MLPR 2023 – Fingerprint spoofing detection

Gabriele Lucca - Matteo Martini
(s314297- s314786)

1. Introduction

In the modern computerized biometric authentication system, a fingerprint spoof detector is utilized to differentiate the real and fake human finger.
A spoofing attack happens in the field of biometrics when an attacker mimics another person's biometric attribute in an attempt to undermine a biometric authentication system. For instance, a fake finger may be created by using widely accessible materials such as latex, adhesive, and gelatin, and the fingerprint can be added subsequently. The patterns of a person are imprinted on the surface. An attacker can mimic the owner of the genuine ridges by using a fake finger on a fingerprint sensor.[1]

Machine Learning tools are being used to automatically label candidates to facilitate rapid analysis. Classification systems in particular are being widely adopted, which treat the candidate data sets as binary classification problems. The primary objective is to determine whether a fingerprint is authentic or a fake using the provided dataset.

Spoofing a fingerprint can be achieved through various methods, resulting in six distinct sub-classes within the given database. However, these sub-classes are not specified and are simply classified as "spoofed" (class 0).

In total, there are 2325 records in the dataset, consisting of 800 authentic fingerprint data and 1525 spoofed fingerprint data. Each record is characterized by a set of 10-dimensional features.

The target application working point is defined by the following triplets: ($\pi = 0.5$, $C_{fn} = 1$, $C_{fp} = 10$).

2. Features analysis

Every candidate is represented by 10 continuous variables, but these components lack a physical interpretation.
In order to improve estimation accuracy, the dataset has been centered using its mean value.

Histogram and 2D scatter plots of the dataset features

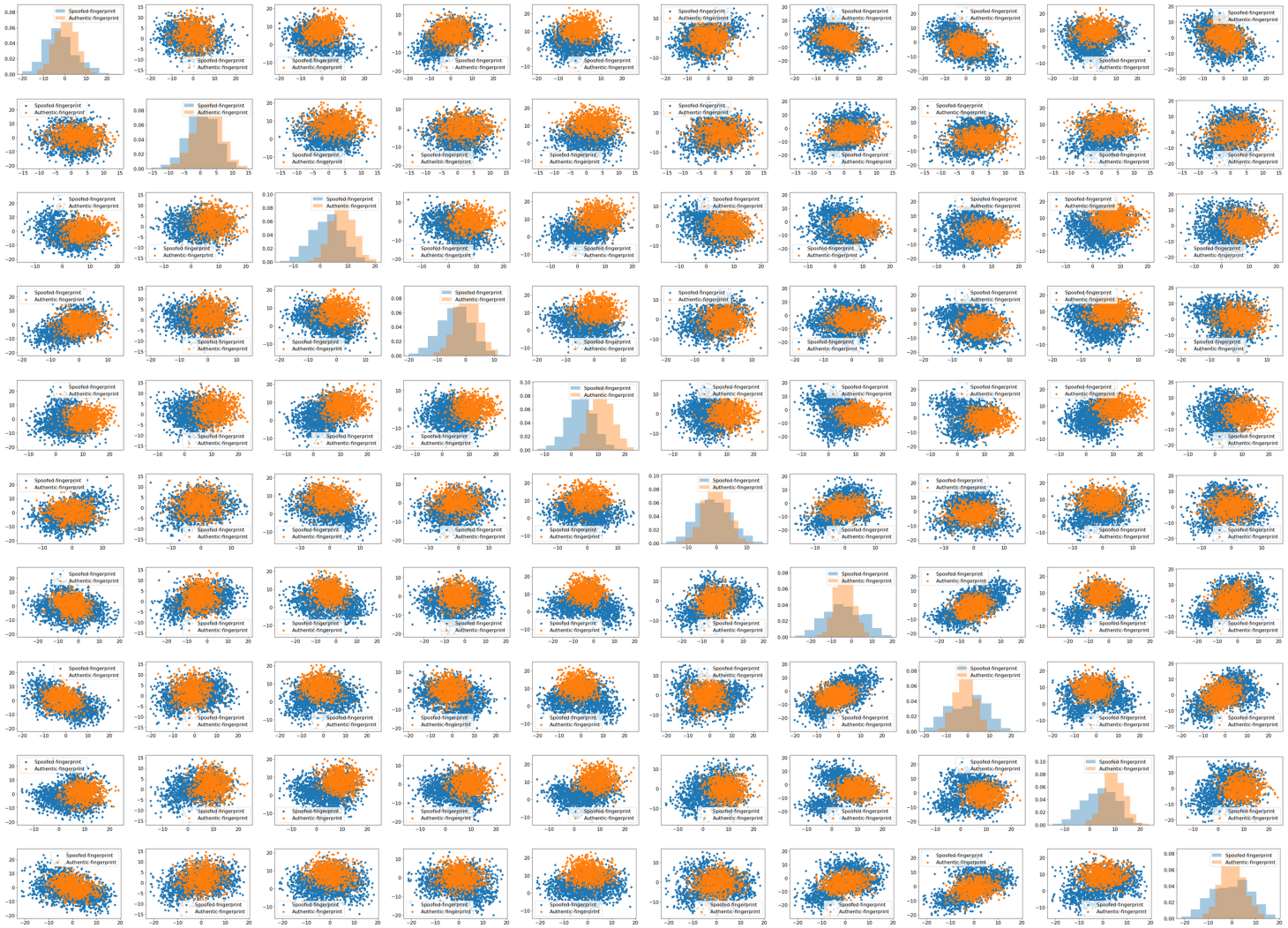


Fig. 1- Histogram and 2D scatter plots of the dataset features, where the target class are represented by blue and the non-target one are in orange

The graphical representations of the given data indicate that the features are not distributed optimally and exhibit a significant degree of overlap. This overlapping of features can potentially pose challenges in accurately analyzing the data and extracting meaningful insights. In order to address this issue, it is advisable to consider employing various pre-processing techniques, such as Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA), to enhance the quality and interpretability of the data.

