Politecnico di Torino

Classification of Gender Identification dataset

Machine Learning and Pattern Recognition

Davide Ariotto
S302489@studenti.polito.it

Mohamed Khaled Hassan Aly Motrash
S295805@studenti.polito.it

July 03, 2023

Abstract

In this paper we will illustrate the variable outcomes from different Machine Learning (ML) Algorithms using database extracted from high-level features representing face images. The results from several (ML) Algorithms are to be compared and analyzed to reach the best conclusion of the best model performance. The data base contains samples of male and female. It is imbalanced with significantly more female samples as will be explored and further analyze it. Afterwards we will apply different classification algorithms and finally calibrate the scores. Ultimately, the result will show which model is a stronger candidate for our problem.

1. Introduction

The dataset consists of image embeddings which are low-dimensional representation of face images. As images dimensionality is reduces significantly to obtain a more tractable model and to keep the computational effort more reasonable. The embeddings have significantly lower dimensions than in real use-cases.
The embeddings are then reduced to a dimension of 12 of a continuous-valued vectors where the data is basically divided to (2 Classes) “Male” (label 0) and “Female” (label 1) classes.
The embeddings don’t have a physical interpretation as they represent reduced image dimensions.
The Database are imbalanced, with the training set contains more female samples than the males, however in the evaluation set we could see the other way around as it contains more male samples.
We will be analyzing a total of 2,400 samples on the training set separated between the 2 classes such that for the female class it has 1680 sample and 720 for the male class.
The evaluation set also is composed of the same number of embeddings (12) such as the training set.
Each record in our training or testing set is composed of 12 features separated by a comma and in the last column the label of that record. Each sample could belong to 3 different age groups, however that age information is not available.
Several Machine Learning Classification Algorithms will be applied with different learning parameters to be optimized and further analyzed.
The data shall be analyzed without any further reduction in dimensionality and with the intervention of pre-processing techniques.

2. Feature Analysis

Each sample is described by 12 continuous-valued features and a single class variable/label. The features do not have a physical interpretation. In figure 1 we can see the distributions of the features. The orange ones refer to the Female class while the blue ones to the Male class. We noticed that the values are quite high and for this reason we decided to apply Z-normalization to the dataset. This operation transforms the dataset such that the mean of all the values is 0 and the standard deviation is 1 (centering and scaling to unit variance in technical terms).

$$z_i = \frac{x_i - \mu}{\sigma} \quad \forall i \in \{1, \dots, n\}$$

Then we can see that the distributions of a couple features are slightly irregular but there isn't a large presence of outliers. For this reason, we decided that "Gaussianizing" the features is not necessary for this task.

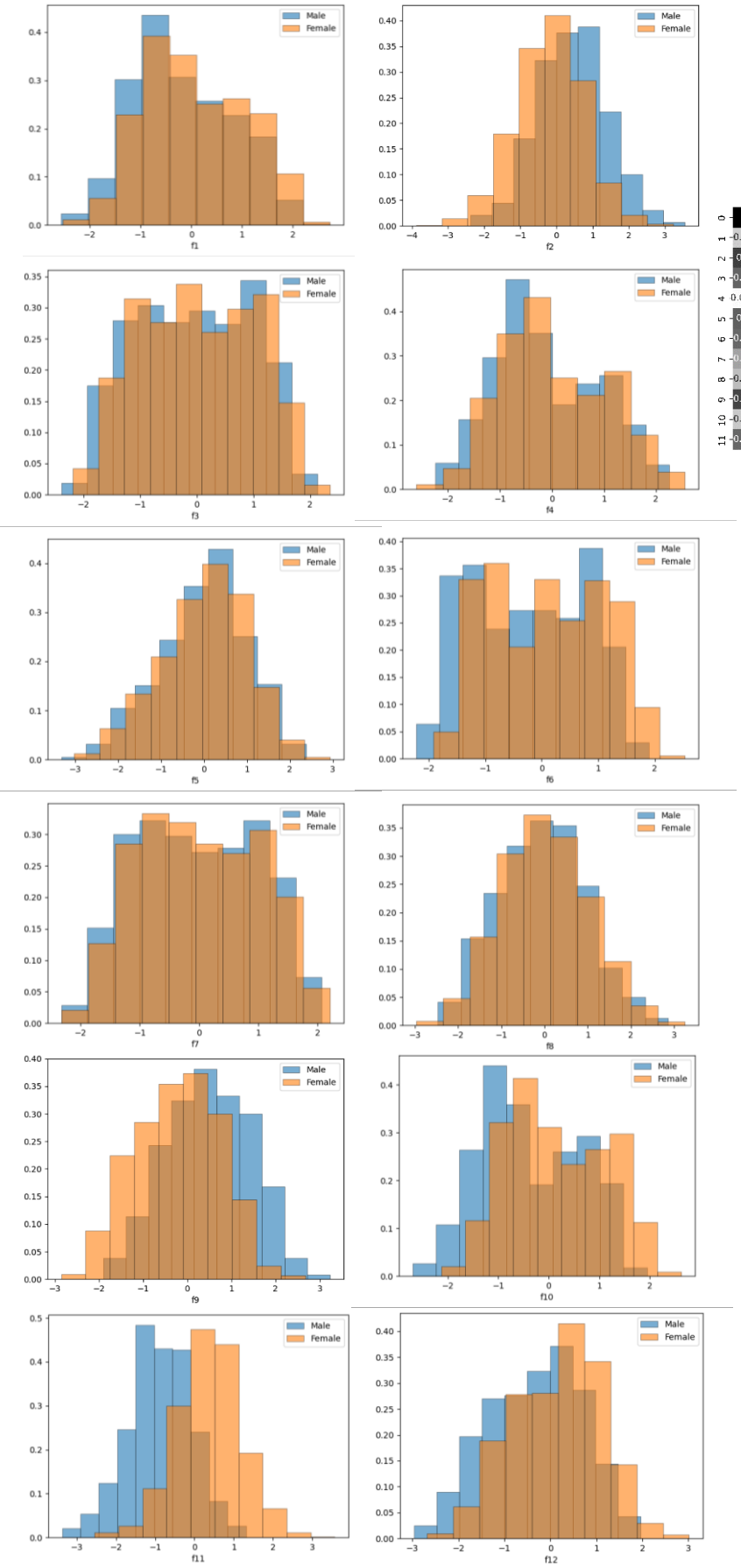


Figure 1: Histograms of the z-normalized Features

"All the original plots and also the scatter one is available in the uploaded folder."

