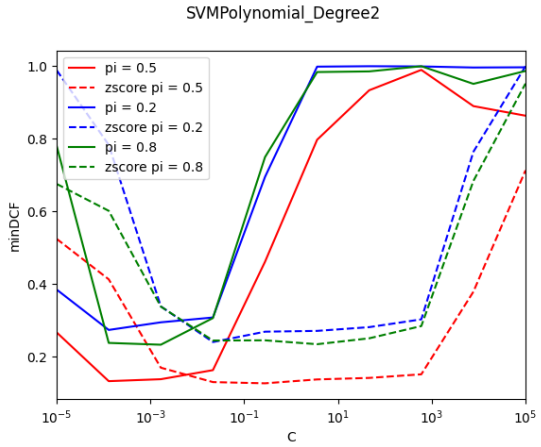

Figure 30 – Polynomial SVM (d=2), Variation of DCF_{min} in function of C

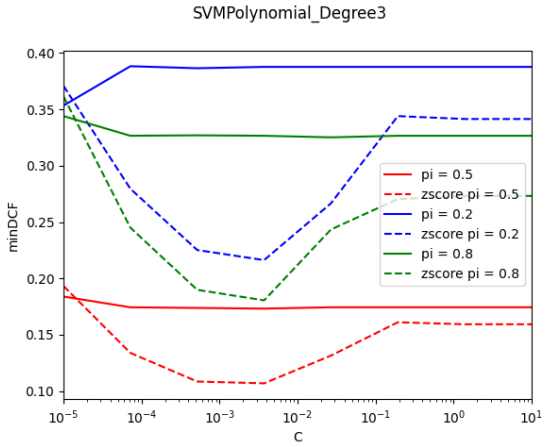

Figure 31 – Polynomial SVM (d=3), Variation of DCF_{min} in function of C

Being the results similar for both values of d, we continue our analysis considering only d=2 because it is less computationally expensive.
Having seen the plot, we select $C = 10^{-4}$ for raw data and $C = 10^{-1}$ and for the data pre-processed with z-score.
The performed measurements are reported below.

		RAW($C = 10^{-4}$)			**ZSCORE**($C = 10^{-1}$)		
Model	$\boldsymbol{\pi_T}$	$\boldsymbol{\pi = 0.5}$	$\boldsymbol{\pi = 0.2}$	$\boldsymbol{\pi = 0.8}$	$\boldsymbol{\pi = 0.5}$	$\boldsymbol{\pi = 0.2}$	$\boldsymbol{\pi = 0.8}$
SVMPolynomial (d = 2)	0.2	0.144	0.261	0.295	0.141	0.280	0.275
SVMPolynomial (d = 2)	0.3	0.138	0.267	0.285	0.132	0.244	0.249
SVMPolynomial (d = 2)	0.4	0.137	0.270	0.251	0.128	0.246	0.251
SVMPolynomial (d = 2)	0.5	0.133	0.262	0.236	0.126	0.245	0.242
SVMPolynomial (d = 2)	0.6	0.130	0.262	0.224	0.126	0.246	0.222
SVMPolynomial (d = 2)	0.7	0.132	0.263	0.218	0.128	0.255	0.229
SVMPolynomial (d = 2)	0.8	0.137	0.276	0.228	0.131	0.240	0.219

We can see that also for polynomial SVM we have better results if we apply z-score.
Again, we can generally see good results, but overall linear SVM performs better than polynomial SVM.
Finally, the last approach we try is **Radial Based Function SVM**, a version of the SVM exploiting the RBF kernel. This kernel depends on the γ parameter, which is a measure of how much support vectors (i.e., the closest points to the margin) influence other points.
We now plot on a graph how $DCF_{\min}$ varies in function of C for various values of γ.

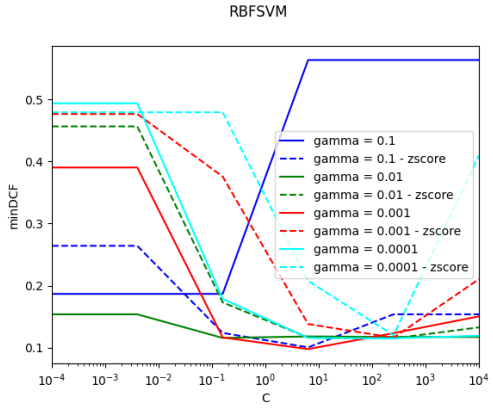

Figure 32 – RBF SVM, Variation of DCF_{min} in function of C

From the graph, we choose $C = 10, \gamma = 0.001$ for raw data and $C = 10, \gamma = 0.1$ for the data pre-processed with z-score.

The performed measurements are reported below.

		RAW($C = 10, \gamma = 0.001$)			**ZSCORE**($C = 10, \gamma = 0.1$)		
Model	$\boldsymbol{\pi_T}$	$\boldsymbol{\pi = 0.5}$	$\boldsymbol{\pi = 0.2}$	$\boldsymbol{\pi = 0.8}$	$\boldsymbol{\pi = 0.5}$	$\boldsymbol{\pi = 0.2}$	$\boldsymbol{\pi = 0.8}$
RBFSVM	0.2	0.105	0.197	0.211	0.105	0.196	0.216
RBFSVM	0.3	0.101	0.192	0.201	0.101	0.203	0.211
RBFSVM	0.4	0.096	0.187	0.192	0.102	0.202	0.201
RBFSVM	0.5	0.094	0.180	0.180	0.100	0.194	0.190
RBFSVM	0.6	0.090	0.179	0.176	0.100	0.206	0.179
RBFSVM	0.7	0.089	0.173	0.172	0.104	0.216	0.178
RBFSVM	0.8	0.096	0.197	0.175	0.103	0.231	0.183

We can notice that, even though z-score does not bring any improvement in terms of performances, these are the best results among the one we obtained for SVM algorithms.
Thus, the one we select as best model is RBF SVM ($\boldsymbol{C = 10,\ \gamma = 10^{-3}, \pi_T = 0.7}$).

Gaussian Mixture Models

Finally, we analyze GMM (**Gaussian Mixture Models**) classifiers.
Since data seems to be well described by Gaussian densities and we also know that the embeddings come from people belonging to 3 age groups, we expect that GMM Classifiers with 2/4 components will be the ones that will perform better.
Furthermore, considering the previous analysis of Gaussian classifiers, we can expect that the tied and the standard versions of the model will perform better and that the diagonal ones will be improved if PCA is applied.
Firstly, we try to understand if the first assumption can be confirmed applying cross-validation in order to estimate the number of components.