By capturing the maximum amount of variance in the data, PCA helps in reducing the complexity of the dataset while preserving the essential information. Applying PCA to the given data can aid in identifying the most significant features that contribute the most to the overall variance, thereby facilitating a clearer separation between classes or clusters.

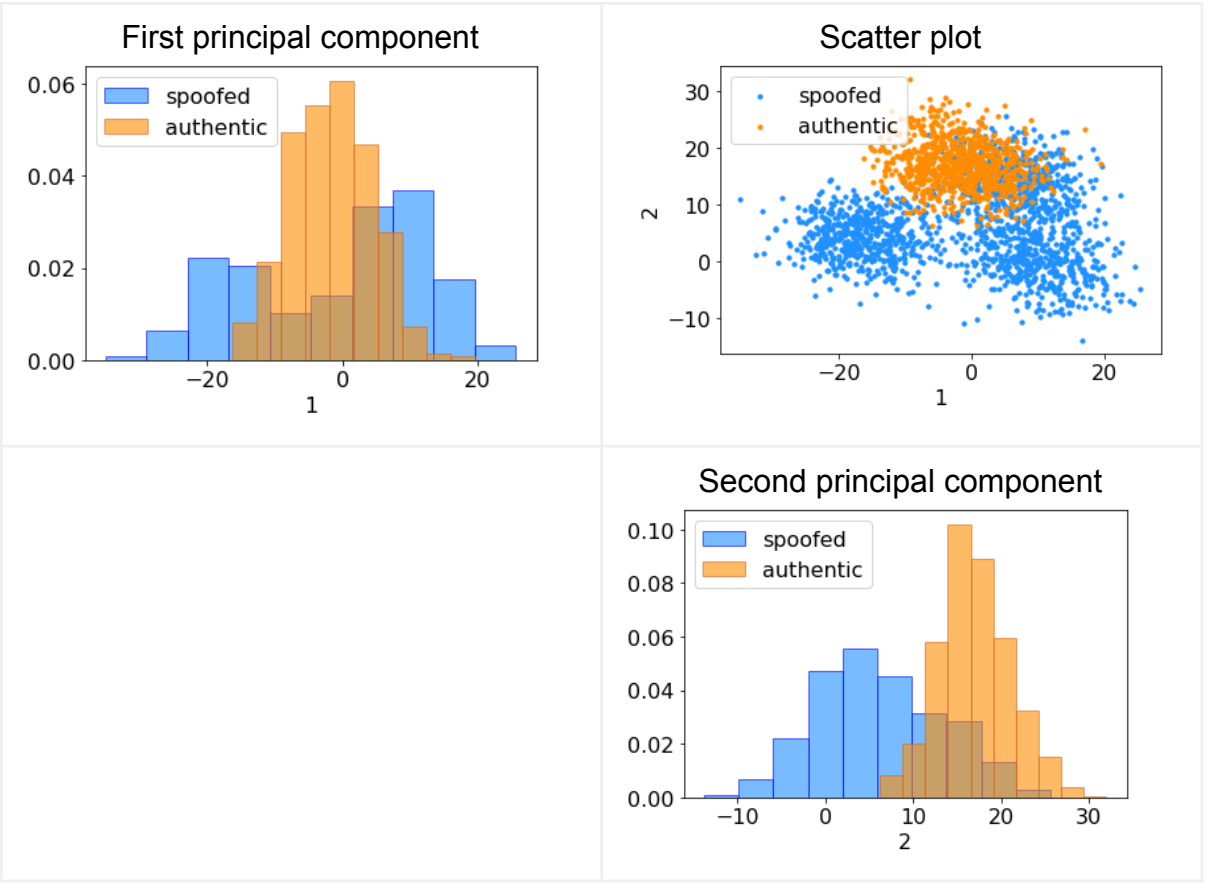


Fig. 2- Histogram and 2D scatter plots of the dataset features after applying PCA (there are represented only the first and second component)

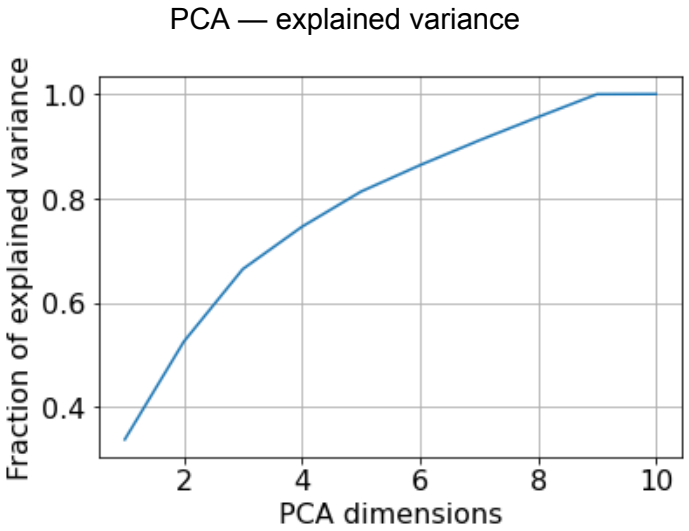


Fig. 3 - The graph describes the fraction of explained variance as a function of the PCA dimension for a given dataset.

