Gender Identification Using Machine Learning And Pattern Recognition

Ahmadreza Farmahini Farahani
dept. Control and Computer Engineering
Politecnico di Torino
Torino, Italy
s300909@studenti.polito.it

***Abstract*—In this project, we try to use machine learning algorithms and the Gender Identification dataset to classify our binary features. We first start to visualize the distribution of each feature and we calculate correlation matrix to see how much our data is correlated. Moreover, we start classification with simple Gaussian classifiers and we analyze discriminate classifiers to understand the pattern behind our dataset. At the end, we start to evaluate the given test data to see how much our models have been accurate and reliable.**

I. Introduction

Pattern recognition is the use of machine learning algorithms to identify patterns. It classifies data based on statistical information or knowledge gained from patterns and their representation.

In this technique, labeled training data is used to train pattern recognition systems. A label is attached to a specific input value that is used to produce a pattern-based output. In this project we try to analyze a specific dataset regarding different traditional machine learning classifiers. Before we start analyzing the classifiers we need to consider below remarks.

A. Dataset

The goal of the project is the development of a classifier to identify the gender from high-level features representing face images. These kind of tools find application, for example, as pre-processing for gender dependent face recognition models. The dataset consists of image embeddings, i.e. low-dimensional representations of images obtained by mapping face images to a common, low-dimensional manifold (typically few hundred dimensions), for example by means of suitable neural networks. To keep the model tractable, the dataset consists of synthetic data, and embeddings have significantly lower dimension than in real use-cases.

B. Classifiers

For analyzing the patterns available in the dataset we use different classifiers to see how each classifiers can fit our dataset and can be used for the final application. Each of the following classifiers can be used in different manners. We analyze vest possible approaches within each classifier.

- Generative models
- Logistic Regressions
- Support Vector Machines
- Gaussian Mixture Models

At the end, we try to select best applications from our analysis. In the post-process section some other usefull approaches like fusion will be tested.

C. Training

The models must be trained over the training set only. When needed, validation data can be extracted from the training set (for example, to compare competing models, to select suitable values for the hyper-parameters of each model, or to train score calibration models). Models should be trained to optimize the target application, but performance of the models for alternative applications should also be analyzed and discussed. At the end of this stage, a candidate solution should be provided for the classification task. To understand which model is most promising, and to assess the effects of using PCA, we have employed K-Fold cross validation. In fact, all of the following results have been obtained with K-Fold Validation with $K = 6$.

- Inside each cell of the following tables, we have reported the $minDCF$. We do not care about $actDCF$ in this initial phase.
- $minDCF$ has been computed with C_fn and C_fp both equal to one, as we do not have any specific requirements regarding the miss-classification costs. In particular, we will consider C:
  1) one balanced application:
  $$(\pi, C_fn, C_fp) = (0.5, 1, 1) \tag{1}$$
  2) two unbalanced cases:
  $$(\pi, C_fn, C_fp) = (0.1, 1, 1) \tag{2}$$
  $$(\pi, C_fn, C_fp) = (0.9, 1, 1) \tag{3}$$

D. Experimental Analysis

The proposed solution must be evaluated on the evaluation set. The evaluation samples should be treated as independent, i.e. the value or the score of an evaluation sample should have no influence on the score of a different sample. The evaluation should report the performance for the target application, but also analyze how the model would perform for alternative use-cases.

II. Pre-Processing

Data pre-processing, a component of data preparation, describes any type of processing performed on raw data to prepare it for another data processing procedure. It has traditionally been an important preliminary step for the training process.

Data pre-processing transforms the data into a format that is more easily and effectively processed in machine learning and other data science tasks.

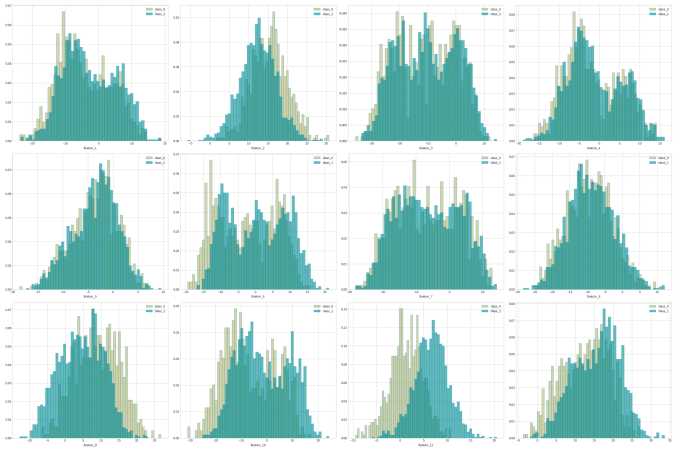

Fig. 1. Distribution of raw features

In the Fig. 1 we can see that the features mainly have Gaussian distribution. However, in almost half of the cases, we have multi-nomial distributions. The multi-nomial distribution is consist of combination of multiple Gaussian distributions. The reason for this phenomenon lies behind the characteristics of the Gender Identification dataset. Maybe, the speakers have different age and since human voice have different pitches in different ages, so the embedding with each age range could be different but it can have gaussian distribution.

Furthermore, we try to visualize the correlation matrix with heat map, to see how much our features are correlated with each other. We try two main methods for pre-processing our dataset. Principal Component Analysis and two data normalization.

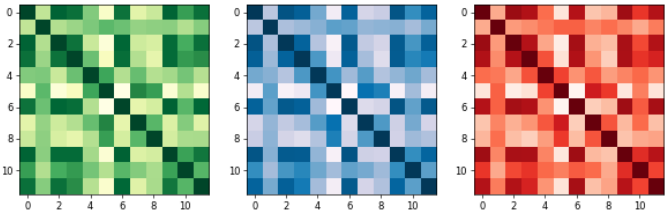

Fig. 2. Features correlation heatmap

We also try to see how correlated are features within each other, In Fig. 2 we can see the correlation heatmap of the features.

A. Normalization

Normalization plays a crucial role in data pre-processing and analysis in various fields, including machine learning and statistics. It involves transforming data to a standardized scale, ensuring fair comparisons and removing biases. We use two kind of normalization, Z-scored and Gaussianization.

Z-scored value is to understand how far the data point is from the mean. Technically, it measures the standard deviations below or above the mean. Z-scored normalization is useful for those kinds of data analysis wherein there is a need to compare a value with respect to a mean(average) value, such as results from tests or surveys. Thus, Z-scored normalization is also popularly known as Standardization. The following formula is used in the case of Z-scored normalization on every single value of the dataset.

$$Z = \frac{(x - \mu)}{\sigma} \tag{4}$$

In the Fig. 3 we can visualize the Z-scored normalized

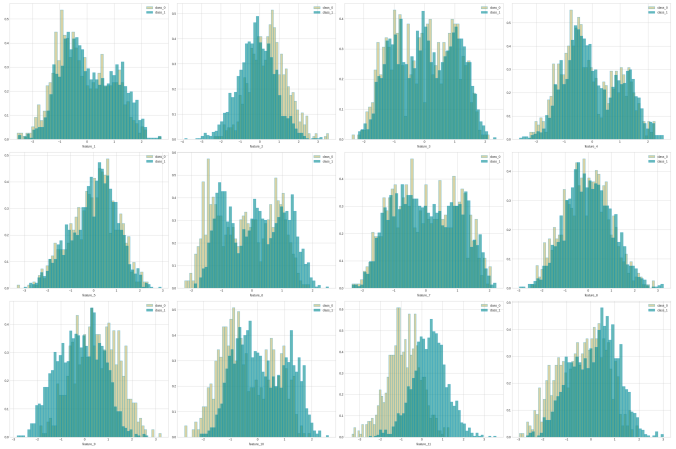

Fig. 3. The distribution of Z-scored features

features distributions. We can see that the shape of distribution is not changed, and the scale of each feature mapped based on the covariance and mean of each features. We can see no changes on the shape of the distributions. But are the scales important for our analysis? We answer this question in the future parts that whether the scale of the dataset is important or not. We have to mention that the original distribution of features that we visualized in Fig. 1 are well centered.

Gaussianization is the process of transforming data or distributions to follow a Gaussian (normal) distribution. It aims to make the data more amenable to analysis and modeling techniques that assume Gaussianity. Techniques such as power transformations, quantile mapping, and standardization are commonly used to achieve Gaussianization. By approximating a Gaussian distribution, the data becomes more suitable for statistical analysis and enables the use of parametric models.

In Fig. 4 the distribution of data has been Gaussianized.

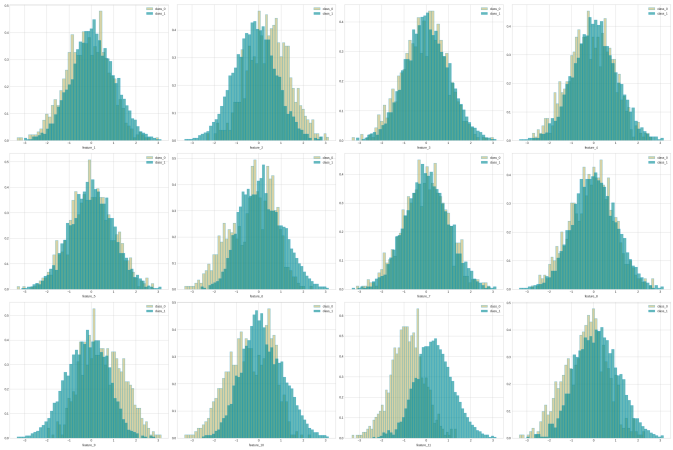

Fig. 4. The distribution of Gaussianized features

B. Principal Component Analysis (PCA)

Principal component analysis (PCA) has been called one of the most valuable results from applied linear algebra. PCA is used abundantly in all forms extracting relevant information from datasets. With minimal additional effort PCA provides a roadmap for how to reduce a complex data set to a lower dimension to reveal the sometimes hidden, simplified dynamics that often underlie it.

With PCA we look for the directions that have the most valuable information about the data. In other words we are looking for directions that have the most variance. Now We will see how and why PCA is intimately related to the mathematical technique of singular value decomposition (SVD). Let x be a zero-mean random variable. Suppose we want the direction w such that the projection of x along this direction has maximum variance:

$$\max_{w} Var(w'x) \quad \text{st.} \quad ||w|| = 1. \tag{5}$$

We have:

$$Var(w'x) = w'xx'w = w'\Sigma w. \tag{6}$$

The Lagrangian is:

$$L = w'\Sigma w + \lambda(w'w - 1) \tag{7}$$

The stationary condition is:

$$\frac{\partial L}{\partial w} = 2\Sigma w - 2\lambda w = 0 \tag{8}$$

$$\Sigma w = \lambda w \tag{9}$$

Thus w is an eigenvector of Σ. Since:

$$w'\Sigma w = w'(\lambda w) = \lambda \tag{10}$$

The direction with maximum variance is the largest eigenvector. This procedure can be iterated to get the second largest variance projection (orthogonal to the first one), and so on. For a set of data points, we use the ML estimate of the covariance matrix.

In the Fig. 2 we can see that the data has correlation between some features, there is a possibility that PCA dimensionality reduction works well on the dataset, in the next steps we try to use PCA to see how well it works on validation. Based on this figure we will try to use PCA with 6, 8 and 10 dimensions to find the most promising dimension. We also try to use Z-scored and Gaussianization on the dataset.