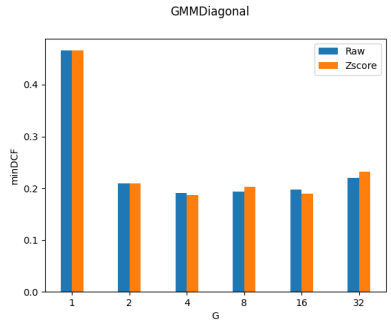

Figure 33 – Diagonal GMM on raw data

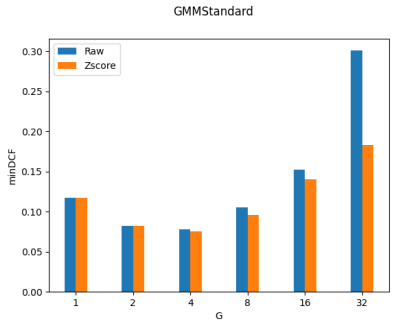

Figure 34 – Standard GMM on raw Data

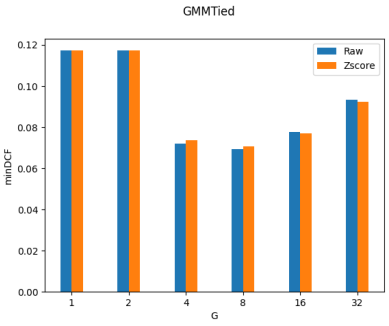

Figure 35 – Tied GMM on raw data

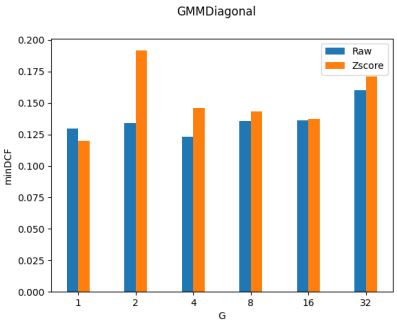

Figure 36 – Diagonal GMM on raw data + PCA(12)

As we can see from the histograms reported above, in general the GMM with 4 components are the ones that perform better.

We applied PCA with $m = 12$ on Diagonal GMM because we knew that it could improve the model performances because the algorithm assumes that the covariance matrix is diagonal.
After this initial phase we can see that the best value for the number of components is probably 4 and that the best performing models are probably the Standard and the Tied GMMs.
Being the dataset unbalanced, it may be useful to try different K value combinations for the classes.
In particular, we will try to combine $K = 2 - 4$ for Standard GMM and $K = 4 - 8$ for Tied GMM, that seem to be the most promising values given the previous analysis.
The performed measurements are reported below.

			RAW			ZSCORE		
Model	**K0**	**K1**	$\boldsymbol{\pi = 0.5}$	$\boldsymbol{\pi = 0.2}$	$\boldsymbol{\pi = 0.8}$	$\boldsymbol{\pi = 0.5}$	$\boldsymbol{\pi = 0.2}$	$\boldsymbol{\pi = 0.8}$
GMMStandard	2	2	0.082	0.180	0.130	0.082	0.180	0.129
GMMStandard	2	4	0.076	0.170	0.142	0.076	0.170	0.139
GMMStandard	4	2	0.100	0.206	0.156	0.098	0.207	0.158
GMMStandard	4	4	0.077	0.159	0.135	0.075	0.157	0.132
GMMTied	4	4	0.072	0.149	0.141	0.073	0.154	0.144
GMMTied	4	8	0.070	0.137	0.126	0.072	0.139	0.135
GMMTied	8	4	0.064	0.161	0.134	0.066	0.163	0.136
GMMTied	8	8	0.069	0.157	0.126	0.070	0.154	0.136

As we can see from the table, for Standard GMM the best results come from $K_0 = 2$, $K_1 = 4$ and $K_0 = K_1 = 4$.
For Tied GMM, instead, the best model is described by $K_0 = 8$, $K_1 = 4$.
We continue the validation for PCA m values with Standard GMM $K_0 = K_1 = 4$, Tied GMM $K_0 = 8, K_1 = 4$ and Tied GMM $K_0 = K_1 = 4$.
We also choose Tied GMM $K_0 = K_1 = 4$ because the version with $K_0 = 8, K_1 = 4$ could overfit the training dataset that has few samples for class 0.
We discarded the model Standard GMM $K_0 = 2, K_1 = 4$ because there is a slight difference with the other Standard GMM model and $K = 4$ could better describe the data considering the natural division into the 3 age groups of the dataset.
Hereafter we will focus on PCA dimensionality reduction validation on the best models.
We also included the Diagonal GMM since PCA should help it to perform better, as we have seen in figure 36.
The performed measurements are reported below.

			PCA(12)			PCA(11)			PCA(10)		
Model	$\mathbf{K_0}$	$\mathbf{K_1}$	$\boldsymbol{\pi = 0.5}$	$\boldsymbol{\pi = 0.2}$	$\boldsymbol{\pi = 0.8}$	$\boldsymbol{\pi = 0.5}$	$\boldsymbol{\pi = 0.2}$	$\boldsymbol{\pi = 0.8}$	$\boldsymbol{\pi = 0.5}$	$\boldsymbol{\pi = 0.2}$	$\boldsymbol{\pi = 0.8}$
GMMStandard	4	4	0.078	0.160	0.135	0.074	0.155	0.120	0.101	0.198	0.183
GMMDiagonal	4	4	0.123	0.244	0.215	0.127	0.266	0.219	0.204	0.379	0.377
GMMTied	4	4	0.072	0.150	0.141	0.073	0.139	0.143	0.093	0.177	0.188
GMMTied	8	4	0.064	0.162	0.135	0.075	0.151	0.145	0.103	0.202	0.205

We can observe that PCA does not actually improve the performances of the models.
We finally choose Tied **GMM ($\boldsymbol{K = 4}$)** because it performs better than Standard GMM and because a K value closer to 3 (3 age groups) could be a better choice for the performance on the test set.