Now let's analyze the correlation between features. Figure 2 contains the heatmaps with the Pearson coefficient.

$$\frac{Cov(X,Y)}{\sqrt{Var(X)}\sqrt{Var(Y)}}$$

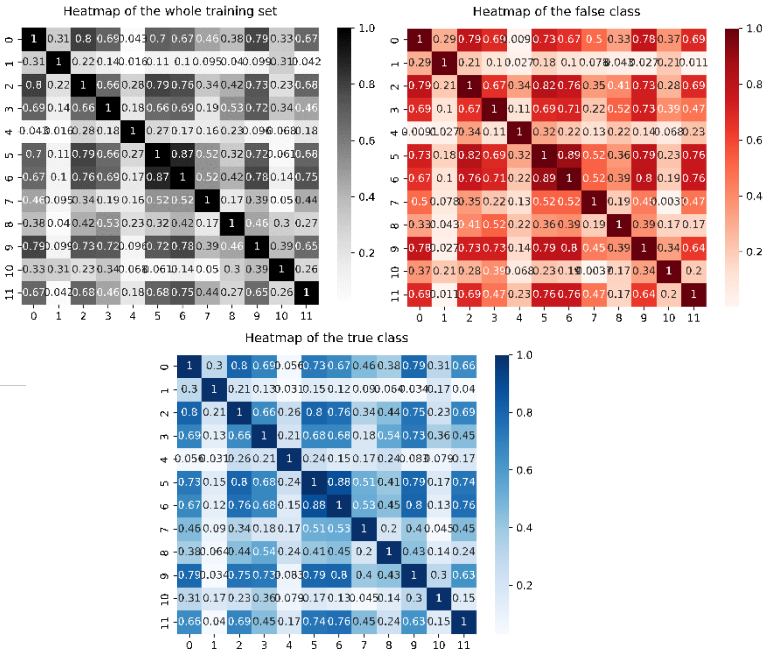


Figure 2: Heatmaps with Pearson correlation coefficient of positive class (blue), negative class (red) and whole training set (grey)

Darker color (and higher values, the maximum is 1) implies a stronger correlation, while lighter color the opposite. Firstly, we can see that the heatmaps aren't that different and this means that the correlation between features doesn't depend on the class. Secondly, we notice that features 5-6 are strongly correlated (coefficient = 0.87) and a few other couples have a coefficient higher than 0.7. This means that we could apply PCA: it reduces the original space with n dimension into a m subspace with m << n, without losing many information and speeding the overall computations. We decided to look for the m value that retains at least 95% of the variance of the data. The results are m = 8. However, m = 10 retains more than 98% of the variance, so we decided to consider three configurations from this point: noPCA, PCA with m = 10 and PCA with m = 8.

3. Classifying the features

To perform the classification task, we can use two approaches: single fold and k-fold. In the first one we basically split the original data (the training set) into validation data (33,3 %) and training data (66,6 %). This approach is faster, but the model has less data to use. K-folds instead consists of dividing the data into "k" folds (we chose k = 5), training the model on k-1 of them and evaluating it on the remaining one. This process is repeated k times, with each fold being used as the test set once. The model may not be optimal because it's not trained over the whole training set, but it has more data available compared to the single fold approach. To tackle the first issue, we'll get the final classifier by re-training over the whole training set. We will consider three different applications: a uniform prior application and two unbalanced applications where the prior is biased towards one of the two classes.

$$(\tilde{\pi}, C_{fp}, C_{fp}) = (\ 0.5,\ 1,\ 1\ )$$
$$(\tilde{\pi}, C_{fp}, C_{fp}) = (\ 0.1,\ 1,\ 1\ )$$
$$(\tilde{\pi}, C_{fp}, C_{fp}) = (\ 0.9,\ 1,\ 1\ )$$

In order to find the best approach, we will measure performances through the normalized minimum Detection Cost Function (min DCF), which measures the cost that we would pay if we made optimal decisions using the recognizer scores. The evaluation is carried out on the validation subset (extracted from the training set).

3.1 Gaussian Classifier

The first classifiers that have been used are the Gaussian classifiers: multivariate gaussian (MVG) classifier, MVG classifier with Naive Bayes assumption and MVG classifier with tied covariance. All of them assume gaussian distributed data:

$$X \sim N(\mu, \Sigma)$$

where μ is the mean and Σ is the covariance matrix. Now we measure the min DCF for the three of them:

	Single Fold			5-fold		
	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$
Z-Normalized Features – no PCA						
Full-Cov	0.092	0.240	0.270	0.114	0.297	0.350
Diag-Cov	0.420	0.729	0.785	0.463	0.771	0.778
Tied Full-Cov	0.082	0.222	0.221	0.109	0.299	0.342
Z- Normalized Features – PCA (m = 10)						
Full-Cov	0.172	0.314	0.477	0.187	0.407	0.538
Diag-Cov	0.173	0.320	0.464	0.184	0.436	0.546
Tied Full-Cov	0.163	0.319	0.491	0.183	0.428	0.535
Z-Normalized Features – PCA (m = 8)						
Full-Cov	0.254	0.481	0.652	0.267	0.549	0.677
Diag-Cov	0.239	0.503	0.651	0.264	0.520	0.673
Tied Full-Cov	0.247	0.451	0.661	0.261	0.508	0.667

Figure 3: min DCF on the validation set for MVG classifiers.

In Figure 3 we can see the min DCF computed on the validation set with different prior probabilities and with the configurations of PCA specified previously. We can notice that PCA degraded performances because the min DCF increased in all cases, except for the diagonal covariance where we the behavior was curious: minDCF decreased from noPCA to PCA with m = 10 but then increased again in the PCA with m = 8 configuration. This means that for the diag-cov, the assumption of independent components may be correct, but we have to be careful, because reducing too much the number of dimensions led us to a loss of discriminant information. However, the tied full-cov classifier obtained the best results and we notice that dimensionality reduction degraded them by a significant factor, so the best configuration at this point is the noPCA. Finally, we could see that the unbalanced tasks leaded to worse results for all models with respect to the balanced one. All the considerations above are valid both for single-fold and k-fold. From this analysis we understand that linear models performed better than quadratic ones.

3.2 Logistic Regression

We start considering regularized linear logistic regression. Since classes are unbalanced, we change the objective function that we need to minimize so that costs of the different classes are re-balanced:

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

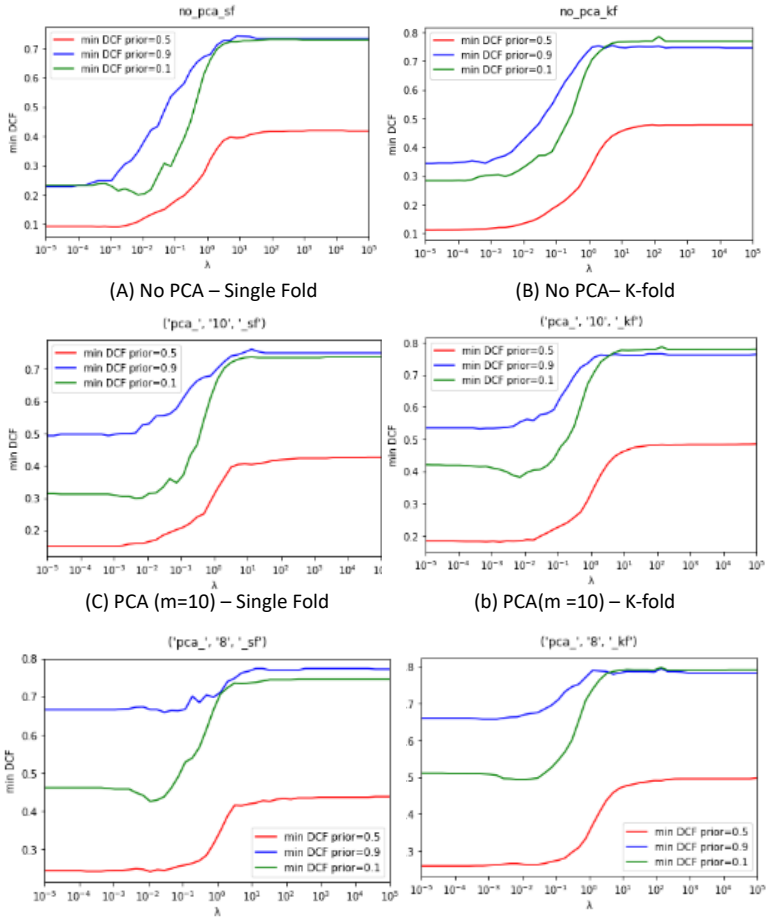
The model parameters are w and b. The decision rules are assumed to be linear hyperplanes orthogonal to w:

$$w^T x + b$$

In the following figures we will attempt to make the model to learn the model parameters which are 'w' and 'b'. we look for λ such that if λ is too small, we will get a solution with a higher norm but poor classification accuracy for test/unseen data.
On the other hand, if we have high value of λ then we will obtain a solution that has a small norm but won't be able to separate well the classes.