The next approach we consider in training phase is using K-Fold cross validation. We use these two methods to calculate minimum K-Fold ($minDCF$) for each model and then we decide which approach is more promising. We also use K-Fold to find best hyper parameters for further use in the experimental analysis. In this project, our main application will be a uniform prior one, but we also consider imbalanced applications.

III. Multi Variate Gaussian Classifiers

The first classifier that we try to analyze is Multi Variate Gaussian classifier. We try four different types of MVG models and we compare the results together.

A. overview

The normal distribution is perhaps one of the commonly used statistical distribution functions. In this project, we discuss multivariate normal distribution, and how we can derive a generative (more on that later) Gaussian classifier using Bayes' theorem. Classifiers like Gaussian are simple yet intuitive and interpretable.

B. Gaussian Distribution

A Gaussian distribution, also known as a normal distribution, is a bell-shaped probability distribution that is symmetric and defined by its mean and variance. It is widely used in statistics and probability theory due to its mathematical properties and applications in various fields.

$$f\left(x \mid \mu, \sigma^2\right) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{3\sigma^2}} \tag{11}$$

Technically, we call it a probability density of x given by mean and variance. A univariate distribution is suitable when we want to express our uncertainty over a quantity like adult weight. What if we want to express our uncertainty over multiple quantities (joint random variables — adult weight and height) This leads to a multivariate normal distribution, the equation of which is given below.

$$\mathcal{N}(x \mid \mu, \Sigma) \overset{def}{=} \frac{1}{(2\pi)^{p/2}|\Sigma|^{1/2}} \exp\left[-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right] \tag{12}$$

Σ is a covariance matrix. Function symbol N and f are used interchangeably.

C. Maximum Likelihood Estimate (MLE)

Maximum Likelihood Estimation (MLE) is a statistical method that finds the parameter values in a model that maximize the likelihood of observing the given data. It is widely used for parameter estimation and provides reliable

TABLE I
MVG RESULTS

	$SingleFold$			$6-Folds$			$SingleFold$			$6-Folds$		
	$\pi=0.5$	$\pi=0.1$	$\pi=0.9$	$\pi=0.5$	$\pi=0.1$	$\pi=0.9$	$\pi=0.5$	$\pi=0.1$	$\pi=0.9$	$\pi=0.5$	$\pi=0.1$	$\pi=0.9$
	Z-scored features — no PCA						Gaussianized features — no PCA					
Full-Cov	0.161	0.257	0.449	0.117	0.306	0.343	0.155	0.347	0.453	0.137	0.374	0.356
Diag-Cov	0.489	0.851	0.816	0.463	0.804	0.793	0.478	0.849	0.762	0.447	0.780	0.750
Tied Full-Cov	0.155	0.288	0.443	0.111	0.293	0.334	0.146	0.333	0.437	0.126	0.352	0.352
Tied Diag-Cov	0.493	0.842	0.815	0.455	0.797	0.798	0.468	0.844	0.776	0.446	0.789	0.782
	Z-scored features — $PCA=10$						Gaussianized features — $PCA=10$					
Full-Cov	0.201	0.467	0.597	0.184	0.413	0.538	0.211	0.471	0.485	0.188	0.430	0.513
Diag-Cov	0.198	0.492	0.570	0.181	0.405	0.563	0.220	0.464	0.511	0.182	0.422	0.508
Tied Full-Cov	0.192	0.456	0.571	0.181	0.413	0.540	0.204	0.456	0.506	0.180	0.415	0.498
Tied Diag-Cov	0.201	0.453	0.581	0.181	0.392	0.553	0.207	0.486	0.509	0.178	0.417	0.502
	Z-scored features — $PCA=8$						Gaussianized features — $PCA=8$					
Full-Cov	0.272	0.496	0.690	0.262	0.558	0.682	0.295	0.568	0.675	0.266	0.527	0.660
Diag-Cov	0.278	0.559	0.669	0.266	0.519	0.690	0.270	0.524	0.667	0.257	0.529	0.667
Tied Full-Cov	0.275	0.502	0.636	0.259	0.517	0.679	0.279	0.538	0.637	0.256	0.533	0.655
Tied Diag-Cov	0.274	0.485	0.655	0.262	0.492	0.686	0.278	0.530	0.648	0.261	0.514	0.665
	Z-scored features — $PCA=6$						Gaussianized features — $PCA=6$					
Full-Cov	0.309	0.511	0.700	0.297	0.576	0.745	0.300	0.539	0.646	0.282	0.600	0.701
Diag-Cov	0.302	0.604	0.667	0.296	0.588	0.738	0.280	0.607	0.633	0.278	0.617	0.678
Tied Full-Cov	0.313	0.536	0.672	0.295	0.571	0.744	0.299	0.523	0.646	0.277	0.609	0.696
Tied Diag-Cov	0.306	0.566	0.679	0.295	0.583	0.738	0.277	0.577	0.666	0.272	0.614	0.696

estimates based on the observed data.. The likelihood of observed data peaks at the mean value which coincides with sample average. The same process applies to the estimate of variance. Therefore, the MLE of mean and variance:

$$\mu_{ML} = \frac{1}{N}\sum_{i} x_i \tag{13}$$

$$\Sigma_{ML} = \frac{1}{N}\sum_{i=1}^{N} (x_i - \bar{x})\,(x_i - \bar{x})^T \tag{14}$$

D. Gaussian Classifier

In this project we have two classes (1 and 2 — binary classification). The objective here is to predict the class to which new data belongs. In other words, for a given new data(x), we want to estimate $p(y=0 \mid x)$ and $p(y=1 \mid x)$. X is assigned to any class which has the highest probability. Bayes' theorem can help us with estimating $p(y=0 \mid x)$ and $p(y=1 \mid x)$. $p(y=c \mid x)$ is $p(y=0 \mid x)$ and $p(y=1 \mid x)$. $p(x \mid y=c)$ is assumed to be Gaussian/normal distribution. $p(y=c)$ is a class prior which is ratio of number of samples per class to total number of samples. For the purpose of classification, we are only interested in nominator so we can ignore the normalisation constant which is aimed to make class posterior a valid probability distribution. For the curious reader, please refer to the above reference.

All terms in nominator can be estimated from the training dataset. It is called 'Gaussian' classifier because of the assumption that $p(x \mid y=c)$ is Gaussian distribution. It is also known as 'Mixture Gaussian' and 'Discriminant' classifier.

There are two variants of Gaussian classifier, depending on whether covariance matrices of classes are assumed to be equal or not. Covariance matrix assumption has an impact on the class boundary. Shared covariance matrix leads to the linear boundary while separate covariance matrices lead to the quadratic boundary. In all the validations for different classifiers, we consider normalized minimum detection for measuring the performance ($minDCF$).

E. Expectations

Since histograms have shown that features approximately have a Gaussian distribution, it is expected that the generative models work well for this dataset. Furthermore, heatmaps have shown that correlation is significantly spread between the features. Therefore, it is expected that the models based on the Naïve Bayes assumptions will perform badly.

F. Results

As we can see in Table. I, full tied covariance Gaussian classifier and full covariance models have the best performance, and diagonal matrices have the worst performance. In addition, we can see that normalization does not have significant affect in our results. However, we have to analyse more to see how much effective our normalization is. Moreover, the diagonal models will have bad results if we do not consider PCA. In addition, we can see that we have slightly better performance with PCA $(m=10)$ in PCA versions. The results of single-fold and K-fold are inconsistent, suggesting that training whole data will have better performance.

PCA do not brings good results even with 10 features and we have only better scores in PCA models that are based on Naïve assumption, but they aren't remarkable with respect to the full and tied raw features. So, for the discriminative approaches, it is not expected that PCA works well; however, we reported the results anyway.

Tied Models works well, and this suggests that models which exploits linear separation rules are expected to perform effectively. This will be confirmed later. Lastly, Z-score and gaussianized features does not bring remarkable results. We can notice that the ones with diagonal covariance matrices perform worse than full covariance ones, as we have assumed above with the expectations.

TABLE II
MVG RAW FEATURES SCORES

	$SingleFold$		
	$\pi = 0.5$	$\pi = 0.1$	$\pi = 0.9$
Full-Cov	0.126	0.340	0.388
Diag-Cov	0.515	0.784	0.822
Tied Full-Cov	0.125	0.302	0.426
Tied Diag-Cov	0.505	0.784	0.842
	$6 - Folds$		
	$\pi = 0.5$	$\pi = 0.1$	$\pi = 0.9$
Full-Cov	0.115	**0.283**	0.360
Diag-Cov	0.467	0.795	0.789
Tied Full-Cov	**0.113**	0.289	**0.339**
Tied Diag-Cov	0.461	0.790	0.799

We can also see in Fig. II the results within raw features. From the results above we conclude that the best results are within raw features. Again, we have worst results on diagonal versions.

IV. Logistic Regression

This type of statistical model (also known as logarithm model) is often used for classification and predictive analytic. Logistic regression estimates the probability of an event occurring, based on a given dataset of independent variables. Since the outcome is a probability, the dependent variable is bounded between 0 and 1. In logistic regression, a logarithm transformation is applied on the odds—that is, the probability of success divided by the probability of failure.

Since we want to consider different applications (with different priors), we have implemented the prior-weighted version of the Logistic Regression. In other words, the implemented objective function is:

$$R(w) = \frac{\lambda}{2}\|w\|^2 + \frac{\pi_T}{n_T}\sum_{i|z_i=1} l\left(z_i s_i\right) + \frac{1-\pi_T}{n_F}\sum_{i|z_i=-1} l\left(z_i s_i\right) \qquad (15)$$

This version of Logistic Regression applies linear separation rules, but it is possible to define non-linear separation rules by building a certain expanded feature space defined as:

$$\phi(\boldsymbol{x}) = \begin{bmatrix} \text{vec}\left(\boldsymbol{x}\boldsymbol{x}^T\right) \\ \boldsymbol{x} \end{bmatrix} \qquad (16)$$

We can thus train the LR model defined above, but this time using the feature vectors $\phi(x)$ rather than x. It allows computing linear separation rules for $\phi(x)$, and this corresponds to estimate quadratic separation surfaces in the original space.

A. Expectations

Since full tied MVG introduces linear separation rules and has provided good results, it is expected that Logistic Regression will perform well. Furthermore, we do not expect that with 'Gaussianization' or 'Z-scored Normalization' will obtain significantly better results as Logistic Regression does not require specific assumptions on the data distribution.

Lastly, for what concerns the Quadratic version of the Logistic Regression, it is not expected for regularization to have a significant impact on the $minDCF$ computation due to the higher complexity of the model.

B. Results

Linear Logistic Regression; we have the results of linear logistic regression in Table III. We can see that raw features have better results than the gaussianized version.