Score Calibration

We now analyse the score calibration of the best models.
We start considering DCF_{min} and DCF_{act} of the best models respect to the target application ($\pi_T = 0.5,\ C_{ft} = 1, C_{fn} = 1$).
The performed measurements are reported below.

Model	$\boldsymbol{minDCF(\pi = 0.5)}$	$\boldsymbol{actDCF(\pi = 0.5)}$
TMVG(zscore)	0.116	0.121
LinearLogReg($\boldsymbol{\lambda = 10^{-3}, \pi_T = 0.7}$)	0.117	0.126
RBF SVM ($\boldsymbol{C = 10,\ \gamma = 10^{-3}, \pi_T = 0.7}$)	0.089	0.106
GMMTied K - 4	0.072	0.075

As we can observe, Tied GMM and Tied MVG are already well calibrated on the target application.
On the other hand, RBF SVM is poorly calibrated, and the Linear Logistic Regression model may need score calibration as well.
We can consult Bayes Error Plots to have a clearer idea on the performances with other working points.

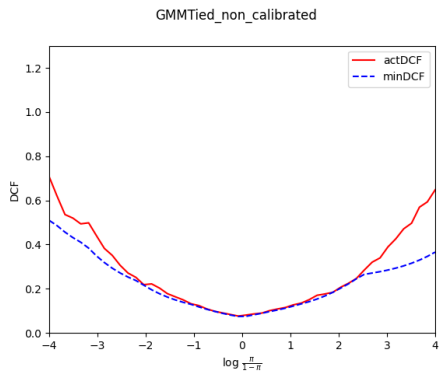

Figure 37 – Tied GMM, Bayes Error plot, non-calibrated data

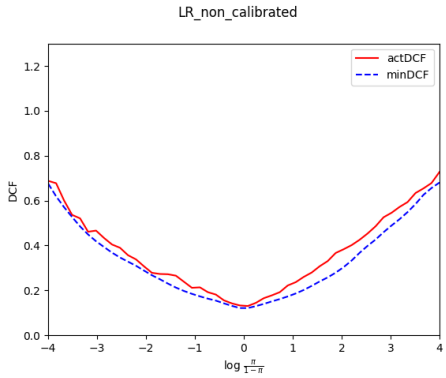

Figure 38 – Linear Logistic Regression, Bayes Error plot, non-calibrated data

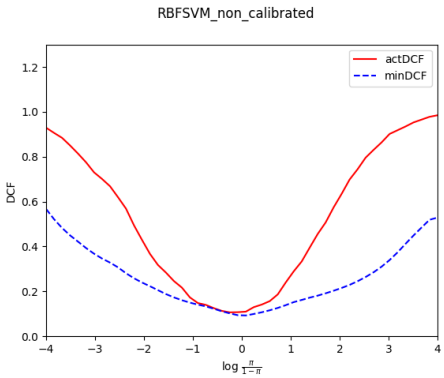

Figure 39 – RBF SVM, Bayes Error plot, non-calibrated data

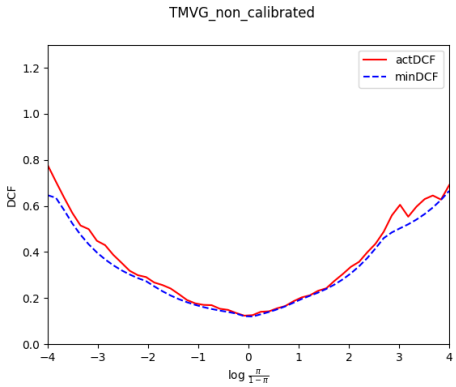

Figure 40 – Tied MVG, Bayes Error plot, non-calibrated data

Having seen the plots, we have an ulterior confirmation that RBF SVM and Linear Logistic Regression are the models that most need score calibration.
On the contrary, for Tied GMM and Tied MVG, DCF_{min} and DCF_{act} almost overlap in some sections.

As already said, we are considering the working point ($\pi_T = 0.5,\ C_{ft} = 1, C_{fn} = 1$) as target application but, since we are also considering other working points and training prior validation has already proved to be useful, we will analyse different priors for the Logistic Regression calibration step.
We use again a K-Fold approach reporting both minimum and actual costs.
The performed measurements are reported below.

Model	Prior	DCF_{min} (calibrated scores) - $\pi = 0.5$	DCF_{act} - $\pi = 0.5$
TMVG	0.2	0.117	0.120
TMVG	0.3	0.117	0.120
TMVG	0.5	0.117	0.121
TMVG	0.7	0.118	0.120
TMVG	0.8	0.118	0.120
RBF SVM	0.2	0.089	0.091
RBF SVM	0.3	0.089	0.092
RBF SVM	0.5	0.090	0.094
RBF SVM	0.7	0.089	0.094
RBF SVM	0.8	0.089	0.094
LR	0.2	0.118	0.124
LR	0.3	0.117	0.123
LR	0.5	0.117	0.124
LR	0.7	0.117	0.124
LR	0.8	0.117	0.124
GMMTied	0.2	0.073	0.076
GMMTied	0.3	0.073	0.075
GMMTied	0.5	0.073	0.074
GMMTied	0.7	0.073	0.075
GMMTied	0.8	0.074	0.075

For the target application, changing the prior does not seem to be much effective.
We finally choose 0.3 as value of prior since it seems the most suitable from the table above.
We now analyse the impact of calibration on other working points through Bayes Error plots.