We'll look for the best value of the parameter with single-fold, k-fold, no PCA and PCA with m=8 and m=10.

The results are summarized here:



(A) No PCA – Single Fold (B) No PCA– K-fold

(C) PCA (m=10) – Single Fold (b) PCA(m =10) – K-fold

(E) PCA (m=8) – Single Fold (F) PCA(m =8) – K-fold

Figure4: min DCF with different values of λ

From the previous figure, we can't see a big difference in performance between the single fold and the k-fold analysis. However, we could notice a best value for λ that provides a min DCF.

Also, a point to be noticed is the performance differences between different applications such as in a no PCA application the performance is better either in single fold or in k-fold and by further reducing the dimensionality of the dataset it is found that the performance becomes poorer.

Based on the previous observations we could select the best value for the hyper-parameter is $\lambda = 10^{-4}$. Now let's compute again the min DCF:

	Single Fold			5-fold		
	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$
Z-Normalized Features – no PCA						
Log Reg ($\lambda = 10^{-4}, \pi_T = 0.5$)	0.093	0.233	0.233	0.112	0.285	0.348
Log Reg ($\lambda = 10^{-4}, \pi_T = 0.1$)	0.105	0.214	0.283	0.121	0.296	0.368
Log Reg ($\lambda = 10^{-4}, \pi_T = 0.9$)	0.099	0.256	0.231	0.111	0.317	0.343
Z- Normalized Features – PCA (m = 8)						
Log Reg ($\lambda = 10^{-4}, \pi_T = 0.5$)	0.244	0.461	0.666	0.259	0.511	0.660
Log Reg ($\lambda = 10^{-4}, \pi_T = 0.1$)	0.246	0.447	0.680	0.263	0.493	0.679
Log Reg ($\lambda = 10^{-4}, \pi_T = 0.9$)	0.243	0.453	0.660	0.256	0.551	0.648
Z-Normalized Features – PCA (m = 10)						
Log Reg ($\lambda = 10^{-4}, \pi_T = 0.5$)	0.150	0.312	0.498	0.183	0.419	0.535
Log Reg ($\lambda = 10^{-4}, \pi_T = 0.1$)	0.169	0.291	0.533	0.181	0.398	0.555
Log Reg ($\lambda = 10^{-4}, \pi_T = 0.9$)	0.166	0.321	0.475	0.185	0.470	0.514

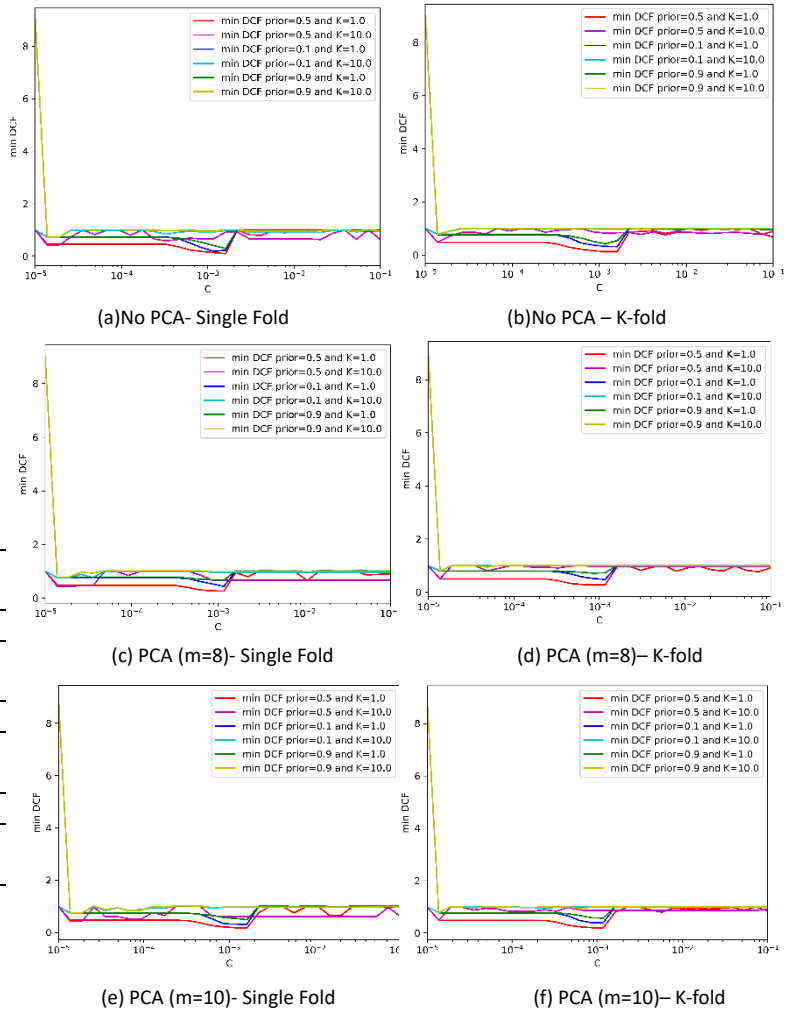
Figure 5: min DCF computed using the estimated hyper-parameter.

From the previous results we could see a superiority in performance using application with a prior of 0.5 over the other applications. Also, It is noted that a no PCA version of the application gives better min-DCF value over all as we use a dimensionality reduction of m =10 the performance decreases slightly and when using further dimensionality reduction of m = 8 the performance is worse and that makes sense as we predicted. Also, rebalancing the cost of different classes(π_T) has no significant impact on the min-DCF.

3.3 Support Vector Machines

We now turn our attention to SVMs. We will see linear svm (with and without class balancing) and quadratic svm (with polynomial and rbf kernel), even if from the previous results we expect the linear SVM to perform better. The main difference between support vector machines and logistic regression is that they try to find the hyperplane that separates the classes with the largest margin.

For linear SVM we need to tune the hyper-parameters C and K. I tried K=1.0 and K=10.0 with both single-fold and k-fold cross-validation.