TABLE III
LINEAR LOGISTIC REGRESSION RESULTS

	$\pi = 0.5$	$\pi = 0.1$	$\pi = 0.9$
MVG (Full-Cov)	0.115	0.283	0.360
MVG (Tied Full-Cov)	0.113	0.289	0.339
Raw features			
Log Reg ($\lambda = 10^{-5}, \pi_T = 0.1$)	0.122	**0.273**	0.392
Log Reg ($\lambda = 10^{-5}, \pi_T = 0.5$)	**0.115**	0.290	0.349
Log Reg ($\lambda = 10^{-5}, \pi_T = 0.9$)	0.116	0.324	**0.324**
Gaussianized features			
Log Reg ($\lambda = 10^{-5}, \pi_T = 0.1$)	0.124	0.320	0.393
Log Reg ($\lambda = 10^{-5}, \pi_T = 0.5$)	0.127	0.362	0.362
Log Reg ($\lambda = 10^{-5}, \pi_T = 0.9$)	0.126	0.358	0.358

The plots show how $minDCF$ is affected by different values of λ. They are exploited to calibrate λ, which is the regularization term.

As we can see in Fig. 5. The tuning of the hyper-parameter λ shows that regularization is not required and brings benefits, with best results obtained for small values of it. Moreover, for $\lambda = 0.1$, the results significantly starts to get worse (for raw features), and for Z-scored Normalization and Gaussianized features around $\lambda = 0.1$. The normalization effects of Z-scored and Gaussianization do not seem much relevant, and it is needed a smaller regularization term than Logistic Regression with raw features.

Quadratic Logistic Regression; now we start analyzing Quadratic Logistic Regression. Now we can see in Fig. 6 regularization is not so helpful for raw features, but it is useful for Z-scored and Gaussianized features.

In the Table. IV we can see that this model gets worse results than the linear one. Using different values of π_T does not improve the classification performance for the specific

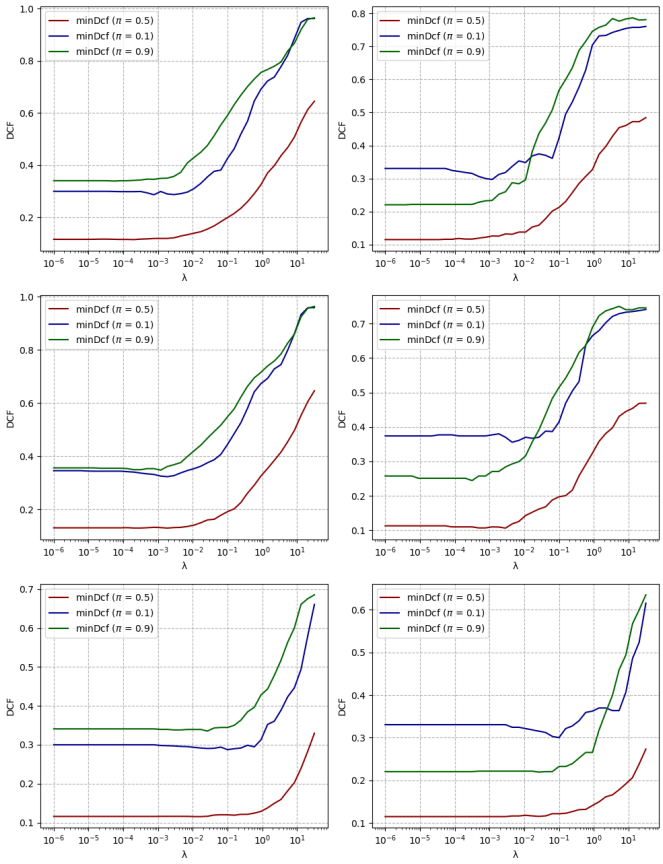

Fig. 5. Linear Log-Reg: $minDCF$ for different values of λ. Top: Z-scored. Middle: Gaussianization. Buttom: No pre-preocessing. Left: 6-Folds. Right: Single-Fold

TABLE IV
QUADRATIC LOGISTIC REGRESSION RESULTS

	$\pi = 0.5$	$\pi = 0.1$	$\pi = 0.9$
Raw features			
MVG (Full-Cov)	0.115	0.283	0.360
MVG (Tied Full-Cov)	0.113	0.289	0.339
Quad Log Reg ($\lambda = 10^{-5}, \pi_T = 0.5$)	**0.134**	**0.331**	**0.336**
Quad Log Reg ($\lambda = 10^{-5}, \pi_T = 0.1$)	0.134	0.367	0.359
Quad Log Reg ($\lambda = 10^{-5}, \pi_T = 0.9$)	0.138	0.360	0.371
Gaussianized features			
MVG (Full-Cov)	0.137	0.374	0.356
MVG (Tied Full-Cov)	0.126	0.352	0.352
Quad Log Reg ($\lambda = 10^{-5}, \pi_T = 0.5$)	0.143	0.362	0.342
Quad Log Reg ($\lambda = 10^{-5}, \pi_T = 0.1$)	0.152	0.420	0.372
Quad Log Reg ($\lambda = 10^{-5}, \pi_T = 0.9$)	0.149	0.333	0.343

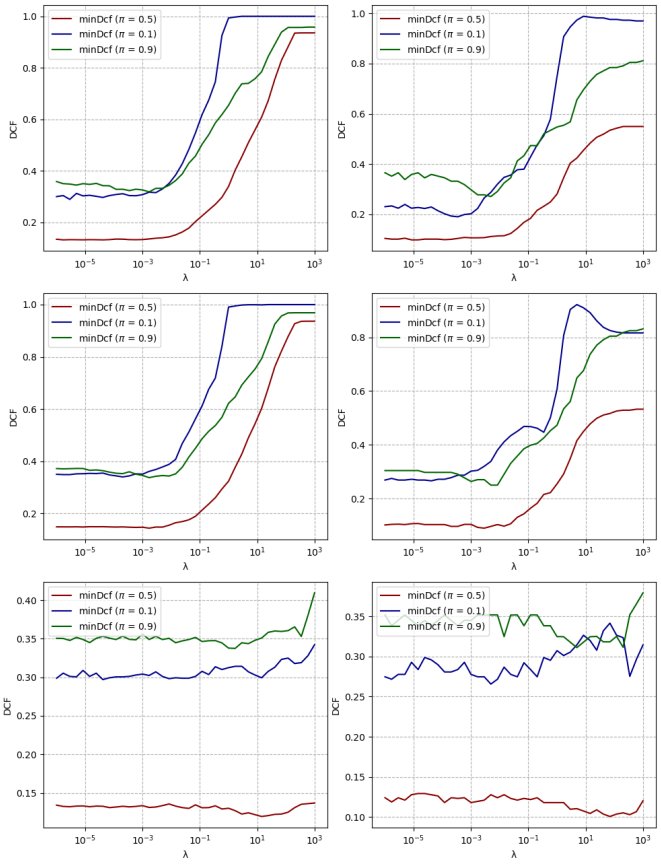

Fig. 6. Quadratic Log-Reg: $minDCF$ for different values of λ. Top: Z-scored. Middle: Gaussianization. Buttom: No pre-preocessing. Left: 6-Folds. Right: Single-Fold

application, and the results for imbalanced applications are pretty similar to each other, even though $\pi_T = 0.5$ performs better than the others.

C. Conclusion

We can conclude that MVG with Tied Covariance Matrix performs similarly. It is reasonably to affirm that Logistic Regression retrieve the same outcomes of the MVG model with linear classification rules. Moreover, different values of π_T does not bring improvements for the other two applications. The study for this application continues with the training of SVMs and to Gaussian-Mixture Models. Lastly, we have better performance on linear seperation.

V. SUPPORT VECTOR MACHINE (SVM)

In the next step, we try to use three different models of the SVM (Linear, RBF and Polynomial). The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space(N — the number of features) that distinctly classifies the data points.

To separate the two classes of data points, there are many possible hyperplanes that could be chosen. Our objective is to find a plane that has the maximum margin, i.e the maximum distance between data points of both classes. Maximizing the

margin distance provides some reinforcement so that future data points can be classified with more confidence. In the SVM algorithm, we are looking to maximize the margin between the data points and the hyperplane. The loss function that helps maximize the margin is hinge loss.

A. Expectations

Up to now, models which exploit linear separation rules have performed very well. So it is expected that Linear SVM will provide good results as well. Results are discussed in below sections.

B. Linear SVM

Linear SVM, obtained by solving the primal problem, expressed as the minimization of:

$$\widehat{J}(\widehat{\boldsymbol{w}}) = \frac{1}{2}\|\widehat{\boldsymbol{w}}\|^2 + C\sum_{i=1}^{n}\max\left(0, 1 - z_i\left(\widehat{\boldsymbol{w}}^T\widehat{\boldsymbol{x}}_i\right)\right) \quad (17)$$

Where:

$$\widehat{\boldsymbol{x}}_i = \begin{bmatrix} \boldsymbol{x}_i \\ K \end{bmatrix}, \quad \widehat{\boldsymbol{w}} = \begin{bmatrix} \boldsymbol{w} \\ b \end{bmatrix} \quad (18)$$

The K is a regularization term which mitigates the effects of the bias term due to the regularization of the norm of w_b. In fact, it is equal to $\widehat{w_b}^2 = w^2 + b^2$.

Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.

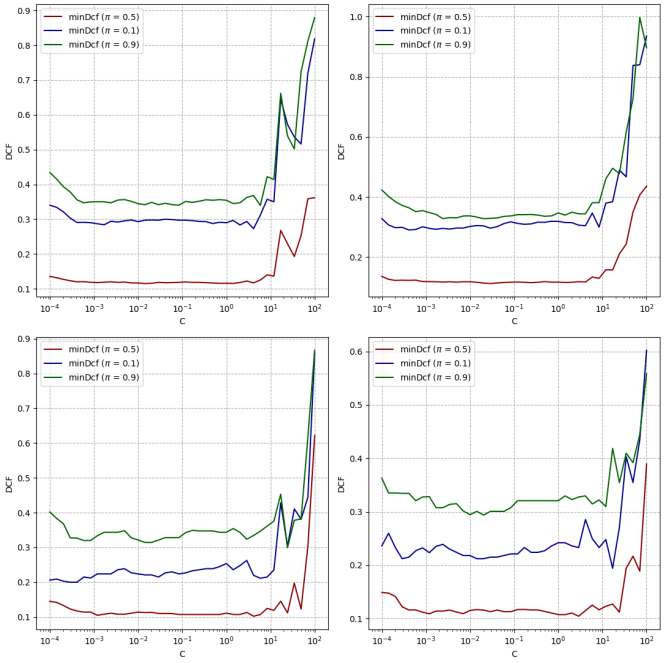

Fig. 7. Linear SVM: $minDCF$ for different values of C. Top: 6-Folds. Buttom: Single-fold. Left: $Balanced$. Right: Standard

The plots show how $minDCF$ is affected by different values of the hyper-parameter of C. They have been generated, as in the previous models, employing the K-Fold with $k = 6$. We can see in Fig. 7 setting $C = 0.01$ seems a reasonable choice. We can also see than imbalanced version of Linear SVM has slightly better results than balanced version with $\pi_T = 0.5$.

Again, the K-fold results are not consistent with the single fold results, suggesting that our model should be trained with all data. Linear SVM performs similarly to other linear approaches, as expected. Class re-balancing is not necessary, therefore we will use the default SVM formulation.

In the Table. V we can see the results of models that following a linear pattern. The SVM model outperforms among the linear models.