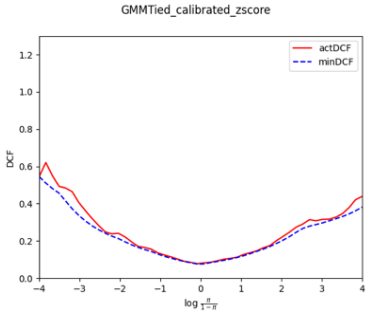

Figure 41 – Tied GMM, Bayes Error plot, calibrated data

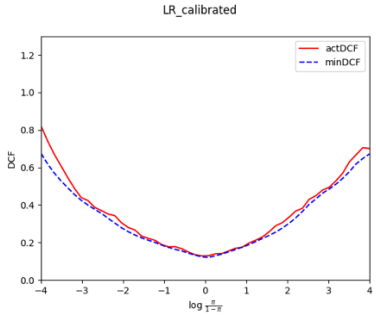

Figure 42 – Linear Logistic Regression, Bayes Error plot, calibrated data

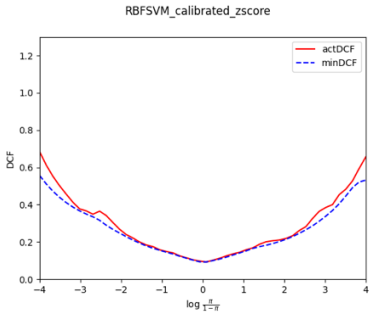

Figure 43 – RBF SVM, Bayes Error plot, calibrated data

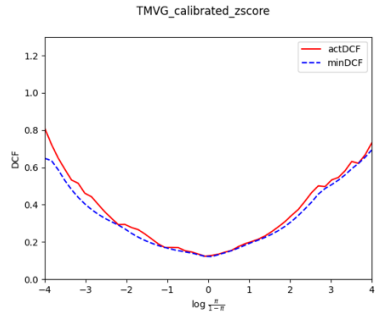

Figure 44 – Tied MVG, Bayes Error plot, calibrated data

In general, we can state that score calibration is useful, but for Tied MVG it does not bring any significant improvement.
On the other hands, it really improves the performances of RBF SVM, that was poorly calibrated.
We can now see how the score calibration performs on the other working points.
The performed measurements are reported below.

Model	Prior	$\mathbf{DCF_{min}}$ $\boldsymbol{\pi = 0.5}$	$\mathbf{DCF_{act}}$ $\boldsymbol{\pi = 0.5}$	$\mathbf{DCF_{min}}$ $\boldsymbol{\pi = 0.2}$	$\mathbf{DCF_{act}}$ $\boldsymbol{\pi = 0.2}$	$\mathbf{DCF_{min}}$ $\boldsymbol{\pi = 0.8}$	$\mathbf{DCF_{act}}$ $\boldsymbol{\pi = 0.8}$
TMVG	0.3	0.118	0.121	0.193	0.217	0.224	0.232
RBF SVM	0.3	0.089	0.092	0.179	0.184	0.175	0.187
LR	0.3	0.117	0.123	0.212	0.222	0.221	0.229
GMMTied	0.3	0.073	0.074	0.154	0.165	0.144	0.147

As we can see, the results are in general acceptable also for the other working points.

Fusion Models

Now we focus our attention on the fusion of the best models.
We stack the output of the models in order to create a sort of "training dataset" and then we use the prior-weighted Logistic Regression, as we have done for score calibration.
We choose as training prior $\pi = 0.3$, considering the results of the score calibration step.
Moreover, we only fuse RBF SVM, LR and Tied GMM, since Tied MVG makes similar assumptions but has worse performances if compared to Tied GMM.
Thus, we expect that the models fused with Tied GMM will have the best results.
Starting from the previous results, we have:

Model	Prior	$\mathbf{DCF_{min}}$ $\boldsymbol{\pi = 0.5}$	$\mathbf{DCF_{act}}$ $\boldsymbol{\pi = 0.5}$	$\mathbf{DCF_{min}}$ $\boldsymbol{\pi = 0.2}$	$\mathbf{DCF_{act}}$ $\boldsymbol{\pi = 0.2}$	$\mathbf{DCF_{min}}$ $\boldsymbol{\pi = 0.8}$	$\mathbf{DCF_{act}}$ $\boldsymbol{\pi = 0.8}$
TMVG	0.3	0.118	0.121	0.193	0.217	0.224	0.232
RBF SVM	0.3	0.089	0.092	0.179	0.184	0.175	0.187
LR	0.3	0.117	0.123	0.212	0.222	0.221	0.229
GMMTied	0.3	0.073	0.074	0.154	0.165	0.144	0.147

Calibrated scores

Model	Prior	$\mathbf{DCF_{min}}$ $\boldsymbol{\pi = 0.5}$	$\mathbf{DCF_{act}}$ $\boldsymbol{\pi = 0.5}$	$\mathbf{DCF_{min}}$ $\boldsymbol{\pi = 0.2}$	$\mathbf{DCF_{act}}$ $\boldsymbol{\pi = 0.2}$	$\mathbf{DCF_{min}}$ $\boldsymbol{\pi = 0.8}$	$\mathbf{DCF_{act}}$ $\boldsymbol{\pi = 0.8}$
RBF SVM + LR	0.3	0.089	0.092	0.173	0.179	0.167	0.177
RBF SVM + GMMTied	0.3	0.070	0.074	0.149	0.150	0.133	0.142
LR + GMMTied	0.3	0.070	0.075	0.151	0.156	0.138	0.138
GMMTied + RBF SVM +LR	0.3	0.069	0.075	0.147	0.150	0.134	0.140

Fusion models scores

We also report here Bayes Error plots.