In Fig. 3, the curve illustrates how increasing the PCA dimension contributes to cumulatively explaining a higher amount of variance in the dataset. We obtain 95% with 8 directions, and 90% with 7 directions. To start, we will consider these three values for PCA.

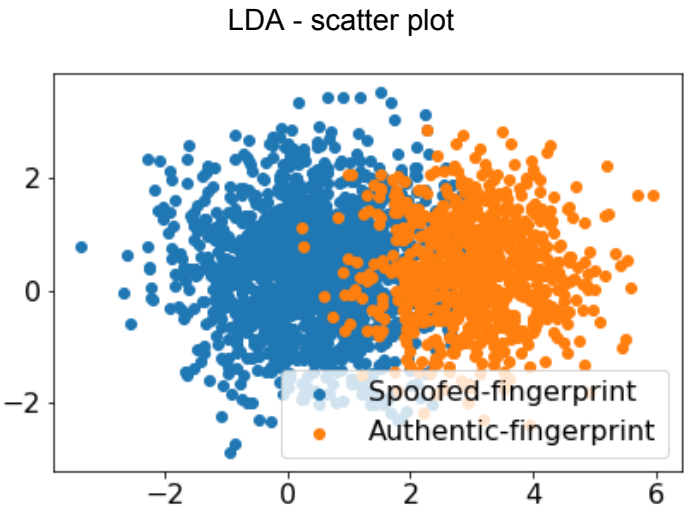


Fig. 4- Scatter plot 2D of the dataset features after applying LDA. We can compute at most 1 discriminant direction, since we have 2 classes.

By observing the results obtained with PCA, it is evident that the data points of the two classes are scattered partially along the same directions as the class mean, and we are still unable to properly separate the classes. However, by calculating the discriminant direction using LDA, we achieve a significantly improved separation of the input data for the two classes.

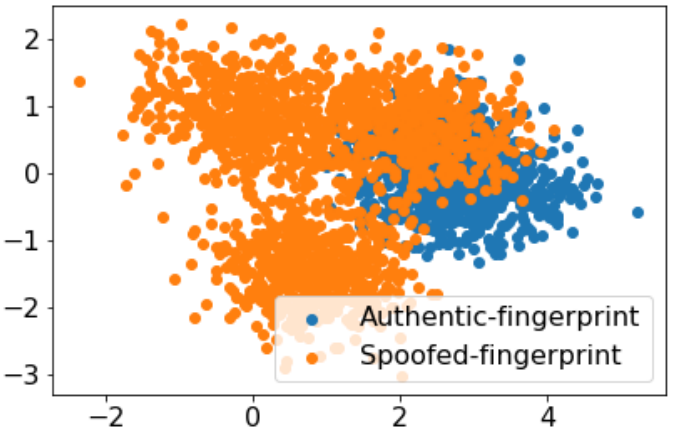


Fig. 5- 2D scatter plot of the dataset features after applying PCA, and secondly, LDA

The results are not promising. There is a lot of overlapping. Therefore, we will not consider PCA+LDA in our research.

Pearson correlation coefficient for the dataset features

Dataset	Target class	Non-target class

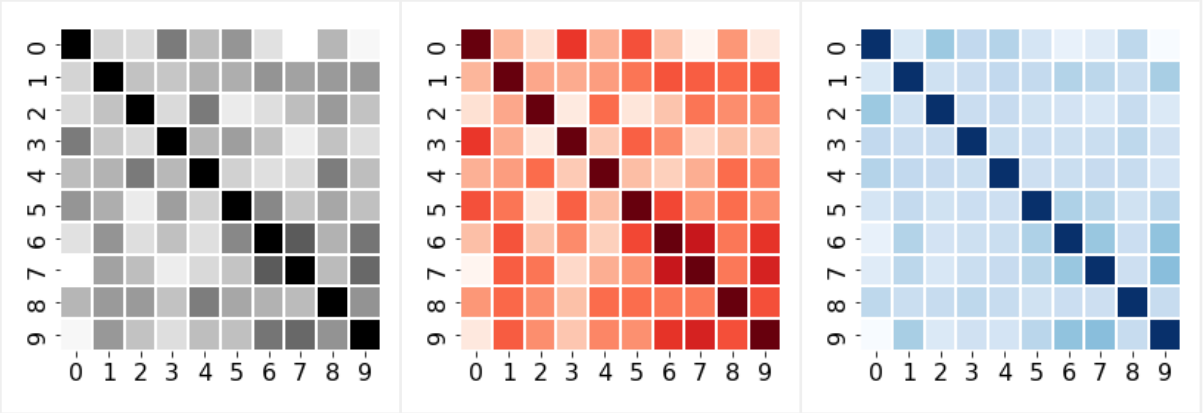


Fig. 6 - Heatmaps showing Pearson correlation coefficient between features.