(a)No PCA- Single Fold (b)No PCA – K-fold

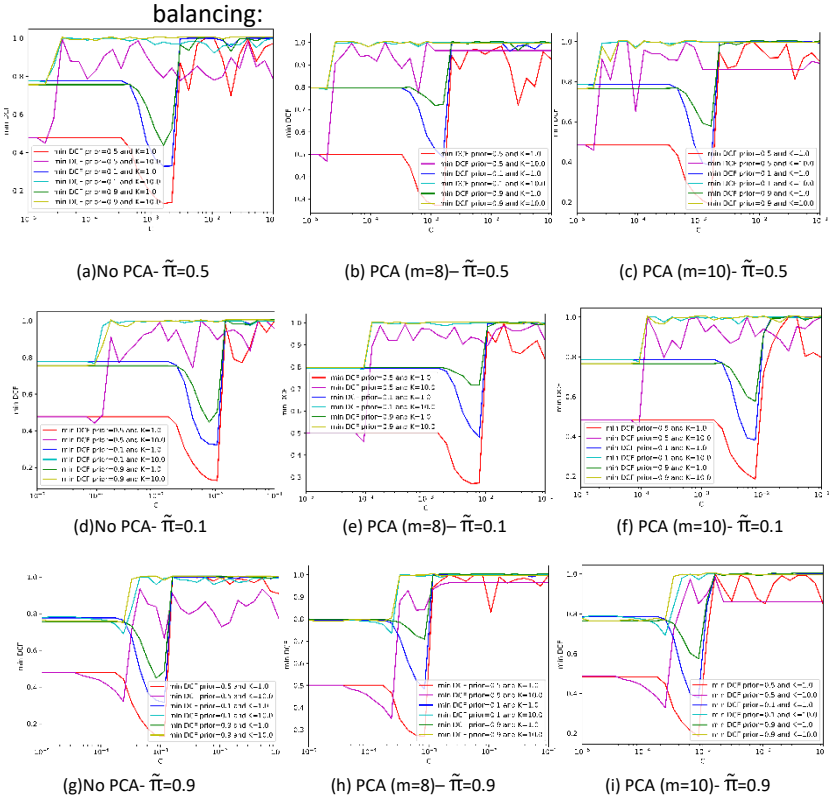
(c) PCA (m=8)- Single Fold (d) PCA (m=8)– K-fold

(e) PCA (m=10)- Single Fold (f) PCA (m=10)– K-fold

Figure 6: min DCF for linear svm without class balancing.

From figure (6) we select as best values K = 1.0 and C = 2*10-3. We notice that k-fold results are consistent with single-fold results, meaning that our model behaves correctly.

Now let's look at the min DCF for linear svm with class balancing:



(a)No PCA- $\tilde{\pi}$=0.5 (b) PCA (m=8)– $\tilde{\pi}$=0.5 (c) PCA (m=10)- $\tilde{\pi}$=0.5

(d)No PCA- $\tilde{\pi}$=0.1 (e) PCA (m=8)– $\tilde{\pi}$=0.1 (f) PCA (m=10)- $\tilde{\pi}$=0.1

(g)No PCA- $\tilde{\pi}$=0.9 (h) PCA (m=8)– $\tilde{\pi}$=0.9 (i) PCA (m=10)- $\tilde{\pi}$=0.9

Figure 7: min DCF for linear svm with class balancing.

Figure (7) shows that K = 1 provides a better and more stable min DCF. We decided to choose a different value of the hyper-parameter C for each prior. The table below summarizes the computed min DCF:

	Single Fold			5-fold		
	$\tilde{\pi}$ = 0.5	$\tilde{\pi}$ = 0.1	$\tilde{\pi}$ = 0.9	$\tilde{\pi}$ = 0.5	$\tilde{\pi}$ = 0.1	$\tilde{\pi}$ = 0.9
Z-Normalized Features – no PCA						
Linear SVM (C=2*10^{-3}, unbalanced)	0.093	0.236	0.236	0.990	0.998	1.005
Linear SVM (C=2*10^{-3}, π_T = 0.5)	0.112	0.216	0.350	0.126	0.315	0.455
Linear SVM (C=10^{-2}, π_T = 0.1)	0.112	0.216	0.350	0.126	0.314	0.457
Linear SVM (C=10^{-3}, π_T = 0.9)	0.127	0.207	0.395	0.127	0.310	0.419
Z- Normalized Features – PCA (m = 10)						
Linear SVM (C=2*10^{-3}, unbalanced)	1.000	1.000	1.015	0.873	0.999	0.990
Linear SVM (C=2*10^{-3}, π_T = 0.5)	0.164	0.307	0.499	0.245	0.411	0.750
Linear SVM (C=10^{-2}, π_T = 0.1)	0.164	0.307	0.499	0.245	0.407	0.752
Linear SVM (C=10^{-3}, π_T = 0.9)	0.170	0.307	0.522	0.194	0.395	0.578
Z-Normalized Features – PCA (m = 8)						
Linear SVM (C=2*10^{-3}, unbalanced)	0.997	1.000	1.011	0.969	0.998	1.005
Linear SVM (C=2*10^{-3}, π_T = 0.5)	0.252	0.440	0.651	0.943	1.000	1.005
Linear SVM (C=10^{-2}, π_T = 0.1)	0.252	0.440	0.651	0.998	0.998	1.005
Linear SVM (C=10^{-3}, π_T = 0.9)	0.252	0.427	0.658	0.703	0.854	0.876

Figure 8: min DCF on the validation set for linear svm with/without balancing.

The class re-balancing has improved the model and the best results are given by the linear model with πT = 0.9 and C = 10-3. Intuitively, this configuration had the best performance due to the fact that the training datatset is imbalanced.

The first thing that stands out is that the unbalanced SVM performs really well in the noPCA and single-fold case, however its results get worse in all other configurations. There could be many reasons and we assumed it's just randomness: its performance happened to align favorably with the particular data in that fold.

In addition to this, we can notice that PCA degraded again the performance especially when focusing on the m = 8 configuration meaning that, even if it retains 95% of the variance, we lost discriminant information.

Regarding the non-linear formulation in SVMs, the non- linearity is obtained through an implicit expansion of the features in a higher dimensional space. The dual svm formulation depends on the training samples only through dot-products, and we can compute a classification score through scalar products between training and evaluation samples. For this reason, it's not required to explicitly compute the feature expansion, it's enough to be able to compute the scalar product between the expanded features, the so-called kernel function k:

$$k(x_1, x_2) = \phi(x_1)^T \phi(x_2)$$

We will test two different kernels:

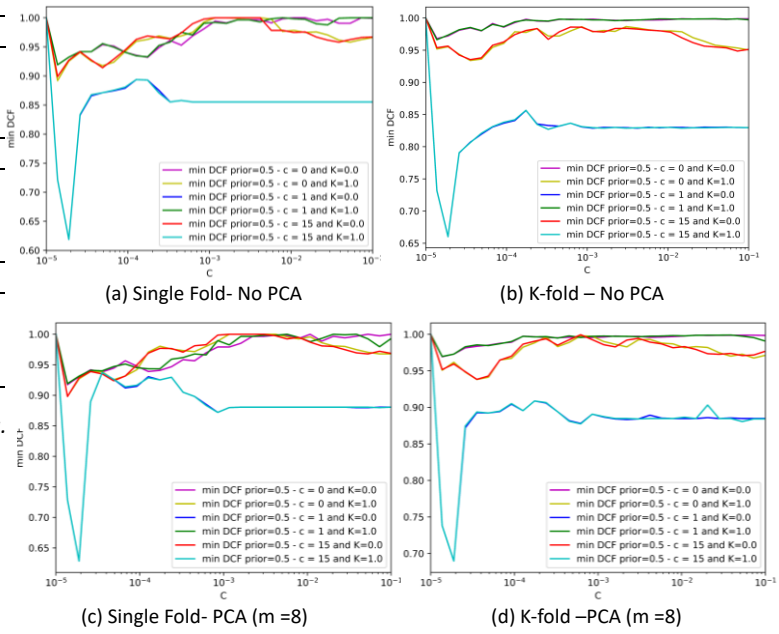
1. Polynomial kernel of d = 2;

$$k(x_i, x_j) = (x^T_i \, x_j + c)^d$$

2. Radial Basis Function (RBF) kernel:

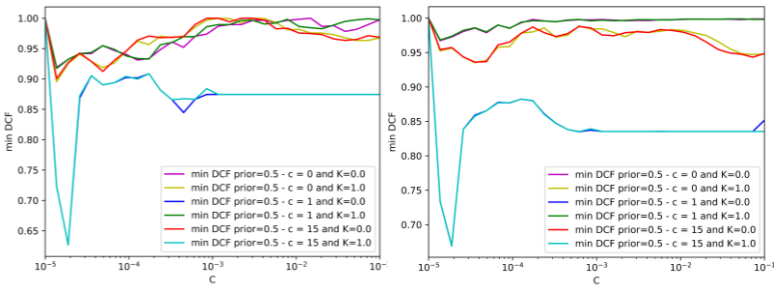
$$k(x_i, x_j) = e^{-\gamma \|xi - xj\|^2}$$

For the quadratic polynomial svm we need to estimate two hyper-parameters c and C through a cross-validation.