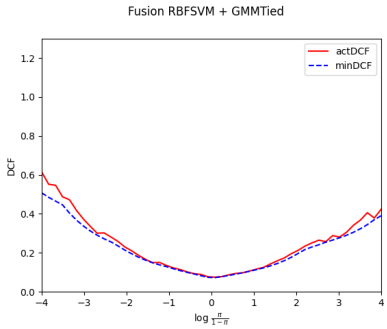

Figure 45 – RBF SVM + Tied GMM, Bayes Error plot

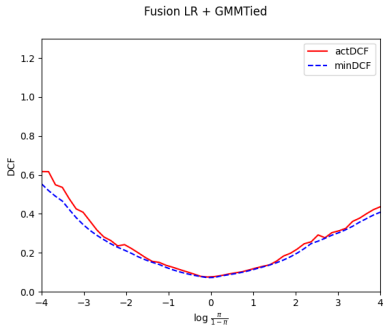

Figure 46 – Linear Logistic Regression + Tied GMM, Bayes Error plot

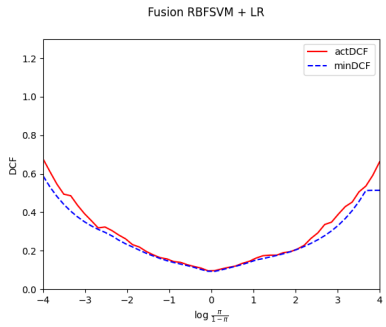

Figure 47 – RBF SVM + Linear Logistic Regression, Bayes Error plot

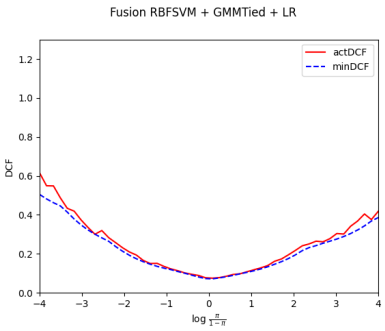

Figure 48 – RBF SVM + Linear Logistic Regression + Tied GMM, Bayes Error plot

Looking at the plots, we can observe that the models are almost well calibrated.
We can see that the fusion models with Tied GMM are the ones that bring better results, as we assumed before, but overall fusion models provide only a slight improvement.
Thus, we choose as candidate model **Tied GMM with calibration**, whose performances are very similar to RBF + Linear Logistic Regression + Tied GMM (which is the best performing fusion model) and to non-calibrated Tied GMM, but it is less computationally expensive than the first and performs better than the latter on a wider range of working points.

Evaluation

In this final phase of our analysis, we focus on the study of how the models which best performed in the validation step behave on the evaluation dataset.
First, we analyse the performances of the best single models.
The performed measurements are reported below.

	Validation-Calibrated		**Evaluation-Calibrated**	
Model	DCF_{min}	DCF_{act}	DCF_{min}	DCF_{act}
TMVG	0.117	0.120	0.116	0.120
RBFSVM	0.089	0.091	0.082	0.083
LR	0.117	0.123	0.120	0.120
GMMTied	0.073	0.074	0.057	0.058

	Validation		Evaluation	
Model	DCF_{min}	DCF_{act}	DCF_{min}	DCF_{act}
TMVG	0.116	0.121	0.116	0.117
RBFSVM	0.089	0.106	0.082	0.095
LR	0.117	0.126	0.120	0.134
GMMTied	0.072	0.075	0.057	0.060

As we can observe, although for some of the models we obtain slightly different results between validation and evaluation, Tied GMM is once again the best performing one.
We also report here the Bayes Error plots with calibrated scores.

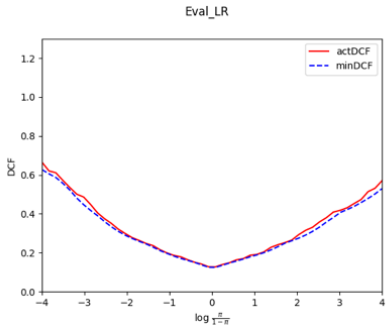

Figure 49 – Evaluation for LLR, calibrated scores, Bayes Error plot

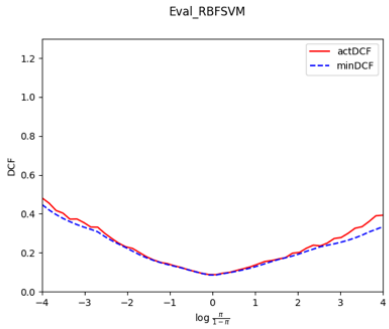

Figure 50 – Evaluation for RBF SVM, calibrated scores, Bayes Error plot

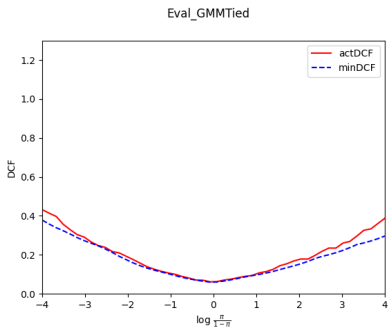

Figure 51 – Evaluation for Tied MVG, calibrated scores, Bayes Error plot

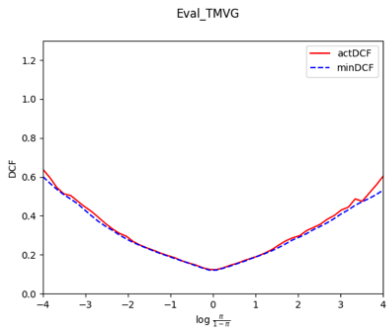

Figure 52 – Evaluation for Tied GMM, calibrated scores, Bayes Error plot

Another useful tool we may employ are det plots, which show for each model how the False Positive Rate varies in function of the False Negative Rate; they are reported below.

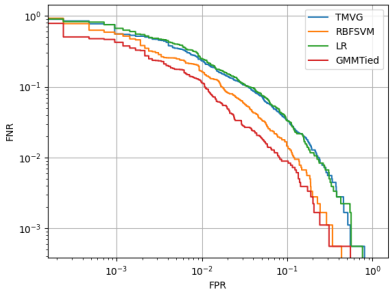

Figure 53 – Det plot, calibrated scores

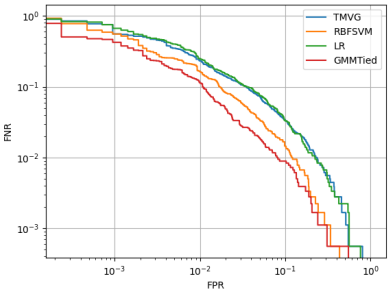

Figure 54 – Det plot, non-calibrated scores

From the det plots, we confirm one more time that the Tied GMM is the best performing model.
Now we analyse the performances of the fusion models, without ulterior calibrations.

The performed measurements are reported below.

	Validation		Evaluation	
Model	DCF_{min}	DCF_{act}	DCF_{min}	DCF_{act}
RBFDVM + LR	0.089	0.091	0.082	0.086
RBFSVM + GMMTied	0.070	0.073	0.058	0.061
LR + GMMTied	0.070	0.074	0.061	0.062
RBFSM + LR + GMMTied	0.069	0.075	0.059	0.061

On the contrary of what we have seen for the single models, here we have some more remarkable differences between validation and evaluation.
Although, we have again that RBF SVM + Tied GMM + Linear Logistic Regression is the best performing fusion model and, like in the validation phase, it has almost the same performances of Tied GMM (but with a higher computational cost).
We have also generated their Bayes Error plots.