TABLE V
LINEAR MODELS RESULTS

	$\pi = 0.5$	$\pi = 0.1$	$\pi = 0.9$
MVG (Tied Full-Cov)	0.113	0.289	0.339
Log Reg ($\lambda = 10^{-5}, \pi_T = 0.5$)	0.115	0.290	0.349
Linear SVM ($C = 10^{-2}$)	**0.112**	**0.286**	0.334
Linear SVM ($C = 10^{-2}, \pi_T = 0.5$)	0.116	0.296	**0.328**

We consider two non-linear SVM formulations. The first will use a **Polynomial Quadratic Kernel** (similar to the quadratic Logistic Regression model). The second will employ a **Radial Basis Function kernel**.

C. Polynomial SVM

Quadratic (polynomial) SVM, obtained by solving the dual problem, expressed as the maximization of:

$$\alpha^T\mathbf{1} - \frac{1}{2}alpha^T H\alpha \quad (19)$$

subject to:

$$0 \leq \alpha_i \leq C, i = 1, ..., n, \quad \sum_{i=1}^{n}\alpha_i z_i \quad (20)$$

Directly solving the formulated expression would increase the scores computation complexity. This is the reason why a kernel function is employed. Specifically, it computes the dot product in the expanded space $\phi_(x_i)^T\phi_(x_j)$. Formally, the kernel function is then defined as $k(x_1, x_2) = \phi_(x_1)^T\phi_(x_2)$. Actually, in order to add a regularized bias in the non-linear SVM, it is sufficient to add a constant value $\zeta = K^2$ to the kernel function:

$$\widehat{k}(x_1, x_2) = k(x_1, x_2) + \zeta \quad (21)$$

It can be computed in different ways. For the polynomial kernel of degree d, it:

$$k(x_1, x_2) = (x_1^T + 1)^d \quad (22)$$

Now we analyze the results of polynomial SVM. For the results we consider degree as a hyper-parameter to see the behaviour of the Quadratic kernel. We can see in Fig. 8 the $minDCF$ changes with respect to the degrees and C

values. This figure corresponds to the Gaussianized features. The 6-folds interpretation and single-fold approaches are not consistent. In the 8 we can see the change of $minDCF$ for different values of $degree$s. For better behaviour of the model, pre-processing technique should be used.

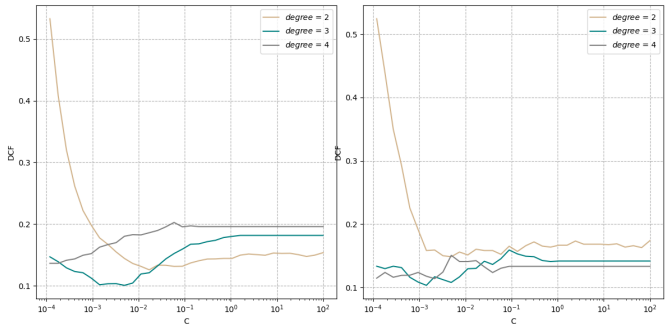

Fig. 8. Polynomial SVM (Gaussianized features) with respect to degrees: $minDCF$ for different values of C. Left: 6-Folds. Right: Single-fold

We also need to see the impact of the different values of π_T on the $minDCF$. In the Fig. 9 we can see that we should avoid low values of c. It has better performance than linear SVM. We choose $degree = 2$ and $C = 10^{-2.3}$.

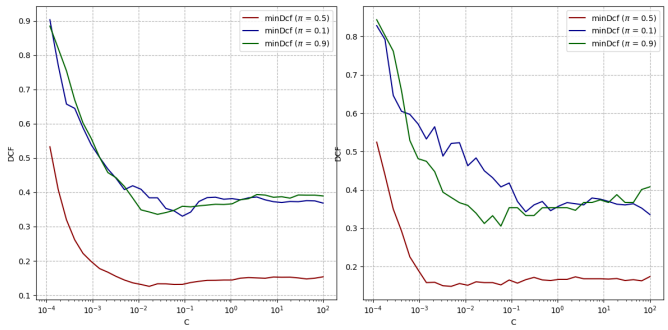

Fig. 9. Polynomial SVM (Gaussianized features) with respect to π_T: $minDCF$ for different values of C. Left: 6-Folds. Right: Single-fold

Now in the Table. VI we compare non-linear models we have analyzed till now. We can see that Quadratic Polynomial SVM outperforms among non-linear models. We choose Gaussianized version of polynomial SVM with $C = 10^{-2.3}$.

TABLE VI
NON-LINEAR MODELS RESULTS

	$\pi = 0.5$	$\pi = 0.1$	$\pi = 0.9$
MVG (Full-Cov)	0.115	**0.283**	0.360
Quad Log Reg ($\lambda = 10^{-5}, \pi_T = 0.5$)	0.134	0.331	0.336
Quad SVM ($C = 10^{-2.3}, degree = 3$)	**0.109**	0.320	**0.320**

The worse performance seems due to class balanced versions. Retraining with the estimated C but with imbalanced we obtain slightly better results.

We conclude that the SVM quadratic discriminative methods are equivalent on this dataset, and both perform better than the MVG classifier. Moreover, quadratic SVM perform better results than quadratic LR. We try anther quadratic kernel which is RBF kernel.

TABLE VII
REBALANCED QUADRATIC SVM RESULTS

	$\pi = 0.5$	$\pi = 0.1$	$\pi = 0.9$
Quad SVM ($C = 10^{-2.3}, \pi_T = 0.5$)	0.111	0.350	**0.318**
Quad SVM ($C = 10^{-2.3}, \pi_T = \pi_T^{emp}$)	**0.109**	**0.320**	0.320

D. Radial Basis Kernel Function SVM

SVM with Radial Basis kernel Function (RBF), obtained by solving the same dual problem as above but, in this case, the kernel function is defined:

$$k(x_1, x_2) = e^{-\gamma||x_1 - x_2||^2} \tag{23}$$

where γ is an hyper-parameter. Since the model provides more complex separation surfaces, it may benefit more than the other approaches from additional data. Now we analyze SVM RBF

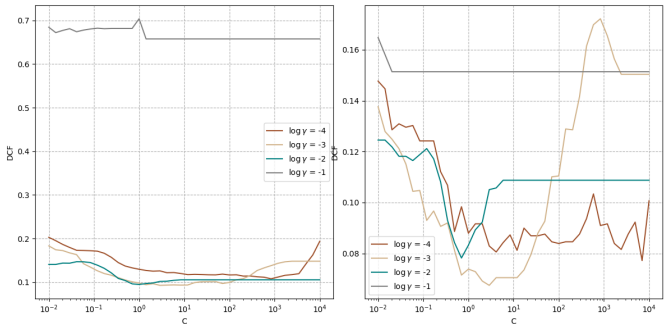

Fig. 10. RBF SVM (raw features) with respect to π_T: $minDCF$ for different values of C. Left: 6-Folds. Right: Single-fold

results. The Fig. 10 is different from the ones generated in the other SVM cases and more like polynomial first figure. In particular, this plot allows to tune two hyper-parameters at the same time ($Cand\gamma$). In this case, C and γ influence the results. The best results are within $\log(\gamma) = 3$. On the other hand, they get worse for greater values of γ. For $\log(\gamma) = -1$ we have the worst performance. Let's continue the analysis for $C = 10$ and $\log(\gamma) = -3$.

Moreover, based on the Fig. 10, we have better performance within single-fold approach. However, the results are not reliable as K-fold approach. We have to see the behaviour of this approach in feature experimental analysis part.

In the Table. VIII we can see that the RBF kernel SVM significantly outperforms our previous models on our cross-validation set.

We also analyze the $minDCF$ of Gaussianized scores. In Fig. 11 unlike the raw features we have best performance on $\log(\gamma) = 1$ for small values of C. For other values of γ, the results are worse than raw features. In the Table. X we can see the $minDCF$ of different re-balanced RBF models with

TABLE VIII
NON-LINEAR MODELS RESULTS COMPARED TO RBF

	$\pi = 0.5$	$\pi = 0.1$	$\pi = 0.9$
MVG (Full-Cov)	0.115	0.283	0.360
Quad Log Reg ($\lambda = 10^{-5}, \pi_T = 0.5$)	0.134	0.331	0.336
Quad SVM ($C = 10^{-2.3}, degree = 3$)	0.109	0.320	0.320
RBF SVM ($\log(C) = 1, \log(\gamma) = -3$)	**0.091**	**0.240**	**0.228**

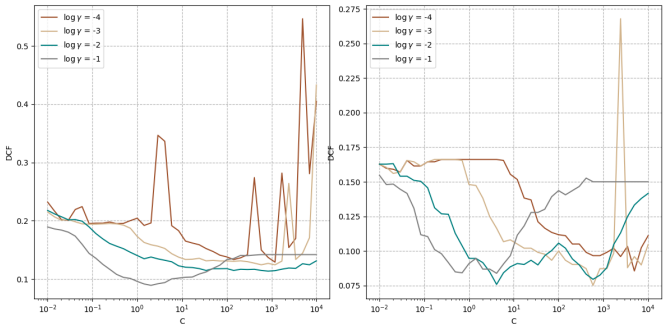

Fig. 11. RBF SVM (Gaussianized features) with respect to π_T: $minDCF$ for different values of C. Left: 6-Folds. Right: Single-fold

raw features with a specific hyperparameter on validation set ($\log(C) = 1, \log(\gamma) = -3$).

TABLE IX
REBALANCED SVM RBF RESULTS

	$\pi = 0.5$	$\pi = 0.1$	$\pi = 0.9$
RBF SVM ($No\ balancing$)	**0.090**	0.240	**0.228**
RBF SVM ($\pi_T = 0.5$)	0.091	**0.224**	0.262
RBF SVM ($\pi_T = 0.1$)	0.111	0.260	0.361
RBF SVM ($\pi_T = 0.9$)	0.097	0.330	0.236

If we are targeting different priors it may be worth training a task specific classifier in this case. In the following we select the model trained with imbalanced classes.

We also repeated the analysis using the gaussianized features. The min DCF results below show the performance of the models that best perform on the validation set ($\log(C) = 1, \log(\gamma) = -3$). We can see the performance of these models in Table. IX.

TABLE X
VALIDATING RBF WITH GAUSSIANIZED FEATURES

	$\pi = 0.5$	$\pi = 0.1$	$\pi = 0.9$
RBF SVM ($Nobalancing$)	**0.092**	**0.263**	**0.242**
RBF SVM ($\pi_T = 0.5$)	0.093	0.269	0.246
RBF SVM ($\pi_T = 0.1$)	0.115	0.274	0.251
RBF SVM ($\pi_T = 0.9$)	0.099	0.269	0.248

E. Conclusion

Among all the different types of SVM models, quadratic models outperforming the results. Moreover, in quadratic kernels, RBF kernel works better than Polynomial version.

VI. GAUSSIAN MIXTURE MODEL (GMM)

GMMs can approximate generic distributions, so we expect to obtain better results than with the Gaussian model. We consider both full covariance and diagonal models, with and without covariance tying.

A. Gaussian Mixture Model (GMM)

As we saw in in MVG, we have obtained best results within full tied covariance matrix. We expect the same here but better results than MVG ones. Furthermore, according to feature distributions in fig. 1, some features have multinomial distributions. These features can indicate different age groups for different genders. Therefore, we expect low $minDCF$ for GMM models.