Non-target class features are weakly correlated, whereas target class shows significantly larger correlations (feature 6, 7 and 9 are correlated, like feature 0 with 3…).

Prior to discovering the most suitable classifier, we have the option to apply z-normalization to our dataset. Z-normalization is a technique employed to normalize data by establishing a mean of zero and a standard deviation of one.

The decision to incorporate z-normalization is supported by its capacity to eliminate scale dependency, handle outliers, enhance coefficient interpretation, and optimize the convergence of optimization algorithms. Notably, the histogram and 2D scatter plots of the dataset features exhibit minimal alterations, as observed in the PCA (Principal Component Analysis) explained variance plot.

A notable distinction shows when utilizing PCA+LDA:

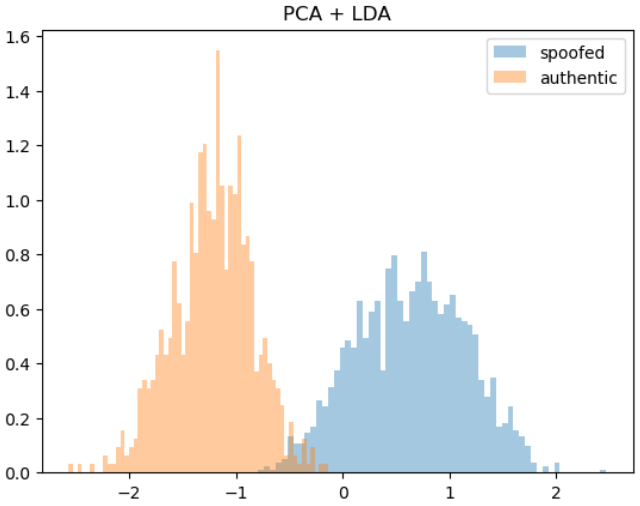


Fig. 7 - Histogram plot of the dataset features after applying PCA, and secondly, LDA

The data separation achieved with LDA+PCA can also simplify result interpretation. The principal components identified by PCA can be associated with specific dataset features or patterns, while the enhanced class separation from LDA can facilitate the identification of the most significant differences between them.

3. Classifying features

In order to perform our analysis, we will adopt two approaches. In the first one we will split the training set into training and validation subsets (this approach is called single-fold in the following), but we will also exploit the K-fold cross-validation approach. The singlefold consists of 80% training data and 20% validation data. The K-fold is implemented with K=5. In both cases data has been shuffled before splitting.

3.1 Gaussian classifiers

We start considering Gaussian classifiers (MVG classifier, MVG classifier with Naive Bayes assumption, MVG classifier with tied covariance). All of them assume gaussian distributed data, given the class:

$$X|C = c \sim N(\mu c, \Sigma c)$$

Multivariate models perform better if we have enough data to reliably estimate the covariance matrices.
Tied covariance models can capture correlations, but may perform poorly when classes have very different distributions. This classifier may not be the best to use.
On the other half, despite its "naive" assumption of feature independence, Naive Bayes can still perform reasonably well in the presence of correlated features. It has been observed to work surprisingly well in practice, especially when the correlations do not affect the class separability significantly.
To verify that and find the most promising approach, we can measure performances through the metric of 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.

Single-fold RAW					
	No PCA/LDA	PCA (m=8)	PCA (m=7)	LDA(m=2)	PCA+LDA
Full Cov	0.364	0.364	0.364	0.488	0.445
Naive Bayes	0.490	0.400	0.400	0.495	0.431
Tied Cov	0.495	0.473	0.473	0.495	0.481
Naive Bayes Tied Cov	0.490	0.400	0.400	0.495	0.431

K-fold (K=5) RAW					
	No PCA/LDA	PCA (m=8)	PCA (m=7)	LDA(m=2)	PCA+LDA
Full Cov	0.331	0.338	0.329	0.473	0.428
Naive Bayes	0.472	0.381	0.391	0.471	0.424
Tied Cov	0.486	0.472	0.476	0.480	0.473
Naive Bayes	0.472	0.381	0.391	0.471	0.424

Tied Cov					

Single-fold Z-normalized					
	No PCA/LDA	PCA (m=8)	PCA (m=7)	LDA(m=2)	PCA+LDA
Full Cov	0.364	0.395	0.421	0.502	0.515
Naive Bayes	0.490	0.425	0.436	0.502	0.507
Tied Cov	0.495	0.531	0.524	0.488	0.507
Naive Bayes Tied Cov	0.490	0.425	0.436	0.502	0.507

K-fold (K=5) Z-normalized					
	No PCA/LDA	PCA (m=8)	PCA (m=7)	LDA(m=2)	PCA+LDA
Full Cov	0.331	0.358	0.336	0.478	0.457
Naive Bayes	0.472	0.384	0.383	0.474	0.451
Tied Cov	0.486	0.486	0.476	0.483	0.460
Naive Bayes Tied Cov	0.486	0.384	0.383	0.474	0.451