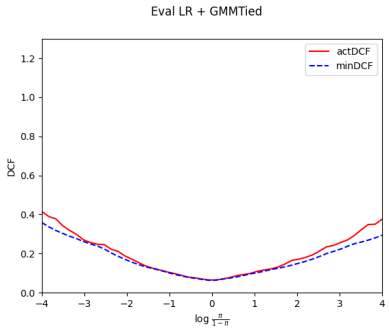

Figure 55 – Evaluation for LLR + Tied GMM Error plot

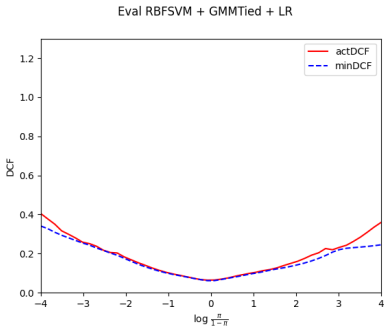

Figure 56 – Evaluation for RBF SVM + Tied GMM + LLR, Bayes Error plot

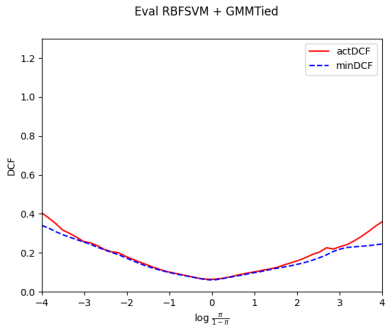

Figure 57 – Evaluation for RBF SVM + Tied GMM, Bayes Error plot

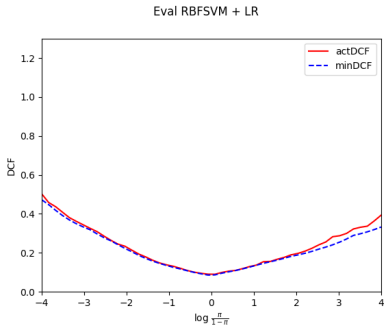

Figure 58 – Evaluation for RBF SVM + LLR, Bayes Error plot

Again, the plots confirm that the score do not need any ulterior calibration process.
We now want to analyse the goodness of the choices made on the hyperparameters for each model during the validation step.
We start with Linear Logistic Regression.
In the validation phase, we chose $\lambda = 10^{-3}$ for Linear Logistic Regression with no z-score, no PCA, $\pi_T = 0.7$.
We analyse the performances of the model for various working points and various values of λ, with and without z-score pre-processing.

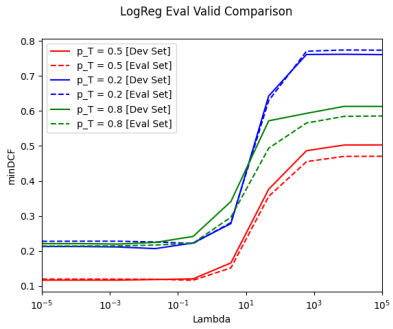

Figure 59 – Evaluation of λ for LLR, no z-score, no PCA

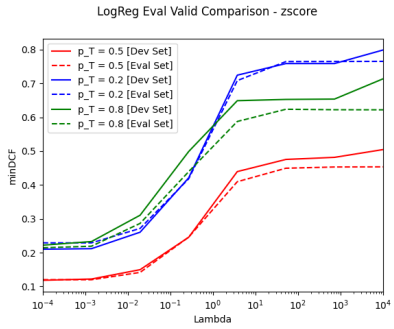

Figure 60 – Evaluation of λ for LLR, with z-score, no PCA

From the plot in figure 59 we can see that a better choice might have been $\lambda = 10^{-0.7} = 0.2$, so we analyse the performances of the same model (Linear Logistic Regression, no z-score, no PCA), but with $\lambda = 0.2$, for various values of π_T.
The performed measurements are reported below.

		RAW($\lambda = 0.2$)			**ZSCORE($\lambda = 10^{-3}$)**		
Model	$\boldsymbol{\pi_T}$	$\boldsymbol{\pi = 0.5}$	$\boldsymbol{\pi = 0.2}$	$\boldsymbol{\pi = 0.8}$	$\boldsymbol{\pi = 0.5}$	$\boldsymbol{\pi = 0.2}$	$\boldsymbol{\pi = 0.8}$
LogReg	0.2	0.118	0.222	0.235	0.119	0.224	0.229
LogReg	0.3	0.117	0.220	0.226	0.117	0.223	0.226
LogReg	0.4	0.117	0.219	0.222	0.116	0.223	0.222
LogReg	0.5	0.118	0.222	0.220	0.117	0.219	0.220
LogReg	0.6	0.118	0.224	0.223	0.117	0.224	0.218
LogReg	0.7	0.117	0.224	0.220	0.119	0.228	0.216
LogReg	0.8	0.118	0.225	0.217	0.120	0.229	0.212

Below are reported the results from the validation step.

		RAW($\lambda = 10^{-3}$)			**ZSCORE($\lambda = 10^{-3}$)**		
Model	$\boldsymbol{\pi_T}$	$\boldsymbol{\pi = 0.5}$	$\boldsymbol{\pi = 0.2}$	$\boldsymbol{\pi = 0.8}$	$\boldsymbol{\pi = 0.5}$	$\boldsymbol{\pi = 0.2}$	$\boldsymbol{\pi = 0.8}$
LogReg	0.2	0.124	0.203	0.235	0.125	0.219	0.244
LogReg	0.3	0.123	0.206	0.228	0.125	0.212	0.237
LogReg	0.4	0.121	0.209	0.227	0.125	0.213	0.234
LogReg	0.5	0.118	0.211	0.221	0.122	0.210	0.232
LogReg	0.6	0.117	0.209	0.222	0.121	0.209	0.228
LogReg	0.7	0.117	0.212	0.220	0.121	0.209	0.229
LogReg	0.8	0.116	0.217	0.220	0.121	0.212	0.231

If we compare those data with the values computed during the validation step, we actually obtain better results for $\lambda = 0.2$, but the best value of π_T varies depending on the considered working point:
$\pi_T = 0.7$ for $\pi = 0.5$, $\pi_T = 0.2$ for $\pi = 0.2$, $\pi_T = 0.7$ for $\pi = 0.8$.
As regards the application of the z-score pre-processing, we can see from the graph in 60 that $\lambda = 10^{-3}$ is actually a good candidate for the best choice, but we have also observed from the table that the application of z-score does not bring any advantage.
We now analyse Quadratic Logistic Regression.
In the validation phase, we chose $\lambda = 10^2$ for Quadratic Logistic Regression, no z-score, no PCA, $\pi_T = 0.7$.