(a) Single Fold- No PCA (b) K-fold – No PCA

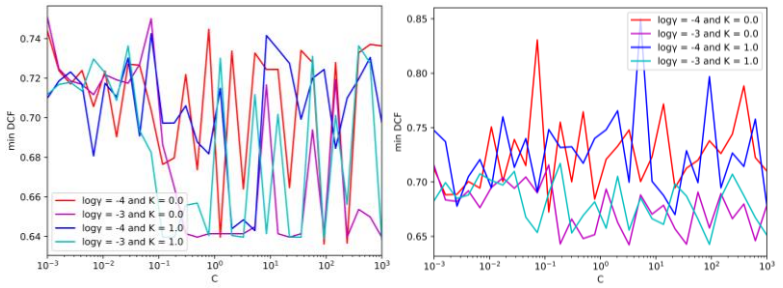
(c) Single Fold- PCA (m =8) (d) K-fold –PCA (m =8)



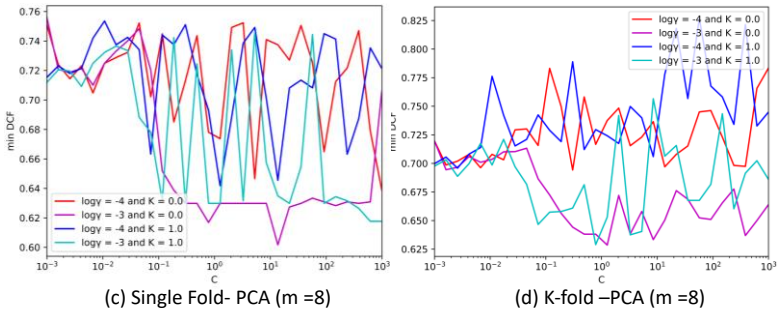
(e) Single Fold- PCA (m =10) (f) K-fold –PCA (m =10)

Figure 9: min DCF for different values of C (polynomial svm)

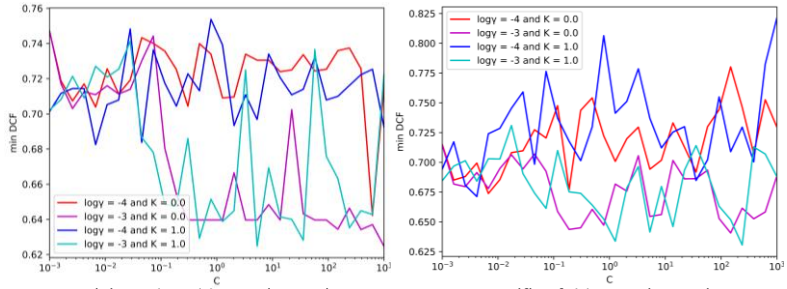
From the above figure we see that the minimum min DCF is found with c=15 and C=3∗10-5. Now let's estimate the two hyper-parameters γ and C for the RBF SVM:



(a) Single Fold- No PCA (b) K-fold – No PCA



(c) Single Fold- PCA (m =8) (d) K-fold –PCA (m =8)



(e) Single Fold- PCA (m =10) (f) K-fold –PCA (m =10)

Figure 10: min DCF for different values of C (RBF svm)

Looking at figure (ADD NUMBER) we selected γ = 10-3 and C = 1. Now it's time to look at the performances:

	Single Fold			5-fold		
	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$
Z-Normalized Features – no PCA						
Linear SVM ($C = 10^{-3}$, $\pi_T = 0.9$)	0.127	0.207	0.395	0.127	0.310	0.419
Polynomial SVM ($C = 3*10^{-5}$, c = 15, d = 2)	0.852	1.000	0.976	0.794	0.999	0.994
RBF SVM ($C = 1$, $\gamma = 10^{-3}$)	0.640	0.885	0.961	0.677	0.957	0.985
Z- Normalized Features – PCA (m = 10)						
Linear SVM ($C=10^{-3}$, $\pi_T = 0.9$)	0.170	0.307	0.522	0.194	0.395	0.578
Polynomial SVM ($C = 3*10^{-5}$, c = 15, d = 2)	0.891	1.000	0.989	0.851	0.999	0.997
RBF SVM ($C = 1$, $\gamma = 10^{-3}$)	0.679	0.894	0.978	0.674	0.944	0.981
Z-Normalized Features – PCA (m = 8)						
Linear SVM ($C=10^{-3}$, $\pi_T = 0.9$)	0.252	0.427	0.658	0.703	0.854	0.876
Polynomial SVM ($C = 3*10^{-5}$, c = 15, d = 2)	0.922	1.000	1.011	0.884	0.999	0.998
RBF SVM ($C = 1$, $\gamma = 10^{-3}$)	0.624	0.926	0.965	0.656	0.980	0.977

Figure 11: min DCF on the validation set for the three types of SVMs

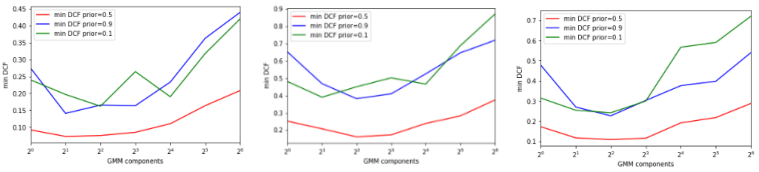
The RBF SVM performed slightly better that the polynomial sums but anyway we can conclude that the linear approaches fit better the task. This time PCA didn't degrade the results but it didn't improve them either, so we cans say that it hasn't been effective in improving the results quality. However, it helped reducing the complexity and speeding computations. Anyway, it is important to notice that SVMs performed worse than both gaussian classifiers and linear logistic regression.

3.4 Gaussian Mixture Models

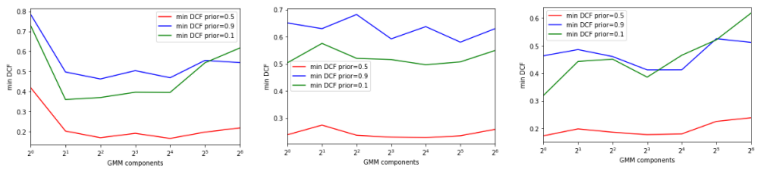
In the GMM and as we know from the dataset analysis that the data consists of 3-different age groups and therefore we expect a better training phase with different components as this model will further analyze the training data and could perform slightly better. We will use GMM with full covariance, full diagonal, and tied covariance. In the tied covariance model, tying takes place at class level, so different classes have distinct covariance matrices.

We'll use the single fold and cross-validation to evaluate how good the model is on the validation set and based on that we'll select the number of components.