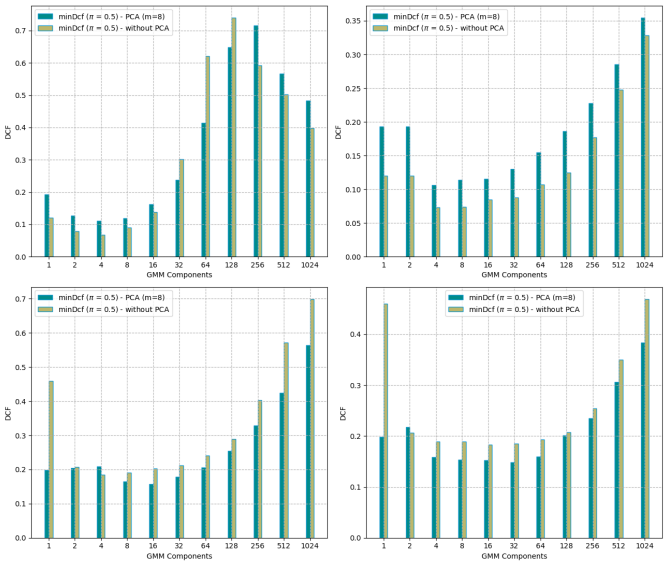

Fig. 12. GMM models. Up: Full. Right: Tied

For tied covariance models, tying takes place at class level (i.e. different classes have different covariance matrices). We use the K-fold protocol to select the number of Gaussians and to compare different models. We can see in Fig. 12 that the best models are within Full and Full-tied models.

In the Table. XI we can compare best models of GMM with SVM RBF from the previous section. The results of GMM are superior in minDCF. All the RBF results are captured with tuned hyperparameters ($\log(C) = 1, \log(\gamma) = -3$).

B. Conclusions

As we saw the results of previous sections, Full tied model perform well. This can be explained by the presence of a large number of clusters that have similar within-class variability. We also do not need high numbder of components which indicates the low number of clusters in the dataset for each class.

TABLE XI
GMM MODELS COMPARED TO RBF

	$\pi = 0.5$	$\pi = 0.1$	$\pi = 0.9$
Raw features			
RBF SVM ($Nobalancing$)	**0.090**	0.240	**0.228**
RBF SVM ($\pi_T = 0.5$)	0.091	**0.224**	0.262
RBF SVM ($\pi_T = 0.1$)	0.111	0.260	0.361
RBF SVM ($\pi_T = 0.9$)	0.097	0.330	0.236
Full Cov, 4 Gau	**0.068**	0.223	**0.185**
Diag Cov, 4 Gau	0.185	0.430	0.475
Tied Full Cov, 4 Gau	0.073	**0.218**	0.213
Tied Diag cov, 16 Gau	0.183	0.455	0.467
Gaussianized features			
Tied Full Cov, 4 Gau	0.179	0.426	0.468
Tied Diag cov, 16 Gau	0.203	0.493	0.572

VII. POST-PROCESSING

Post-processing in machine learning refines model output to enhance performance and align with specific requirements. Techniques like recalibration, thresholding improve accuracy and interpretability. Post-processing ensures transparent, unbiased predictions and empowers informed decision-making. For post-processing we use three major post-processing techniques:

- Near optimal threshold calculation
- Calibration using logistic regression
- Fusion of different classifiers

In each step we try to analyze the benefits of each approach and implementation method. Before we start we need to summarize our previous works. In all of our analysis we use Detection Cost Function (DCF) interpretation.

A detection cost function is a measure used to evaluate the performance of a binary classifier in detection tasks. It quantifies the trade-off between costs associated with different types of classification errors.

By considering the costs, the function helps find an optimal threshold for classification, minimizing overall cost or maximizing utility. We first visualize and compare models we have seen so far. We have achieved better results by far from GMM models.

A DET plot as we can see in Fig. 13 confirms that the GMM model is superior to SVM for many operating points, and for the main application. We consider two of best full GMM models, tied and non-tied. In this part we try to analyze the calibration of our models. Remember that we have selected SVM as the best model among non-GMM models.

Up to now we have considered only $minDCF$ measures which is the cost we would pay if we made optimal decisions for the evaluation set using the recognizer scores. The cost that we actually pay, however, depends on the goodness of the decisions we make using those scores (in the binary case, on the goodness of the threshold we use in practice to perform class assignment). We therefore turn our attention to actual DCFs.

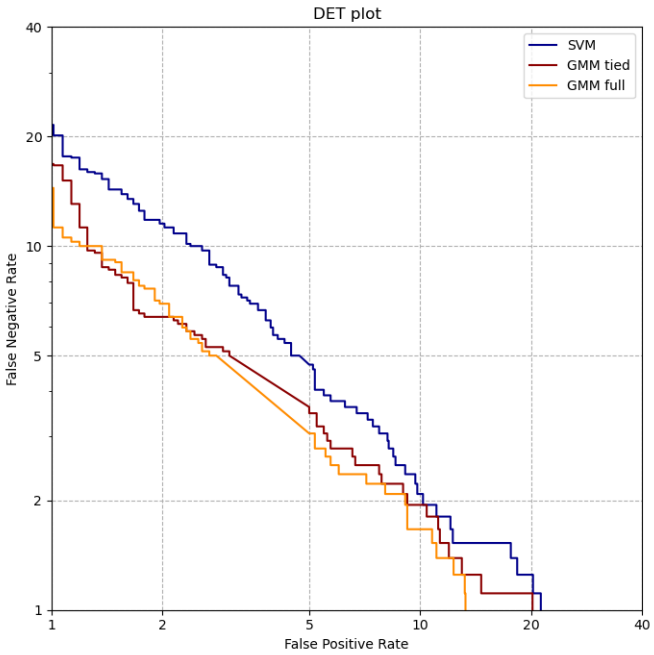

Fig. 13. Det plot for GMM and SVM scores

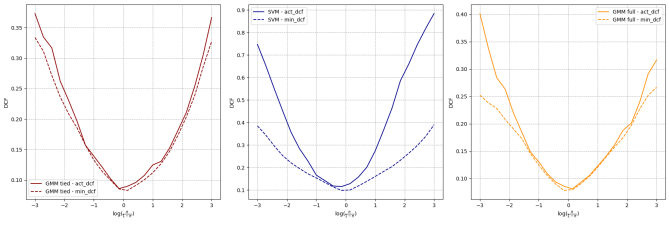

Fig. 14. Bayes error plot for not calibrated scores

In the Table. XII we can see that for different target priors, the $actDCF$ achieved are different from the $minDCF$ in GMM models. We will use first two proposed approaches to recalibrate the scores of these models. Furthermore, we can observe that the SVM provides scores that are not calibrated for our other two applications. Moreover, the model lacking a probabilistic interpretation.

A. Near Optimal Threshold Calculation

We first try to take a look at bayes error plot of our three models in the Fig. 14. We can see that GMM models are well calibrated. Furthermore, the SVM model is mis-calibrated. For the first approach we calculate near optimal threshold for a given application. It won't produce scores that are well-calibrated, we just have to re-estimate an optimal threshold for different applications. The procedure is as follows:

1) Pool the K-fold scores of the classifier (this gives a number of scores equal to the number of samples).
2) Shuffle the scores and then split the shuffled scores into 2 partitions (again, we could use a second K-fold approach if needed).

TABLE XII
ACTDCF AND MINDCF OF THE BEST MODELS

	$\pi = 0.5$		$\pi = 0.1$		$\pi = 0.9$	
	$minDCF$	$actDCF$	$minDCF$	$actDCF$	$minDCF$	$actDCF$
RBF SVM	0.091	0.102	0.224	0.516	0.262	0.602
GMM Full Tied 4 Gau	0.073	0.076	0.218	0.260	0.213	0.226
GMM Full 4 Gau	0.068	**0.071**	0.223	**0.243**	0.185	**0.212**

3) Estimate the threshold on one partition, apply the threshold on the remaining one and compare minimum and actual DCFs over the latter.

TABLE XIII
CALIBRATED SCORES WITH OPTIMAL THRESHOLD CALCULATION

	$minDCF$	$actDCF$ $t = -\log \frac{\tilde{\pi}}{1-\tilde{\pi}}$	$actDCF$ t^*
		$\pi = 0.9$	
RBF SVM	0.305	0.520	**0.325**
GMM Full Tied 4 Gau	0.148	0.275	**0.263**
GMM Full 4 Gau	0.260	0.291	0.291
		$\pi = 0.1$	
RBF SVM	0.191	0.620	**0.342**
GMM Full Tied 4 Gau	0.184	**0.301**	0.308
GMM Full 4 Gau	0.226	0.315	**0.313**
		$\pi = 0.5$	
RBF SVM	0.082	0.095	**0.094**
GMM Full Tied 4 Gau	0.042	**0.054**	0.056
GMM Full 4 Gau	0.095	**0.105**	0.111

As we can see in Table. XIII we have improved SVM for different applications.

TABLE XIV
CALIBRATED SCORES OF FULL-TIED GMM USING LOGISTIC REGRESSION

	$minDCF$ $(\tilde{pi} = 0.5)$	$minDCF$ $(\tilde{pi} = 0.1)$	$minDCF$ $(\tilde{pi} = 0.9)$
	0.070	0.244	0.231
	$actDCF$ $(\tilde{pi} = 0.5)$	$actDCF$ $(\tilde{pi} = 0.1)$	$actDCF$ $(\tilde{pi} = 0.9)$
Uncalibrated	**0.074**	0.259	**0.231**
Log Reg ($\tilde{pi} = 0.5$)	0.075	**0.252**	0.240
Log Reg ($\tilde{pi} = 0.1$)	0.147	0.423	0.345
Log Reg ($\tilde{pi} = 0.9$)	0.135	0.397	0.381

B. Calibration using logistic regression

The above results are good but not better than the previously compared models. Furthermore, the actDCF curve indicates that scores are mis-calibrated. It was an expected fact, since SVM scores have no probabilistic interpretation.

This is even more remarkable in the following case where Radial Basis kernel Function has been used in Fig. 14. In this case, scores are highly uncalibrated. It means that they can be improved by applying a transformation function f to previously computed scores, in order to obtain calibrated scores $s_{cal} = f(s)$. To estimate f, discriminative score models, such as Logistic Regression, can be used by passing the scores to the model. They would act as if they were a feature of the model. Following this path, $f(s)$ can be interpret as the log-likelihood ratio:

$$f(s) = \log \frac{f_{S|C}(s|H_T)}{f_{S|C}(s|H_F)} = \alpha s + \beta \tag{24}$$

while the class posterior probability for a given prior $\tilde{\pi}$ would be equal to:

$$\log \frac{P(C = H_T|s)}{P(C = H_F|s)} = \alpha s + \beta + \log \frac{\tilde{\pi}}{1 - \tilde{\pi}} \tag{25}$$

where $\tilde{\pi}$ is a model parameter which represents the application we are optimizing the scores for. Finally, to retrieve the calibrates scores, $f(s)$ can thus be computed as:

$$f(s) = \alpha s + \beta = \alpha s + \acute{\beta} - \log \frac{\tilde{\pi}}{1 - \tilde{\pi}} \tag{26}$$

Now we use log-reg calibration method on the three models we discussed before. In Fig. 15 we can see the results are now well calibrated. We can see huge effect of this calibration method on SVM results. This significant improvement can be due to a new probabilistic interpretation for SVM.