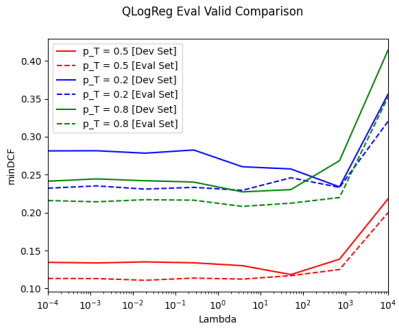

Figure 61 – Evaluation of λ for QLR, no z-score, no PCA

From the plot in figure 61, we can see that a better choice might have been $\lambda = 0.02$, so we analyse the performances of the same model (Quadratic Logistic Regression), but with $\lambda = 0.02$, for various values of PCA m.
The performed measurements are reported below.

		RAW			ZSCORE		
Model	**PCA(m)**	$\pi = 0.5$	$\pi = 0.2$	$\pi = 0.8$	$\pi = 0.5$	$\pi = 0.2$	$\pi = 0.8$
QLR	-	0.112	0.232	0.215	0.112	0.228	0.206
QLR	12	0.113	0.235	0.214	0.112	0.228	0.205
QLR	11	0.116	0.238	0.232	0.120	0.227	0.220
QLR	9	0.181	0.331	0.314	0.197	0.358	0.359
QLR	8	0.184	0.351	0.325	0.246	0.435	0.450

After this evaluation we notice that the best model would have been QLR $\lambda = 0.02$ prior = 0.7 without data pre-processing.
After Logistic Regression, we evaluate the SVM.
First, we focus on the linear version of the model.
We evaluate the selection of C using as training prior 0.8, which is the best value found in the validation phase.

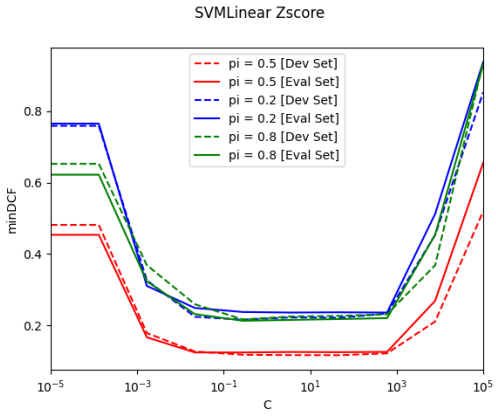

Figure 62 -Evaluation of Linear SVM on z-score pre-processed data

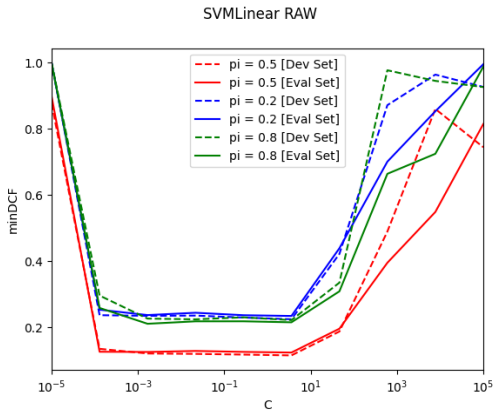

Figure 63 - Evaluation of Linear SVM on raw data

During the validation phase we chose $C = 10$ for z-score version and $C = 10^{-1}$ for raw data and we now can observe that they are also a good choice on the evaluation set.
We do not go further on SVM Linear because the cost seems to be like the one computed in the validation phase.

We also skip Quadratic SVM because it was the worst model among SVMs and it is similar to Quadratic Logistic Regression, that we already analyzed.
We think that it is better to focus our attention on RBF SVM, because during the validation it was the best model among SVMs.

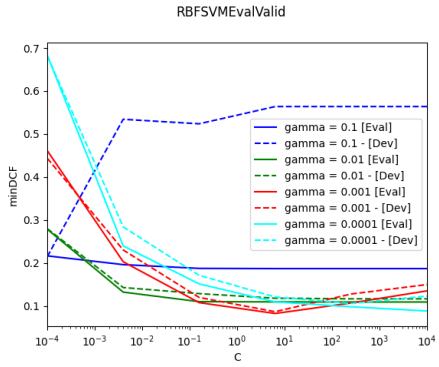

Figure 64 – Evaluation of RBF SVM on raw data

We specify that the models were trained with $\pi_T = 0.7$ and on raw data.
We can see that the minimum cost is the same of validation phase.
For completeness, we also consider the most promising values of γ found during the validation with z-score applied in the most promising C range.

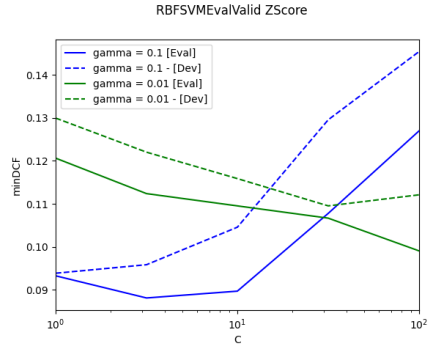

Figure 65 Figure 64 – Evaluation of RBF SVM on z-score pre-processed data

We can see that also with z-score the behavior is similar and in general set the system performs slightly better on the evaluation.
After the evaluation of the C hyperparameter, we can evaluate the choice of the prior.

		RAW($C = 10, \gamma = 0.001$)			**ZSCORE**($C = 3, \gamma = 0.1$)		
Model	$\boldsymbol{\pi_T}$	$\boldsymbol{\pi = 0.5}$	$\boldsymbol{\pi = 0.2}$	$\boldsymbol{\pi = 0.8}$	$\boldsymbol{\pi = 0.5}$	$\boldsymbol{\pi = 0.2}$	$\boldsymbol{\pi = 0.8}$
RBFSVM	0.2	0.091	0.165	0.190	0.093	0.162	0.202
RBFSVM	0.3	0.084	0.153	0.174	0.088	0.163	0.169
RBFSVM	0.4	0.081	0.151	0.165	0.085	0.152	0.201
RBFSVM	0.5	0.080	0.148	0.161	0.083	0.151	0.163
RBFSVM	0.6	0.082	0.155	0.156	0.085	0.156	0.157
RBFSVM	0.7	0.082	0.160	0.154	0.088	0.164	0.161
RBFSVM	0.8	0.086	0.179	0.145	0.093	0.183	0.160

The results from the validation step are reported below.