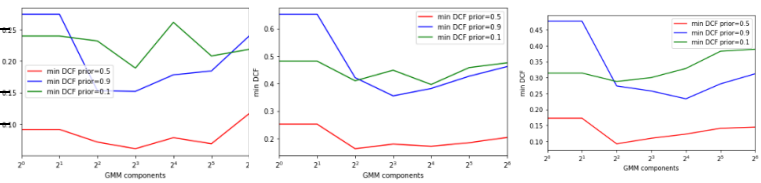
Results for Single-fold validation: -



(a) No PCA- Full Covariance (b) PCA (m=8)- Full Covariance (c) PCA (m =10)- Full Covariance



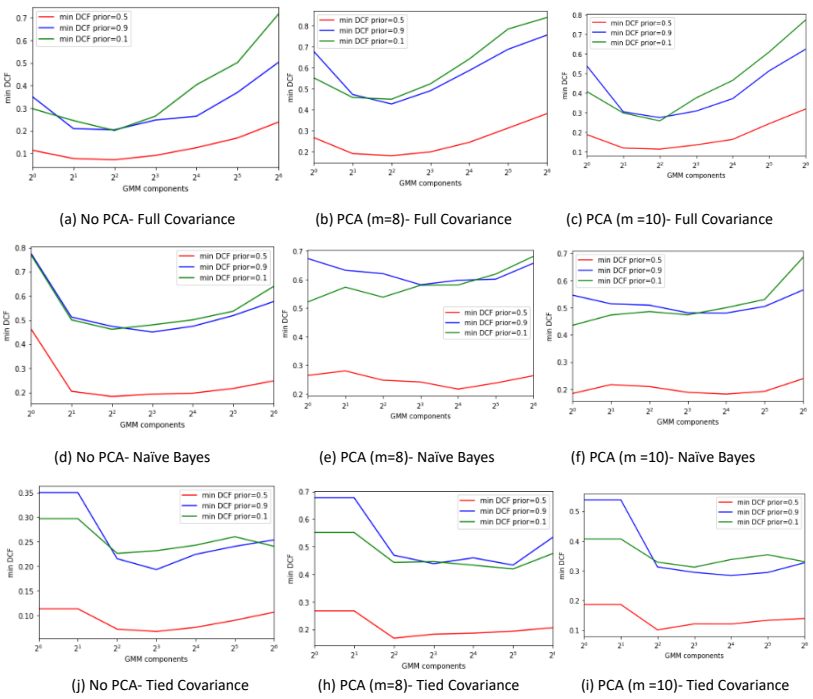
(d) No PCA- Naïve Bayes (e) PCA (m=8)- Naïve Bayes (f) PCA (m =10)- Naïve Bayes



(j) No PCA- Tied Covariance (h) PCA (m=8)- Tied Covariance (i) PCA (m =10)- Tied Covariance

Figure 12: min DCF on different priors, different models and numbers of GMM components (Single Fold)

Applying K-fold validation and compare the results:-



(a) No PCA- Full Covariance (b) PCA (m=8)- Full Covariance (c) PCA (m =10)- Full Covariance

(d) No PCA- Naïve Bayes (e) PCA (m=8)- Naïve Bayes (f) PCA (m =10)- Naïve Bayes

(j) No PCA- Tied Covariance (h) PCA (m=8)- Tied Covariance (i) PCA (m =10)- Tied Covariance

Figure 13: min DCF on different priors, different models and numbers of GMM components (K- Fold)

The K-fold results give us a better and more accurate perspective on the problem so we will focus on analyzing its graphs.
From the graphs we could see a future prediction on the number of components in general we could see that the best number of components is balancing between [2,16] and that is consistent with our previous prediction such that starting from nearly 3 components the performance is getting better.
And based on the previous results we selected the best number of components for the "Full-Covariance" type to be at (#Components =4)
And for the "Naïve Bayes" type to be (#Components =16) and for the "Tied Covariance" (#Components =8).

The following Table describes the min DCF with both single-fold and k-fold:

	Single Fold			5-fold		
	$\tilde{\pi}$ = 0.5	$\tilde{\pi}$ = 0.1	$\tilde{\pi}$ = 0.9	$\tilde{\pi}$ = 0.5	$\tilde{\pi}$ = 0.1	$\tilde{\pi}$ = 0.9
Z-Normalized Features – no PCA						
Full-Covariance (FC), 4-G	0.075	0.162	0.165	0.071	0.201	0.204
Diagonal-Covariance (NB), 16-G	0.166	0.396	0.469	0.197	0.501	0.474
Tied Full-Covariance (TC), 8-G	0.061	0.189	0.152	0.067	0.232	0.193
Z- Normalized Features – PCA (m = 8)						
Full-Covariance (FC), 4-G	0.162	0.451	0.382	0.181	0.449	0.427
Diagonal-Covariance (NB), 16-G	0.228	0.497	0.638	0.216	0.581	0.597
Tied Full-Covariance (TC), 8-G	0.179	0.448	0.355	0.183	0.446	0.438
Z-Normalized Features – PCA (m = 10)						
Full-Covariance (FC), 4-G	0.108	0.241	0.226	0.112	0.257	0.273
Diagonal-Covariance (NB), 16-G	0.180	0.465	0.413	0.182	0.499	0.480
Tied Full-Covariance (TC), 8-G	0.109	0.300	0.258	0.121	0.312	0.294

Figure 14: min DCF on validation set for GMM

From the previous table we could deduce some important information, the performance of the Naïve Bayes Type performed the worse in all the applications in every case or dimensionality reduction and that is expected as the Naïve Bayes depends on a diagonal Matrix and neglecting the correlation between the features which we know from feature analysis is highly correlated.
The Full-Covariance performed significantly better in most of the applications, and it came close in performance with the Tied-Covariance with a slight improvement for the Tied Covariance when not using Dimensionality reduction methods.
By analyzing different applications, it is clear that using an application with prior = 0.5 gives the best results at all.

4. Score Recalibration

Until now we have only looked at the min DCF metric. As already discussed, it measures the cost we would pay if we made optimal decisions for the validation set using the scores of the recognizer.
However, the cost that we actually pay depends on the goodness of the threshold we use to perform class assignment. We therefore start considering the actual DCFs.
If scores are well calibrated, the optimal threshold that optimizes the Bayes risk is the theoretical threshold:

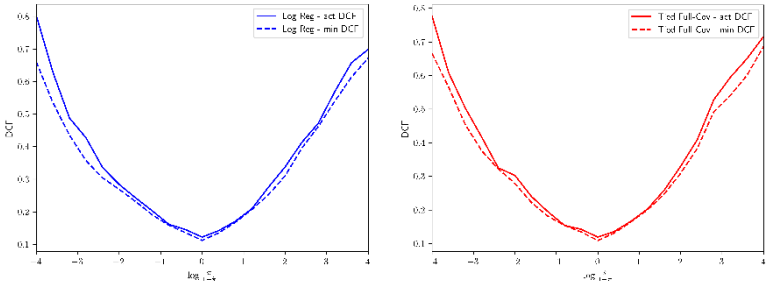
$$t = -\log \frac{\tilde{\pi}}{1-\tilde{\pi}}$$

We now evaluate the actual dcf to measure how good the models would behave if we were using the theoretical threshold for each application instead of the optimal one:

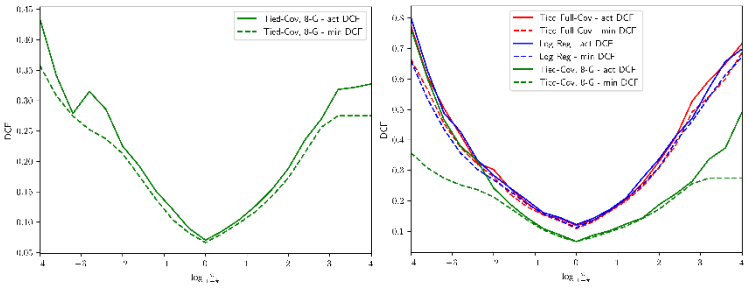
	$\tilde{\pi}$ = 0.5		$\tilde{\pi}$= 0.1		$\tilde{\pi}$= 0.9	
	min DCF	act DCF	min DCF	act DCF	min DCF	act DCF
Z-Normalized Features – no PCA						
MVG Tied Full-Cov	0.109	0.119	0.299	0.310	0.342	0.371
Log Reg ($\lambda = 10^{-4}$, $\pi_T = 0.5$)	0.112	0.122	0.285	0.299	0.348	0.365
Tied-Cov GMM, 8-G	0.067	0.066	0.232	0.281	0.193	0.210

Figure 15: comparison between min DCF and actual DCF

All three models provide scores already well-calibrated. This analysis can also be verified through Bayes error plots, which show the DCFs for different applications:



(a) Logistic Regression (b) MVG-Tied-Cov



(c) tied-Cov-GMM (d) Total

Figure 16: Bayes error plots pre-calibration

Observing the plots we can see that logistic regression and multivariate gaussian classfier are already well-calibrated, especially for prior log-odds values around 0. This confirms what the results of table number ADD NUMBER. However, the gmm tied-cov with 8 components doesn't perform well when the absolute value of the prior log-odds is higher than 2 (the minDCF and actDCF plots diverge).

We will then consider re-calibrating the scores. It consists of transforming the scores so that the theoretical threshold provides values closer to optimal over a wide range of effective prior π~.
More in details, we have to compute a transformation function f that maps the classifiers scores s to well-calibrated scores s_{cal} = f(s). We assume that function f is linear in s:

$$f(s) = \alpha s + \beta$$

Since f (s) should produce well-calibrated scores, it can be interpreted as the log-likelihood ratio for the two class hypotheses:

$$f(s) = \log \frac{f_{S|C}(s|\mathcal{H}_T)}{f_{S|C}(s|\mathcal{H}_F)} = \alpha s + \beta$$

The class posterior probability for prior π~ corresponds to:

$$\log \frac{P(C = \mathcal{H}_T|s)}{P(C = \mathcal{H}_F|s)} = \alpha s + \beta + \log \frac{\tilde{\pi}}{1 - \tilde{\pi}}$$

We can interpret the scores s as features so that this expression will be similar to the log posterior ratio of the logistic regression model. If we let:

$$\beta' = \beta + log \frac{\tilde{\pi}}{1-\tilde{\pi}}$$

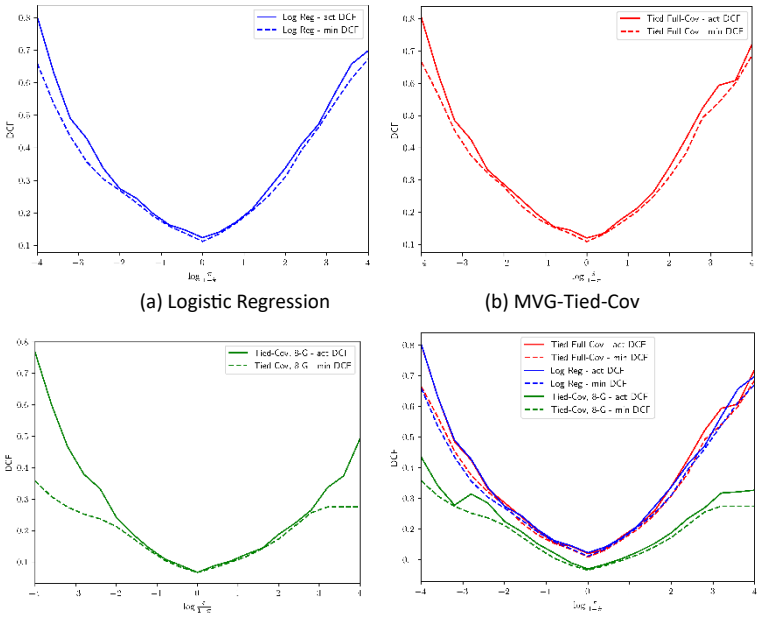
we end up having the same model. We can use the prior-weighted logistic regression model to learn the model parameters α, β' over our training scores. To get the calibrated scores f(s) we need to compute:

$$f(s) = \alpha s + \beta' - log \frac{\tilde{\pi}}{1-\tilde{\pi}}$$

Looking at the formula we see that we have to specify a prior π~ so we are effectively optimizing the calibration for a specific application π~. Anyway, the model will still provide good calibration for different applications. Let's compute DCFs and Bayes error plots

	$\tilde{\pi} = 0.5$		$\tilde{\pi} = 0.1$		$\tilde{\pi} = 0.9$	
	min DCF	act DCF	min DCF	act DCF	min DCF	act DCF
Z-Normalized Features – no PCA						
MVG Tied Full-Cov	0.109	0.120	0.299	0.306	0.342	0.377
Log Reg ($\lambda = 10^{-4}, \pi_T = 0.5$)	0.112	0.123	0.285	0.301	0.348	0.363
Tied-Cov GMM, 8-G	0.067	0.070	0.232	0.259	0.193	0.204

Figure 17: min DCF and actual DCF post-calibration



(a) Logistic Regression (b) MVG-Tied-Cov

(c) tied-Cov-GMM (d) Total

Figure 18: Bayes error plots post-calibration

From figures above we can see that the re-calibration works very well on all models and the values of the actual DCF becomes very similar to the min DCF ones. We notice it especially in the gmm tied-cov where the divergence has been reduced. From the total comparison (subplot d of figure 18), we can see that logistic regression and mvg tied full-cov provide pretty the same value as min DCF, while gmm tied-cov performs better.

5. Experimental Results

In this section we'll observe the behavior of the models on the test set in terms of min DCF. We train the model with the training set and use the test set as evaluation data. Here are the summarized results:

	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$
Z-Normalized Features – no PCA			
MVG Full-Cov	0.121	0.323	0.313
MVG Diag-Cov	0.433	0.817	0.702
MVG Tied Full-Cov	0.116	0.303	0.312
Log Reg ($\lambda = 10^{-4}, \pi_T = 0.5$)	0.120	0.304	0.309
Log Reg ($\lambda = 10^{-4}, \pi_T = 0.1$)	0.121	0.298	0.340
Log Reg ($\lambda = 10^{-4}, \pi_T = 0.9$)	0.121	0.332	0.280
Linear SVM (C = $2*10^{-3}$, *unbalanced*)	0.586	0.992	0.951
Linear SVM (C = $2*10^{-3}$, $\pi_T = 0.5$)	0.530	0.978	0.939
Linear SVM (C = 10^{-2}, $\pi_T = 0.1$)	0.286	0.683	0.594
Linear SVM (C = 10^{-3}, $\pi_T = 0.9$)	0.116	0.301	0.310
Polynomial SVM (C = $3*10^{-5}$, c = 15, d = 2)	0.796	0.999	0.999
RBF SVM (C = 1, $\gamma = 10^{-3}$)	0.881	0.995	1.000
GMM Full-Cov, 4-G	0.122	0.424	0.248
GMM Diag-Cov, 16-G	0.191	0.477	0.446
GMM Tied Full-Cov, 8-G	0.065	0.187	0.211
Z- Normalized Features – PCA (m = 10)			
MVG Full-Cov	0.687	0.983	0.988
MVG Diag-Cov	0.681	0.984	0.985
MVG Tied Full-Cov	0.699	0.991	0.985
Log Reg ($\lambda = 10^{-4}, \pi_T = 0.5$)	0.681	0.988	0.983
Log Reg ($\lambda = 10^{-4}, \pi_T = 0.1$)	0.645	0.976	0.981
Log Reg ($\lambda = 10^{-4}, \pi_T = 0.9$)	0.722	0.993	0.990
Linear SVM (C = $2*10^{-3}$, *unbalanced*)	0.725	0.985	0.994
Linear SVM (C = $2*10^{-3}$, $\pi_T = 0.5$)	0.527	0.960	0.934
Linear SVM (C = 10^{-2}, $\pi_T = 0.1$)	0.890	1.000	0.996
Linear SVM (C = 10^{-3}, $\pi_T = 0.9$)	0.977	0.999	0.999
Polynomial SVM (C = $3*10^{-5}$, c = 15, d = 2)	0.999	1.000	1.005
RBF SVM (C = 1, $\gamma = 10^{-3}$)	0.929	1.000	0.999
GMM Full-Cov, 4-G	0.519	1.000	0.890
GMM Diag-Cov, 16-G	0.432	0.820	0.878
GMM Tied Full-Cov, 8-G	0.494	0.983	0.952

Figure 19: min DCF over test set

The first observation is that the results aren't that different from the ones that we got over the validation set, meaning that the models are consistent. However, there are some differences and they are probably due to the fact that the evaluation population and the training population are unbalanced in the opposite sense (we know the training set has significantly more female samples, while the test set has significantly more male samples).

We can also compare our best classifiers through a ROC plot considering the models without PCA.

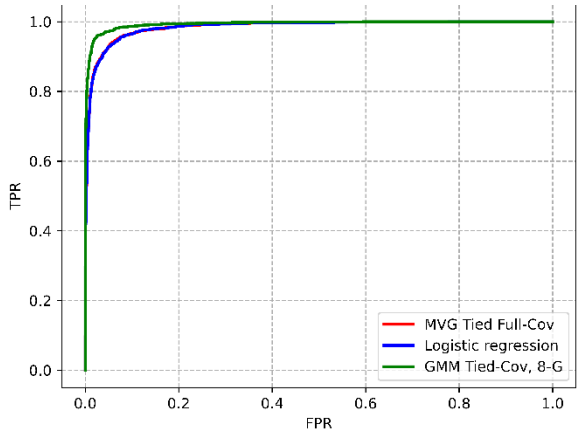
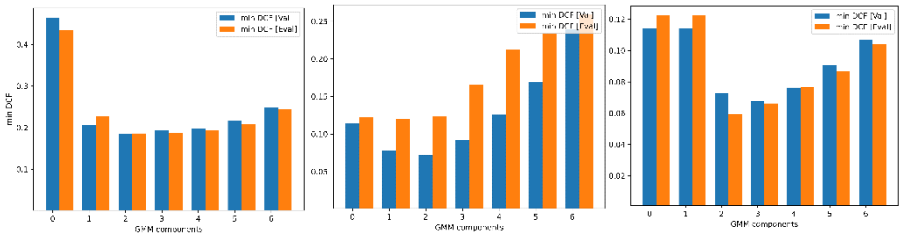


Figure 20: Roc between the three best models

The best classifiers are the ones with the highest AUC (Area Under Curve), in our case the GMM tied-cov with 8 components performs better than the others. Lastly, we can also verify if the chosen hyper-parameters provided close to optimal results on the test set, understanding if we took the correct values. In the next figures, we have compared the min DCF for different values of the hyper-parameters on the test/evaluation set and validation set.



(a) GMM-Diag Cov (b) GMM-Full Cov (c) GMM-tied Cov

Figure 21: min DCF for different components for different modes of GMM

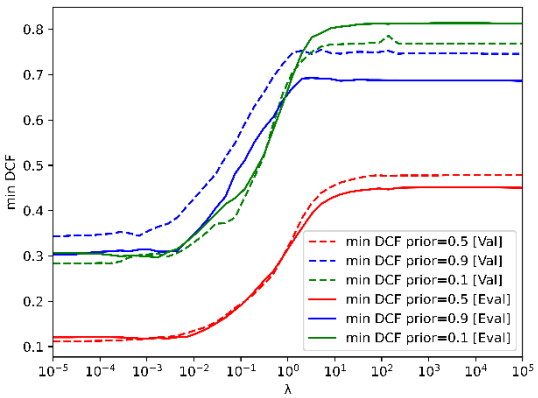
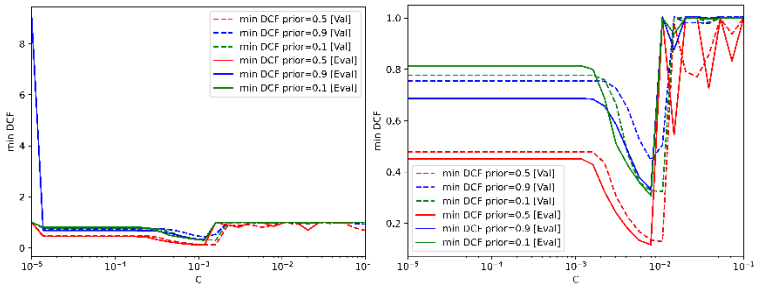
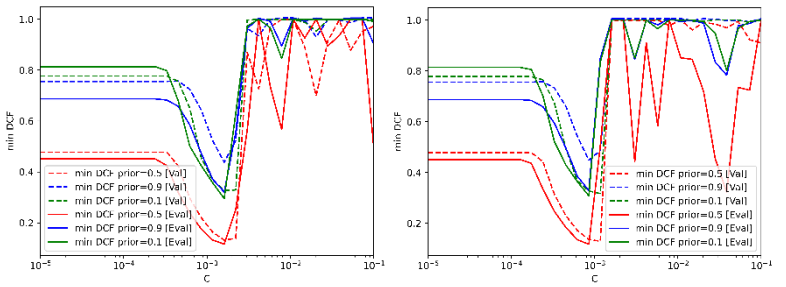


Figure 21: comparison of min DCF for different values of λ



(a) Linear SVM without class balancing (b) Linear SVM with class balancing $\pi_T = 0.5$



(c) Linear SVM with class balancing $\pi_T = 0.1$ (d) Linear SVM with class balancing $\pi_T = 0.9$

Figure 22: Comparison of min DCF for different values of C

We can conclude that the values we chose for the hyper-parameters λ and C were quite accurate for most of the models.

6. Conclusions

To summarize we have understood that linear models performed better than quadratic ones on this dataset.
The best models are the gmm tied full-cov 8G (noPCA) and the mvg tied full-cov (noPCA).
The correct choices for the hyper-parameters led us to a min DCF ≈ 0.1 for the target application ($\tilde{\pi} = 0.5$). We must also notice that the min DCF of the other applications is not that higher (respectively ≈ 0.2 for both $\tilde{\pi} = 0.1$ and $\tilde{\pi} = 0.9$).
In addition to this, we can definitely state that PCA didn't improve our results; it was certainly useful to speed up computations but the increase of min DCF is consistent.
To end up, the choices made on the training/validation sets proved to be effective also for the evaluation set.