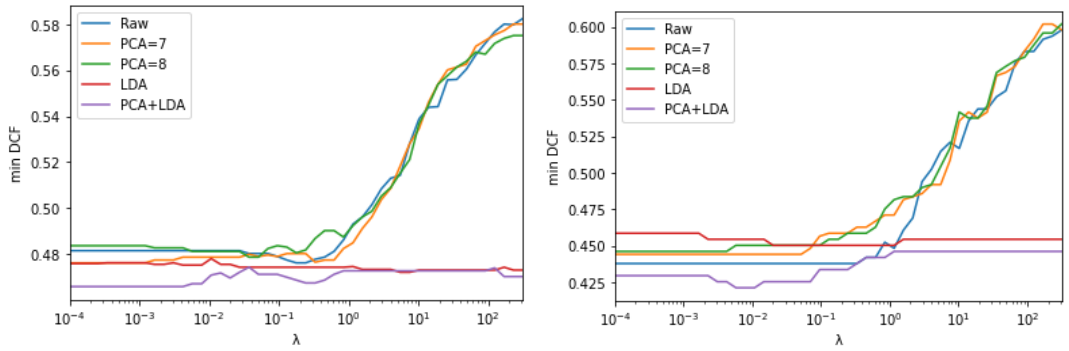
3.2 Linear Logistic Regression

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:

$$J(w, b) = \sum_{i=1}^{n} l(z_i(w^T x_i + b))$$

The model parameters are (w, b). The model assumes that decision rules are linear hyperplanes orthogonal to w.

Raw



Z-Norm

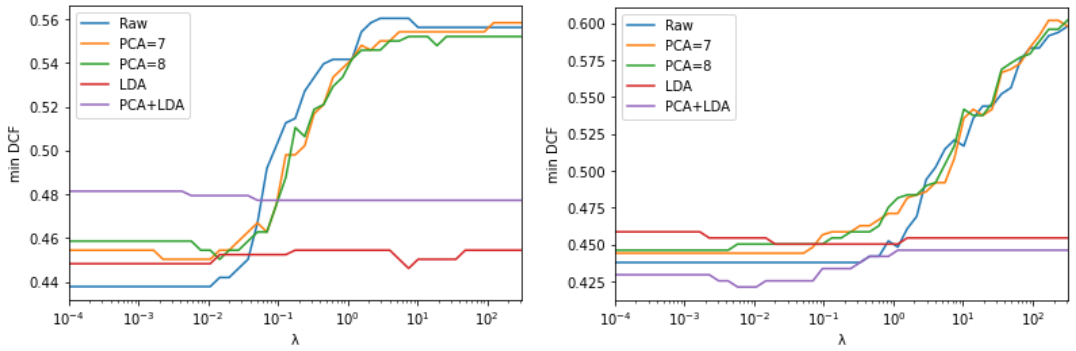


Fig. 8 - Plots using LR as classifier. On the left, a set of curves represents the variation of minDCF with changing C in single-fold mode. On the right, in the k-fold mode with k = 5.

When the value of λ (lambda) becomes excessively large, we notice a struggle within the model to accurately classify the samples. This phenomenon arises due to the amplification of "generalization," causing the model to oversimplify. As a result, it may fail to capture the intricate complexities and nuances present in the data, ultimately leading to subpar classification performance.
On the contrary, when λ assumes smaller values, the model tends to achieve a satisfactory separation on the training data and exhibits good generalization capabilities towards unseen data. In such instances, the regularization term provides little to no discernible benefits in enhancing the model's performance. Hence, it becomes more advantageous to report the outcomes derived from the non-regularized version of the model.

In the experiment, after observing the previously obtained graphs, it has been decided to select the value of lambda as 10^-2 for both.

Linear Regression RAW					
	No PCA/LDA	PCA (m=8)	PCA (m=7)	LDA(m=2)	PCA+LDA
Single fold	0.481	0.481	0.479	0.478	0.468
K fold (K=5)	0.438	0.450	0.444	0.454	0.421

Linear Regression Z-normalized					
	No PCA/LDA	PCA (m=8)	PCA (m=7)	LDA(m=2)	PCA+LDA
Single fold	0.486	0.474	0.480	0.474	0.455
K fold (K=5)	0.438	0.454	0.450	0.448	0.479

Until now, the multivariate Gaussian model outperforms linear regression because it can handle nonlinear relationships, outliers and noise more effectively. Additionally, its flexibility and ability to model complex interactions make it a more powerful approach for prediction in this task.

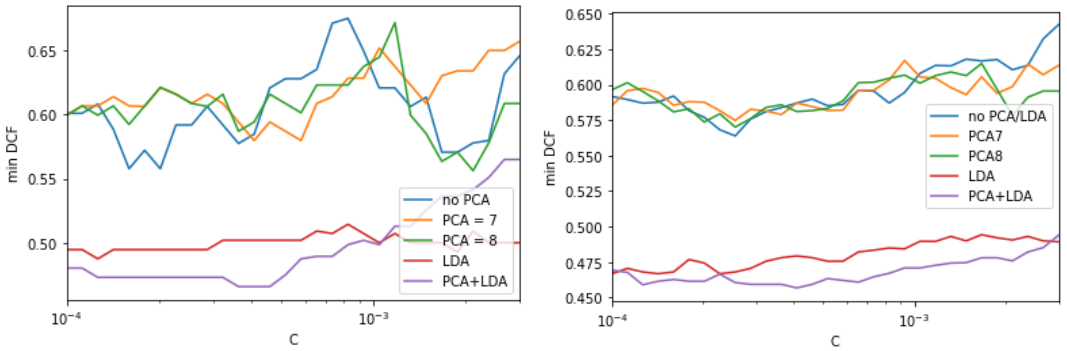
3.3 Support Vector Machine (SVM)