		RAW($C = 10, \gamma = 0.001$)			**ZSCORE**($C = 10, \gamma = 0.1$)		
Model	$\boldsymbol{\pi_T}$	$\boldsymbol{\pi = 0.5}$	$\boldsymbol{\pi = 0.2}$	$\boldsymbol{\pi = 0.8}$	$\boldsymbol{\pi = 0.5}$	$\boldsymbol{\pi = 0.2}$	$\boldsymbol{\pi = 0.8}$
RBFSVM	0.2	0.105	0.197	0.211	0.105	0.196	0.216

RBFSVM	0.3	0.101	0.192	0.201	0.101	0.203	0.211
RBFSVM	0.4	0.096	0.187	0.192	0.102	0.202	0.201
RBFSVM	0.5	0.094	0.180	0.180	0.100	0.194	0.190
RBFSVM	0.6	0.090	0.179	0.176	0.100	0.206	0.179
RBFSVM	0.7	0.089	0.173	0.172	0.104	0.216	0.178
RBFSVM	0.8	0.096	0.197	0.175	0.103	0.231	0.183

We can observe that our choice was suboptimal, and the best system would have been RBF SVM with $C = 10, \gamma = 0.001$, prior $= 0.5$, but the improvement is negligible respect to our chosen RBF SVM, which had the same configuration but with prior$= 0.7$.
We can further analyze the impact of PCA over this system.

Model	**PCA(m)**	$\boldsymbol{\pi = 0.5}$	$\boldsymbol{\pi = 0.2}$	$\boldsymbol{\pi = 0.8}$
RBFSVM	-	0.080	0.148	0.161
RBFSVM	11	0.084	0.153	0.156
RBFSVM	9	0.126	0.230	0.224
RBFSVM	8	0.128	0.230	0.230

PCA, link in the validation phase, does not improve the performances.
Finally, the best system would have been RBFSVM $C = 10$, $\gamma = 0.001$, prior $= 0.5$, but the difference with our choice is negligible, as we have stated before.

In conclusion we analyse GMMs on the evaluation set.
By analysing also the 1 component GMMs we incorporate the Gaussian classifiers analysis.
As we have done before, we first focus on the hyperparameters validation.
In this case we try different component values.
Our main focus is on Tied GMM and standard GMM but we compute also the costs for Diagonal GMM with PCA(12) and then we will consider if it is worth to include it in following analyses.

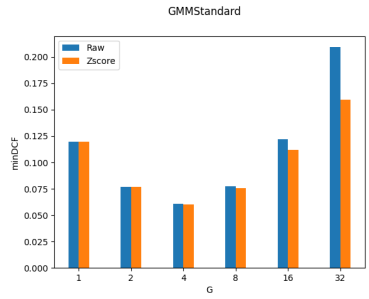

Figure 66 – Evaluation for Standard GMM

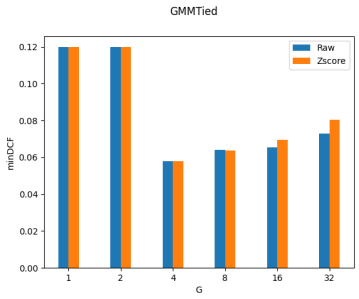

Figure 67 – Evaluation for Tied GMM

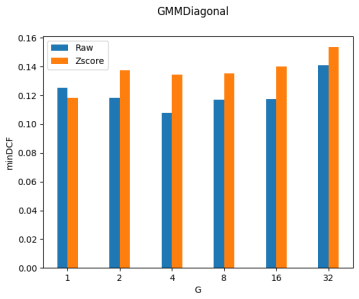

Figure 68 - Evaluation for Diagonal GMM

As we can see, these plots mainly follow the same trend of the plots made for validation.
For this reason, we finally analyze only the models that had the best performances during validation in order to evaluate the goodness of our final choice, Tied GMM with $K = 4$.
We compare the costs on the evaluation set of Standard GMM with $K = 4$, Tied GMM with $K = 4$, Tied GMM with $K_0 = 8/K_1 = 4$ and Standard GMM with $K_0 = 2/K_1 = 4$.
The computed measurements are reported below.

			Validation			Evaluation		
Model	**K0**	**K1**	$\boldsymbol{\pi = 0.5}$	$\boldsymbol{\pi = 0.2}$	$\boldsymbol{\pi = 0.8}$	$\boldsymbol{\pi = 0.5}$	$\boldsymbol{\pi = 0.2}$	$\boldsymbol{\pi = 0.8}$
GMMStandard	2	2	0.082	0.180	0.130	0.061	0.114	0.111
GMMStandard	2	4	0.076	0.170	0.142	0.074	0.135	0.141
GMMTied	4	4	0.072	0.149	0.141	0.057	0.121	0.114
GMMTied	8	4	0.064	0.161	0.134	0.060	0.122	0.109

Evaluation results

The best model is Tied GMM with $K = 4$.
As we have pointed out for the validation phase, probably $K = 4$ better adapts to the real distribution of data (3 age groups) and for this reason it performs good, also on the other working points analyzed.
We can also consider the calibrated versions of the best models found on evaluation set and the corresponding fusion models.

Model	$DCF_{min}(\boldsymbol{\pi = 0.5})$	$DCF_{act}(\boldsymbol{\pi = 0.5})$
RBFSVM($\boldsymbol{C = 10,\ \gamma = 10^{-3}, \pi_T = 0.5}$)	0.080	0.080
QLR ($\lambda = 0.02, \boldsymbol{\pi_T = 0.7}$)	0.112	0.118
TiedGMM K - 4	0.057	0.0.58
RBFSVM + QLR	0.079	0.080
RBFSVM + TiedGMM	0.060	0.061
QLR + TiedGMM	0.059	0.060
RBFSVM + TiedGMM + QLR	0.059	0.060

We can see that the results are almost equal to the ones of our choices made during validation.
This means that our approach was effective and our chosen model, **Tied GMM with $\boldsymbol{K = 4}$**, has a cost on our target application of 0.058, lower than the expected one, and costs of 0.121 and 0.114 respectively for the working points 0.2 and 0.8.