TABLE XV
CALIBRATED SCORES OF SVM USING LOGISTIC REGRESSION

	$minDCF$ $(\tilde{pi} = 0.5)$	$minDCF$ $(\tilde{pi} = 0.1)$	$minDCF$ $(\tilde{pi} = 0.9)$
	0.093	0.268	0.277
	$actDCF$ $(\tilde{pi} = 0.5)$	$actDCF$ $(\tilde{pi} = 0.1)$	$actDCF$ $(\tilde{pi} = 0.9)$
Uncalibrated	0.105	0.482	0.681
Log Reg ($\tilde{pi} = 0.5$)	**0.095**	**0.299**	**0.303**
Log Reg ($\tilde{pi} = 0.1$)	0.201	0.535	0.458
Log Reg ($\tilde{pi} = 0.9$)	0.178	0.470	0.473

Moreover, we can fuse the scores of different classifiers and stack them together and pass them to log-reg classifier to get fusion scores.

C. Fusion of different classifiers

Furthermore, the method allows easily combining the outputs of different classifiers. We can expect that different classifiers agree on some, but not all decisions, especially

TABLE XVI
FUSION SCORES COMPARED TO PREVIOUS RESULTS

	$\pi = 0.5$		$\pi = 0.1$		$\pi = 0.9$	
	$minDCF$	$actDCF$	$minDCF$	$actDCF$	$minDCF$	$actDCF$
RBF SVM	0.091	0.102	0.224	0.516	0.262	0.602
GMM Full Tied 4 Gau	0.073	0.076	0.218	0.260	0.213	0.226
GMM Full 4 Gau	0.068	**0.071**	0.223	**0.243**	0.185	0.212
Fusion	**0.067**	0.075	0.244	0.265	0.208	**0.209**

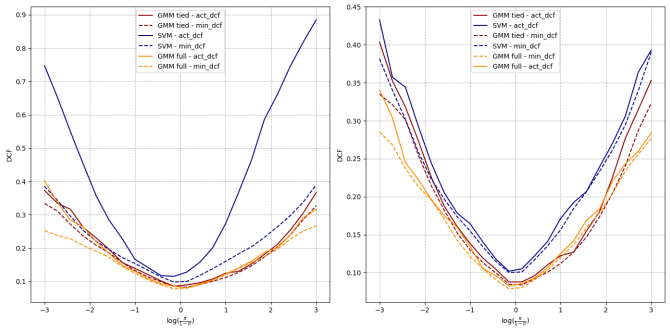

Fig. 15. Bayes error plot for calibrated scores

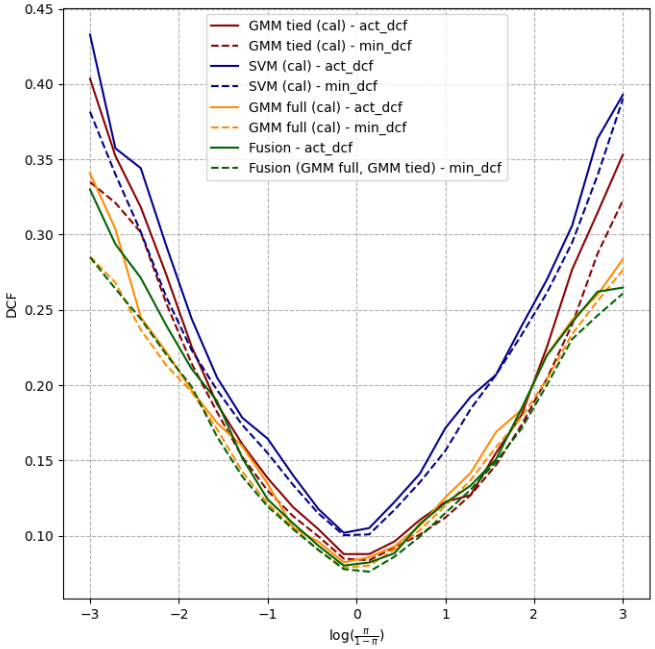

Fig. 16. Bayes error plot for fused gmm full (tied and non-tied)

when classifiers use different approaches and / or different features. Combining the decisions of two classifiers in this case may lead to improvement. In Table. XVI we can see the results within fusion model.

An effective fusion approach consists in performing score-level fusion, rather than decision-level voting. We assume that the fused score is a function of the scores of the different

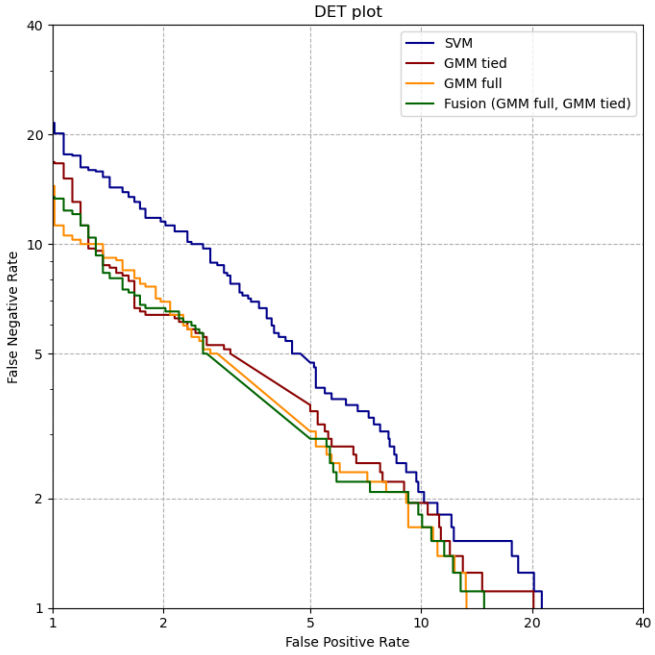

Fig. 17. DET plot for calibrated scores and fusion

classifiers If $s_{t,A}$ and $s_{t,B}$ are the scores of classifiers A and B for sample x_t, then the fused score for sample x_t will be:

$$s_t = f(s_{t,A}, s_{t,B}) \tag{27}$$

We will then use the fused score s_t to perform the decision (i.e. to label the sample x_t) As for calibration, we can assume a simple linear form for f :

$$f(s_{t,A}, s_{t,B}, s_{t,C}, ...) = \alpha_A s_{t,A} + \alpha_B s_{t,B} + \alpha_C s_{t,C} + ... + \beta \tag{28}$$

Again, we can use prior-weighted logistic regression to train the model parameters. The resulting scores will, hopefully, have higher accuracy and benefit from re-calibrating effects of the model.

In Fig, 16 we can see small improvement of fusion on $minDCF$. For the fusion we used the scores of tied and non-tied GMM models. Both models are trained with raw features.

We can now visualize the DET plot in Fig. 17. The results refer to the other partition. The plot differs from the previous DET plot since we are evaluating on different subset of

TABLE XVII
MVG EXPERIMENTAL RESULTS

	80% data			All data			80% data			All data		
	$\pi=0.5$	$\pi=0.9$	$\pi=0.1$	$\pi=0.5$	$\pi=0.9$	$\pi=0.1$	$\pi=0.5$	$\pi=0.9$	$\pi=0.1$	$\pi=0.5$	$\pi=0.9$	$\pi=0.1$
	Z-scored features — no PCA						Gaussianized features — no PCA					
Full-Cov	0.122	0.318	0.321	0.120	0.312	0.312	0.141	0.338	0.339	0.140	0.341	0.347
Diag-Cov	0.427	0.702	0.816	0.434	0.705	0.828	0.412	0.686	0.770	0.423	0.685	0.770
Tied Full-Cov	0.116	0.308	0.304	0.116	0.306	0.306	0.134	0.335	0.334	0.132	0.327	0.336
Tied Diag-Cov	0.426	0.711	0.814	0.433	0.713	0.816	0.414	0.688	0.786	0.425	0.697	0.793
	Z-scored features — $PCA=10$						Gaussianized features — $PCA=10$					
Full-Cov	0.183	0.416	0.445	0.179	0.412	0.433	0.187	0.445	0.450	0.181	0.429	0.430
Diag-Cov	0.180	0.439	0.422	0.175	0.429	0.417	0.178	0.418	0.439	0.174	0.415	0.435
Tied Full-Cov	0.175	0.420	0.418	0.174	0.409	0.416	0.175	0.411	0.431	0.170	0.401	0.423
Tied Diag-Cov	0.177	0.436	0.418	0.174	0.425	0.416	0.179	0.429	0.436	0.174	0.421	0.431
	Z-scored features — $PCA=8$						Gaussianized features — $PCA=8$					
Full-Cov	0.255	0.587	0.593	0.256	0.582	0.591	0.256	0.603	0.600	0.252	0.591	0.582
Diag-Cov	0.257	0.594	0.576	0.254	0.591	0.573	0.253	0.587	0.580	0.251	0.581	0.578
Tied Full-Cov	0.249	0.582	0.593	0.249	0.580	0.570	0.250	0.591	0.576	0.248	0.583	0.576
Tied Diag-Cov	0.257	0.592	0.568	0.255	0.580	0.571	0.256	0.595	0.570	0.252	0.582	0.574
	Z-scored features — $PCA=6$						Gaussianized features — $PCA=6$					
Full-Cov	0.296	0.670	0.664	0.301	0.669	0.663	0.282	0.632	0.641	0.282	0.632	0.642
Diag-Cov	0.296	0.676	0.639	0.299	0.671	0.641	0.283	0.628	0.636	0.283	0.628	0.639
Tied Full-Cov	0.297	0.666	0.645	0.297	0.668	0.646	0.280	0.623	0.627	0.281	0.616	0.631
Tied Diag-Cov	0.294	0.683	0.641	0.298	0.673	0.646	0.279	0.643	0.628	0.280	0.642	0.638

samples (the validation portion of score calibration training protocol). The results are slightly better.

In the Table. XVI we have final results with the validation of our training models. The main models are GMM Full with 4 components and GMM Full-Tied also with 4 components. In the next section we try to test our beliefs about the behaviour of our models and dataset.

VIII. EXPERIMENTAL ANALYSIS

A. Introduction

We now turn the attention to the evaluation part. It is useful to choose the minimum DCF to evaluate because it is an optimistic estimation for the actual DCF in the hypothetical case in which it could be chosen the optimal threshold for the evaluation set. Since validation has been performed with 5-fold cross-validation, the evaluation part will be executed using the whole dataset (all training data for training, all test data for test). The Table, XVII refers to the evaluation results for MVG.

The results are consistent with those obtained in the validation part. We can confirm that the MVG model with tied covariance is the best option to choose, and this also confirms that linear separation rules are the best picks. PCA is not outperforming with respect to the previous, even if it improves the MVG models with diagonal covariance. Gaussianization worsen the data (compared to raw features). However, it performs slightly better than z-scored.

Overall, normalization scores are better than what we saw in validation. We can conclude that for normalization we need all the data.