3.3.1 Linear SVM

We now turn our attention to SVMs. We will consider the linear SVM, the polynomial quadratic kernel and the Radial Basis Function kernel formulations.
We start considering a linear model that does not balance the two classes. To solve the SVM problem, we can consider the dual formulation:

$$J^D(\alpha) = -\frac{1}{2}\alpha^T H\alpha + \alpha^T 1$$

Raw



Z-norm

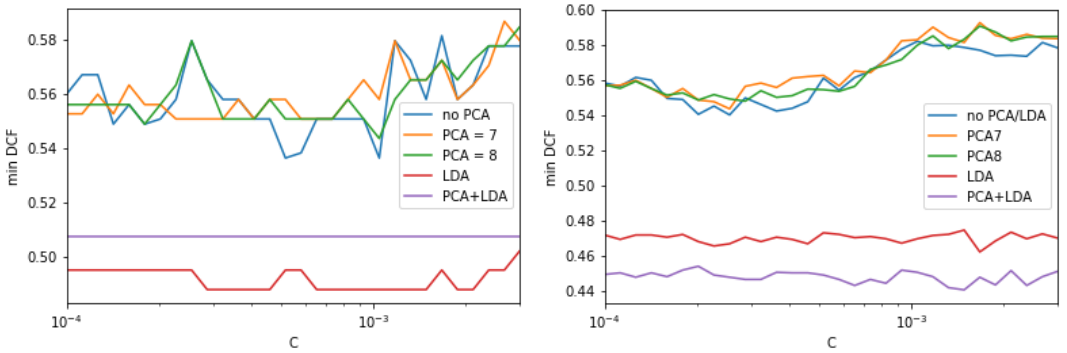


Fig. 9 - Plots using linear SVM as classifier. On the left, a set of curves represents the variation of minDCF with changing C in single-fold mode. On the right, in the k-fold mode with k = 5.

We did not achieve satisfactory results with the linear SVM classifier. The only moderately successful outcome was observed when using the LDA-modified data, but even there, the results are not optimal.

We have decided to use C = $2 * 10^{-5}$ because this hyperparameter yields the lowest minDCF value.

Linear SVM RAW					
	no PCA/LDA	PCA (m=8)	PCA (m=7)	LDA(m=2)	PCA+LDA
Single-fold	0.577	0.570	0.633	0.514	0.550
K-fold (K = 5)	0.623	0.592	0.596	0.491	0.478

Linear SVM Z-normalized					
	no PCA/LDA	PCA (m=8)	PCA (m=7)	LDA(m=2)	PCA+LDA
Single-fold	0.564	0.564	0.564	0.487	0.507
K-fold (K = 5)	0.574	0.582	0.583	0.468	0.443

3.3.2 Non-linear SVM

We now analyze the non-linear formulations.
In SVM, 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 through the dot product and it's possible to compute scores 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:

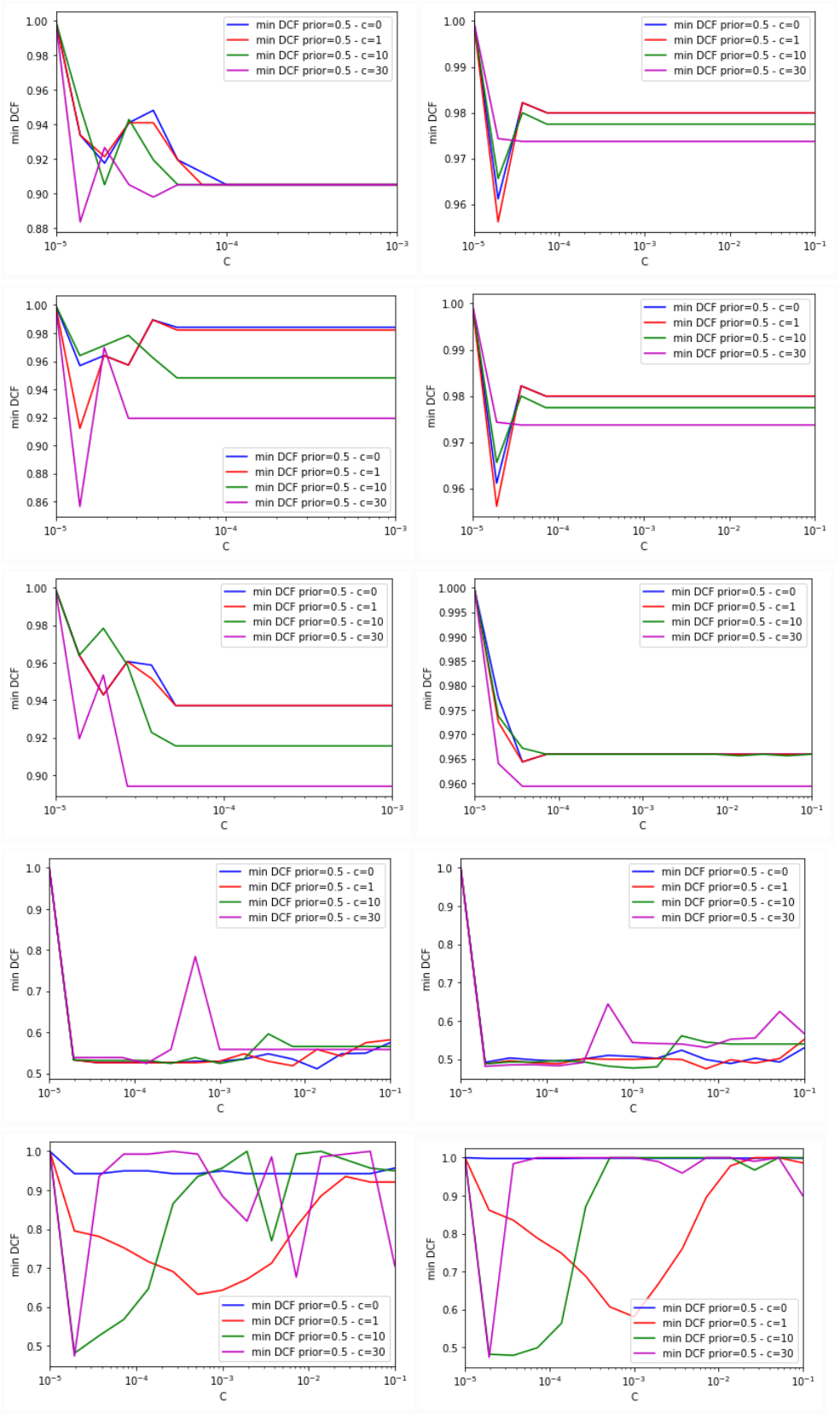
$$k(x_i, x_j) = \phi(x_i)^T \phi(x_j)$$

We will employ two different kernels:

1. Polynomial kernel of degree d = 2: $k(x_i, x_j) = (x_i^T x_j + c)^d$
2. Radial Basis Function (RBF) kernel: $k(x_i, x_j) = e^{-\gamma ||x_i - x_j||^2}$

3.3.3 Polynomial SVM

Raw



Z-norm

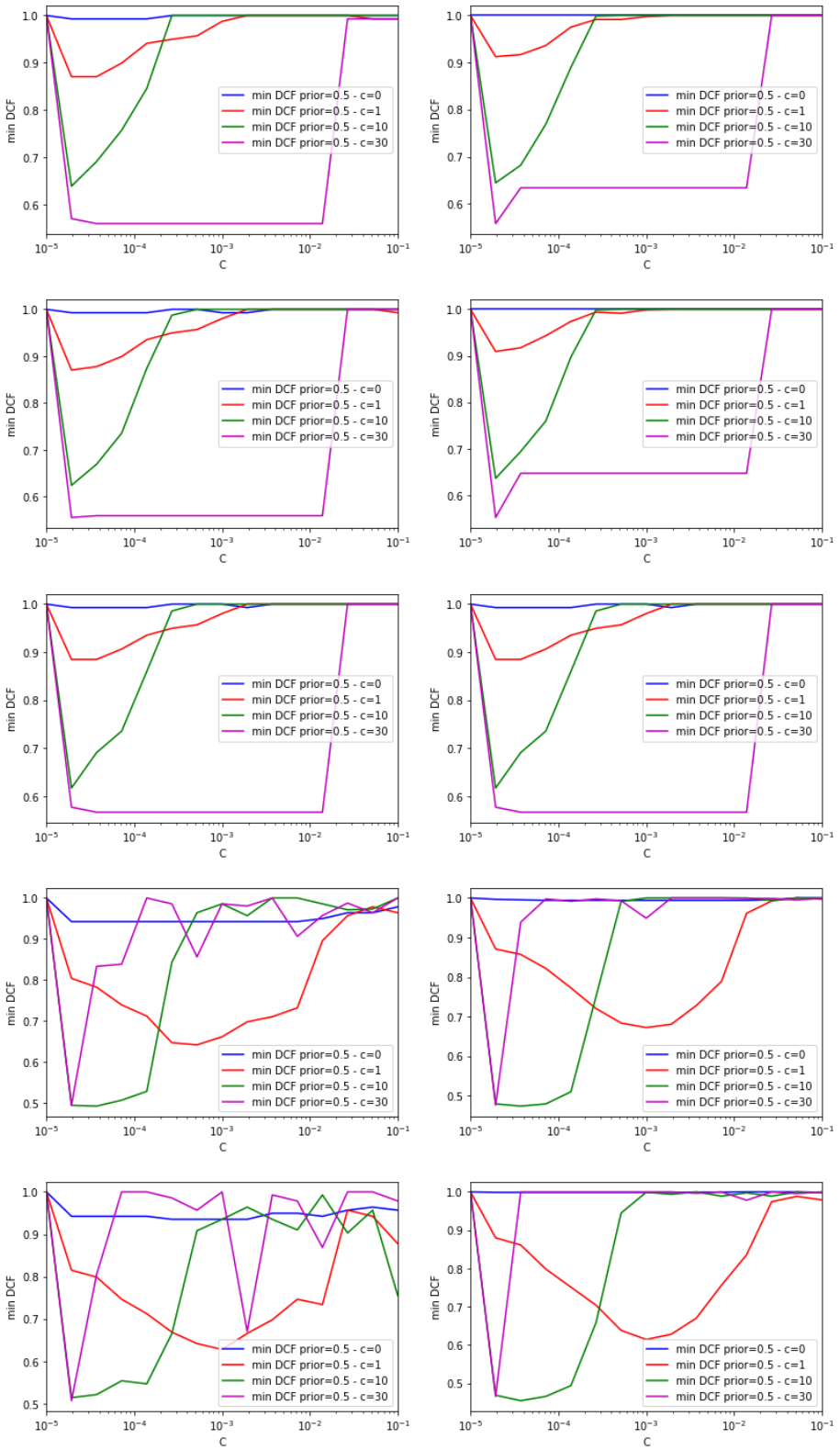


Fig. 10- Plots using polynomial SVM classifier. On the left, a set of curves represents the variation of minDCF with changing C in single-fold mode. On the right, in the k-fold mode with k = 5. (From top to bottom: no PCA/LDA, *PCA = 8, PCA = 7, LDA, PCA+LDA)*