TABLE XVIII
EXPERIMENTAL MVG RAW FEATURES SCORES

	80% data		
	$\pi=0.5$	$\pi=0.9$	$\pi=0.1$
Full-Cov	0.121	0.313	0.313
Diag-Cov	0.412	0.705	0.796
Tied Full-Cov	0.118	0.307	0.309
Tied Diag-Cov	0.413	0.710	0.803
	All data		
	$\pi=0.5$	$\pi=0.9$	$\pi=0.1$
Full-Cov	**0.120**	**0.312**	**0.312**
Diag-Cov	0.434	0.705	0.818
Tied Full-Cov	**0.116**	**0.306**	**0.306**
Tied Diag-Cov	0.433	0.713	0.816

In Table. XVIII we can see the results obtained with raw features. Again, we have have consistent results with validation models.

B. Logistic Regression

We now turn our attention to the Logistic Regression models with estimated $\lambda = 10^{-5}$. In Fig. 18 we see the results of **Linear Logistic Regression**. For what concerns the DCF curves, they confirms our expectations since validation and evaluation ones have the same trend, and confirms that our choice of $\lambda = 10^{-5}$ has given quite remarkable results. We have better results with $\pi_T = 0.9$ compared to validation. Again, it indicates that having all data for training is needed.

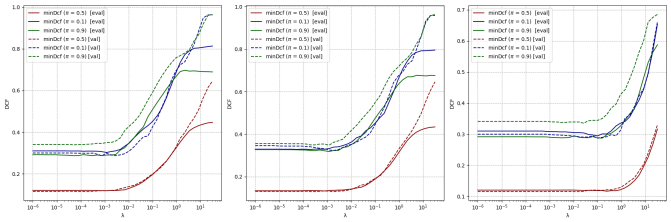

Fig. 18. Experimental Linear Log-Reg: $minDCF$ for different values of λ. Left: Z-scored. Middle: Gaussianization. Right: No pre-preocessing.

We now move on the analysis for different values of π_T. In Table. XX Results are consistent with our expectations. Linear models performs similar to the MVG linear classifiers. The outcomes are slightly worse but very similar than the expectations, and changing target prior is not helpful at all since they are similar to each other, with a little improvement for $\pi_T = 0.1$ and $\pi_T = 0.9$ but not significant.

TABLE XIX
EXPERIMENTAL LINEAR LOGISTIC REGRESSION RESULTS

	80% data			All data		
	$\pi = 0.5$	$\pi = 0.9$	$\pi = 0.1$	$\pi = 0.5$	$\pi = 0.9$	$\pi = 0.1$
Raw features						
LR($\pi_T = 0.5$)	0.120	0.303	0.307	**0.120**	0.305	0.301
LR($\pi_T = 0.1$)	0.122	0.346	0.298	0.122	0.339	**0.296**
LR($\pi_T = 0.9$)	0.129	0.284	0.350	0.123	**0.279**	0.339
Gaussianized features						
LR($\pi_T = 0.5$)	0.132	0.347	0.338	0.133	0.327	0.326
LR($\pi_T = 0.1$)	0.137	0.332	0.334	0.135	0.329	0.330
LR($\pi_T = 0.9$)	0.136	0.313	0.348	0.131	0.313	0.332

We repeat the analysis for **Quadratic Logistic Regression**. As we can see in Fig. 19 model trained with raw features do not retrieve the same trend on the evaluation and validation set. We have better results in evaluation method than validation part. However, we can conclude that our choice of $\lambda = 10^{-5}$ was effective also in this case. We now turn the attention to the $minDCF$ for different values of λ on the evaluation set.

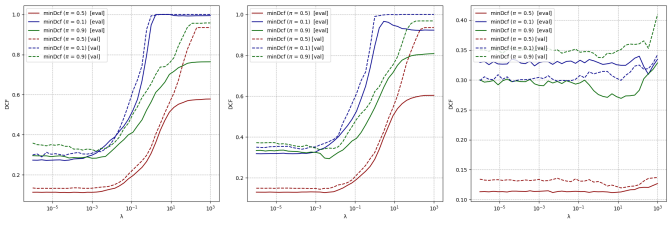

Fig. 19. Experimental Quadratic Log-Reg: $minDCF$ for different values of λ. Left: Z-scored. Middle: Gaussianization. Right: No pre-preocessing.

Moreover, the outcomes of Z-score Normalization and Gaussianization for the evaluation set are consistent with those on the validation one except for the large values of λ. Also we retrieve quite better results than Linear Logistic Regression.

C. Support Vector Machine

Briefly, for each implemented version of SVM, the selected values for the model parameters, during the previous phase, are:

- Linear SVM: Raw features, $C = 1.0$
- Polynomial SVM: Gaussianized normalized features, $degree = 3$, $\log(C) = -2.3$
- SVM RBF: Raw features, $C = 10$, $\log(\gamma) = -3$

We now analyze the consistency of our validation. Let's turn the attention to assessing whether the selected hyper-parameters values have been right or not. In the Table. XXI we can see the experimental results of linear and quadratic SVM. The results for different priors both for balanced and not-balanced SVMs are consistent with training.

TABLE XX
EXPERIMENTAL QUADRATIC LOGISTIC REGRESSION RESULTS

	80% data			All data		
	$\pi = 0.5$	$\pi = 0.9$	$\pi = 0.1$	$\pi = 0.5$	$\pi = 0.9$	$\pi = 0.1$
Raw features						
LR($\pi_T = 0.5$)	0.117	0.335	0.317	**0.112**	0.303	0.322
LR($\pi_T = 0.1$)	0.123	0.362	0.327	0.113	0.342	**0.316**
LR($\pi_T = 0.9$)	0.136	0.318	0.358	0.123	**0.298**	0.352
Z-scored features						
LR($\pi_T = 0.5$)	0.120	0.311	0.284	0.115	0.307	0.278
LR($\pi_T = 0.1$)	0.133	0.341	0.329	0.132	0.337	0.331
LR($\pi_T = 0.9$)	0.128	0.303	0.327	0.287	0.287	0.273

TABLE XXI
EXPERIMENTAL LINEAR AND POLYNOMIAL SVM RESULTS

	All data		
	$\pi = 0.5$	$\pi = 0.9$	$\pi = 0.1$
Raw features			
Linear SVM	0.123	**0.293**	0.317
Linear SVM ($\pi_T = 0.5$)	**0.119**	0.296	**0.307**
Gaussianized features			
Quad SVM	**0.104**	**0.236**	**0.317**
Quad SVM ($\pi_T = 0.5$)	0.109	0.270	0.321

In the Fig. 20 despite the differences in some operating points, as far as the selected hyper-parameters, there are no differences so they have been effective. Again we can see that results are consistent with the validation.

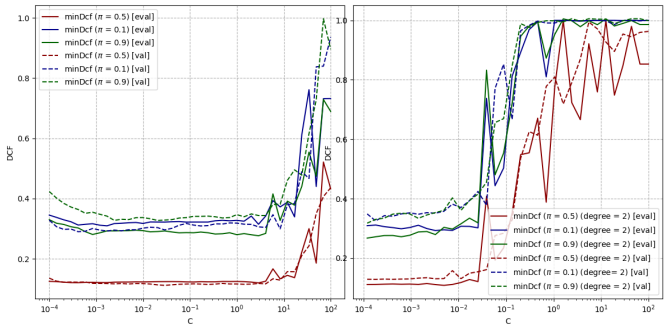

Fig. 20. Experimental quad Log-Reg: $minDCF$ for different values of λ. Left: Linear SVM. Right: Polynomial SVM.

Models perform similarly to logistic regression, the latter being slightly better for our target application. Class rebalancing is not necessary in this case. The choice of hyperparameter C was effective both for linear and quadratic models.

In Fig. 21 we have thre results with respect to $degree$ and C values. K-fold apprach in validation is consistent with evaluation but with single-fold we have slightly different results.

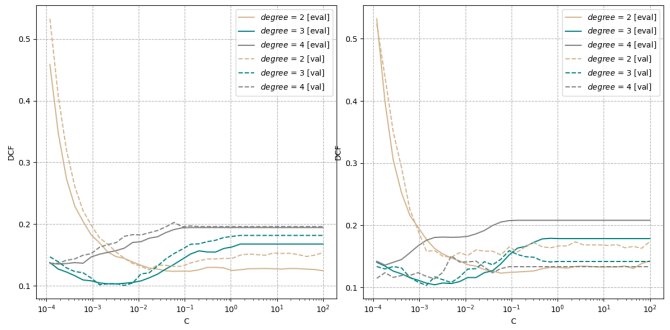

Fig. 21. Experimental quad Log-Reg: $minDCF$ for different values of $degree$ and c with gaussianized features. Left: All data. Right: 80% of data.

Now we look one of the models in our primary system. SVM RBF clearly performed better than its counterparts in terms of minDCF on validation set. In the Table. XXII we can see the results of SVM RBF. The results are consistent with what we saw in training part.

Among all the SVM models, best results are obtained with the RBF version of SVM. We have chosen this model as one of the models in our primary system as the best performance among Non-GMM models.

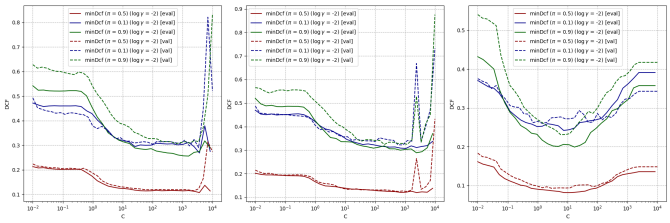

Fig. 22. Experimental rbf svm: $minDCF$ for different values of λ. Left: Z-scored. Middle: Gaussianized. Right: Raw features

In Fig. 23 we see the results with respect to different values of γ and C. The results are consistent with the validation. Raw features achieving better results in terms of $minDCF$.

In the validation part, we had better results with single-fold approach. However, the results were not reliable. As we expected, the results of training 80% of data are not consistent with single-fold approach that we observed during validation.

In Fig. 22 we again have consistent results with the validation.

TABLE XXII
EXPERIMENTAL RBF SVM RESULTS

	Raw features		
	$\pi = 0.5$	$\pi = 0.9$	$\pi = 0.1$
	80% data		
RBF SVM *No balancing*	0.091	0.214	0.257
RBF SVM ($\pi_T = 0.5$)	0.085	0.230	0.224
RBF SVM ($\pi_T = 0.1$)	0.106	0.312	0.242
RBF SVM ($\pi_T = 0.9$)	0.108	0.211	0.353
	All data		
RBF SVM *No balancing*	0.097	0.199	0.342
RBF SVM ($\pi_T = 0.5$)	**0.081**	0.215	**0.221**
RBF SVM ($\pi_T = 0.1$)	0.101	0.324	0.227
RBF SVM ($\pi_T = 0.9$)	0.083	**0.207**	0.244

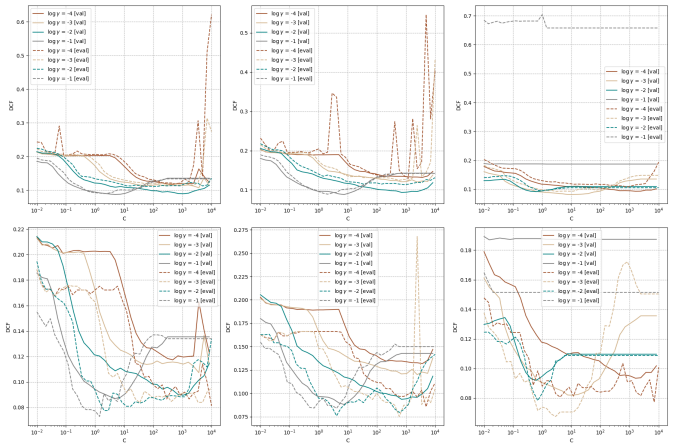

Fig. 23. Experimental rbf svm: $minDCF$ for different values of λ and C. Left: Z-scored. Middle: Gaussianized. Right: Raw features. Up: All-data. Bottom: 80% of data

D. Gaussian Mixture Models (GMM)

We now consider the GMM classifiers. Again, we consider min DCF on the evaluation set of models trained with different number of Gaussians.

As we can see in Fig 24 the optimal number of components is similar between evaluation and validation set:

- For diagonal models, the 16-components version achieves the best results
- For full-covariance models, the 4-components model is slightly better

As we can see in Fig. 24 the results are consistent with validation set. Moreover, in Table XXIII we demonstrated the best results within experimental evaluation of GMM models. We have the models that we expected in validation part. We choose RBF from non-GMM models and full (tied and non-tied) verrsion for final evaluation.

E. Fusion

Now we start evaluating our beliefs about the post-processing model evaluation. First we start with fusion model. In Table. XXIV we see the fusion models work slightly better.

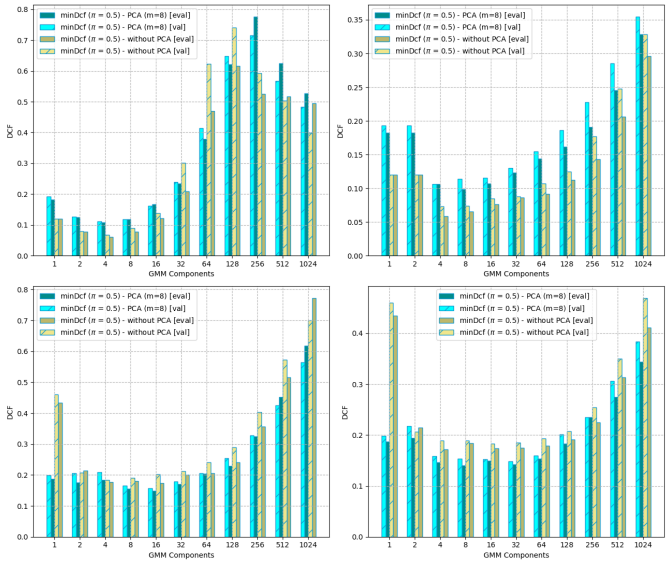

Fig. 24. Experimental gmm models. Up: Full. Right: Tied

TABLE XXIII
EXPERIMENTAL GMM MODELS. UP: FULL. RIGHT: TIED

	Raw features		
	$\pi = 0.5$	$\pi = 0.9$	$\pi = 0.1$
	80% data		
4-G Full-Tied ($\pi_T = 0.5$)	0.06	0.170	0.195
4-G Full ($\pi_T = 0.1$)	0.062	0.163	0.159
	All data		
4-G Full-Tied ($\pi_T = 0.5$)	**0.058**	0.169	0.193
4-G Full ($\pi_T = 0.1$)	0.061	**0.156**	**0.156**

As we did fusion of GMM Tied and Non-Tied, we do it the same for evaluation part. The $minDCF$ have been used for this comparison.

TABLE XXIV
EXPERIMENTAL FUSION SCORES COMPARED TO PREVIOUS RESULTS

	$\pi = 0.5$	$\pi = 0.9$	$\pi = 0.1$
RBF SVM	0.081	0.215	0.221
GMM Full Tied 4 Gau	0.058	0.169	0.193
GMM Full 4 Gau	0.061	0.156	0.156
Fusion	**0.057**	**0.155**	**0.155**

In Fig. 25 we have the DET plot for the best chosen models and fusion. Fusion model slightly works better in terms of $minDCF$. It also provides better results than what we expected from validation part.

F. Score Calibration

In the previous part of the analysis we have considered both threshold optimization on validation sets and a score calibration approach. In this section we try to analyze the effectiveness of both approaches. As we mentioned, actual DCFs

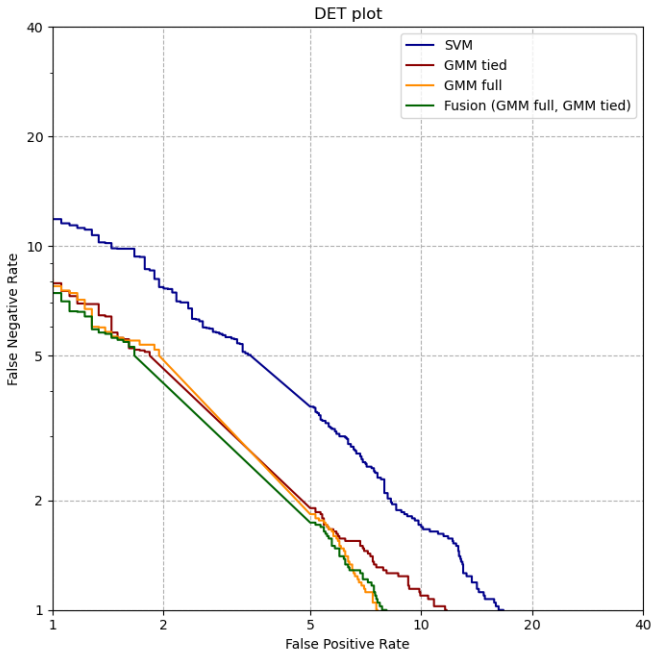

Fig. 25. DET for fusion model and other best performanced models

are computed using the actual decisions (i.e. the actual class predictions) we make for evaluation data. Class predictions require that we compare scores with a threshold. If scores are calibrated, then we have shown that the optimal threshold depends only on the cost of errors and the class. We take the threshold that gives the minimum DCF for our application on the validation set.

TABLE XXV
EXPERIMENTAL THRESHOLD CALCULATION

	$\pi = 0.5$	$\pi = 0.9$	$\pi = 0.1$
	$minDCF$		
RBF SVM	0.082	0.206	0.243
GMM Full Tied 4 Gau	0.059	0.169	0.193
GMM Full 4 Gau	0.061	0.155	0.155
	raw scores - actDCF t^* (re-estimated for each application)		
RBF SVM	0.096	0.220	0.256
GMM Full Tied 4 Gau	0.061	0.180	0.215
GMM Full 4 Gau	0.065	0.181	**0.158**
	calibrated scores - actDCF $t = -\log \frac{\tilde{\pi}}{1-\tilde{\pi}}$		
RBF SVM	**0.083**	**0.218**	**0.252**
GMM Full Tied 4 Gau	**0.059**	**0.178**	**0.204**
GMM Full 4 Gau	**0.061**	**0.173**	0.173

We have seen that the first approach is already effective on the validation data, but the second approach provides better results and produces scores that are calibrated over a wide range of applications. In Table. XXV we see the results after these two approaches.

TABLE XXVI
FUSION SCORES COMPARED TO PREVIOUS RESULTS

	$\pi = 0.5$		$\pi = 0.1$		$\pi = 0.9$	
	$minDCF$	$actDCF$	$minDCF$	$actDCF$	$minDCF$	$actDCF$
RBF SVM (cal)	0.082	0.083	0.243	0.252	0.206	0.219
GMM Full Tied 4 Gau (cal)	0.059	**0.059**	0.193	0.203	0.169	0.178
GMM Full 4 Gau (cal)	0.061	0.061	0.156	0.173	0.156	0.174
Fusion	**0.058**	0.060	**0.155**	**0.170**	0.156	**0.162**

Both approaches are quite effective in this case. The actual DCFs are almost always very close to the optimal $minDCFs$. As it can be seen from the Bayes error plot in Fig. 26, score calibration, however, does not require re-estimation of a different threshold for other applications, as scores are well calibrated over a wide range of applications.

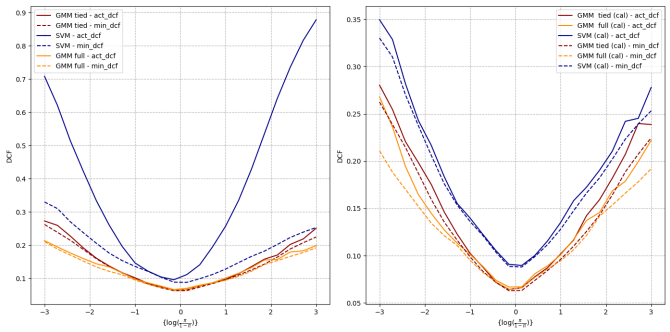

Fig. 26. Experimental calibration bayes error plot

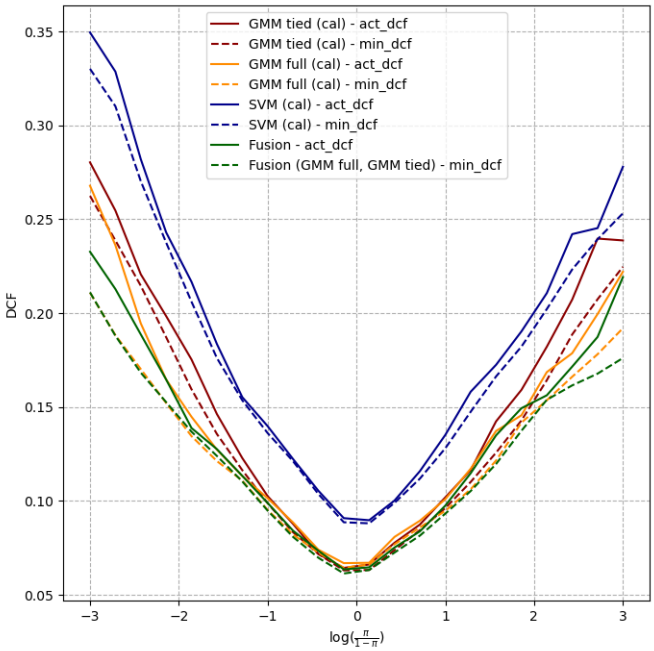

Fig. 27. Experimental calibration bayes error plot with fusion

We also pretty aware that fusion model also produce well-calibrated scores, we only have to assess whether the actual DCFs computed using the theoretical thresholds are close to the minimum ones. In Fig. 27 we can see the error bayes plot for fusion and other best models. We can see that fusion models works better in terms of minDCF.

We can also take a close look at minDcf and actDCF for our applications. In the Table. XXVI we have best models scores (calibrated version). We can see the fusion model wins for our primary application.

IX. CONCLUSION

The approach discussed for final model, consisting of the fusion of two subsystems, is effective, and produces well-calibrated scores for a wide range of applications. However, the models are can also be effective for applications with imbalanced costs and prior proportions. Overall, the similarity between validation and evaluation results suggests that the evaluation population is sufficiently similar to the training population. The choices we made on our training / validation sets proved effective also for the evaluation data.

REFERENCES

[1] Bishop, Christopher M. ‘Pattern Recognition and Machine Learning”. New York :Springer, 2006.
[2] S. Cumani ”Machine Learning and Pattern Recognition”. Politecnico di Torino, materials and slides 2023,