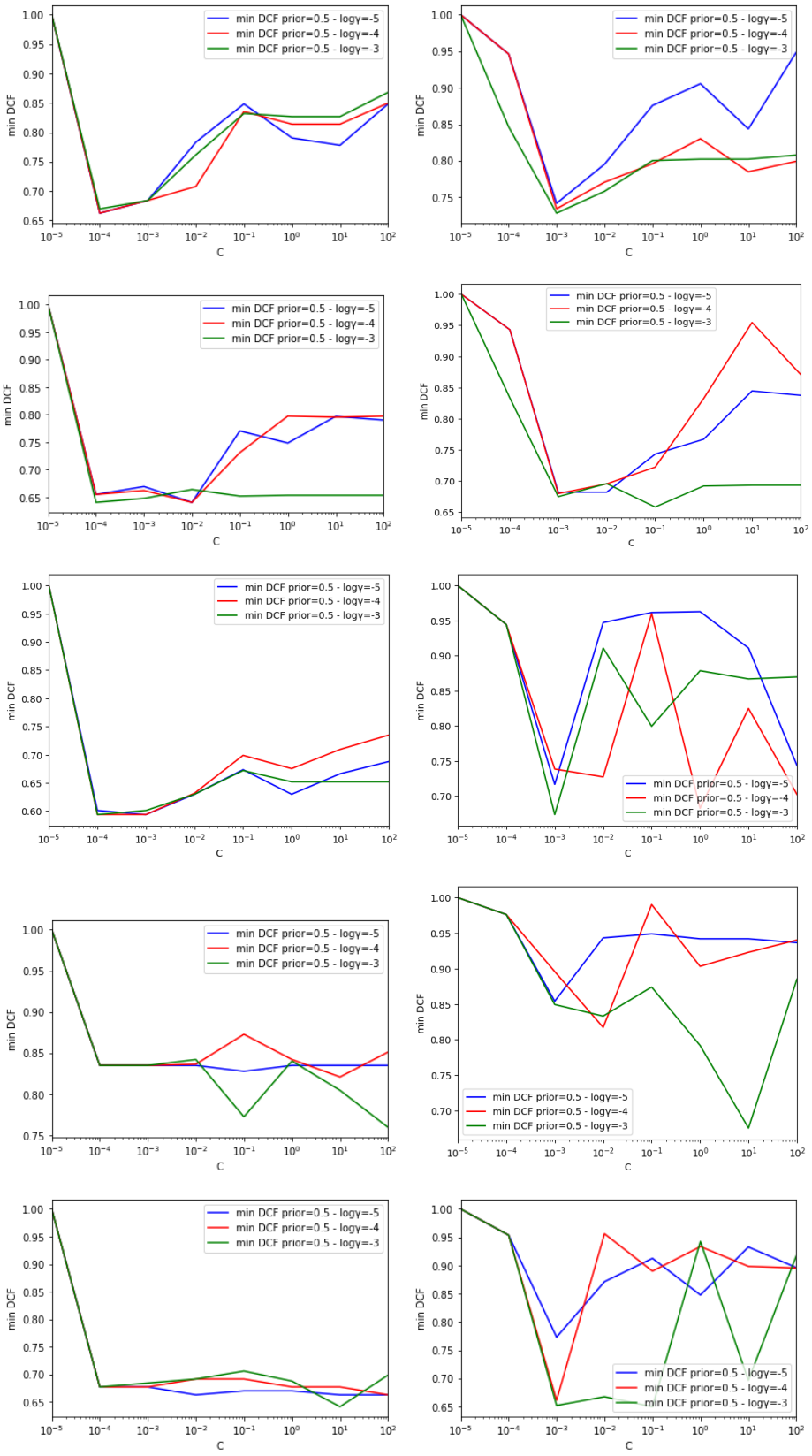
We have decided to use C = $2 * 10^{-5}$ and c = 30 because these hyperparameters yield the lowest minDCF value.

Polynomial SVM RAW					
	no PCA/LDA	PCA (m=8)	PCA (m=7)	LDA(m=2)	PCA+LDA
Single-fold	1.000	1.000	1.000	0.494	1.000
K-fold (K = 5)	1.000	1.000	1.000	0.474	0.473

Polynomial SVM Z-normalized					
	no PCA/LDA	PCA (m=8)	PCA (m=7)	LDA(m=2)	PCA+LDA
Single-fold	0.570	0.555	0.577	0.494	0.557
K-fold (K = 5)	0.557	0.552	0.552	0.475	0.465

3.3.4 RBF SVM

Raw



Z-norm

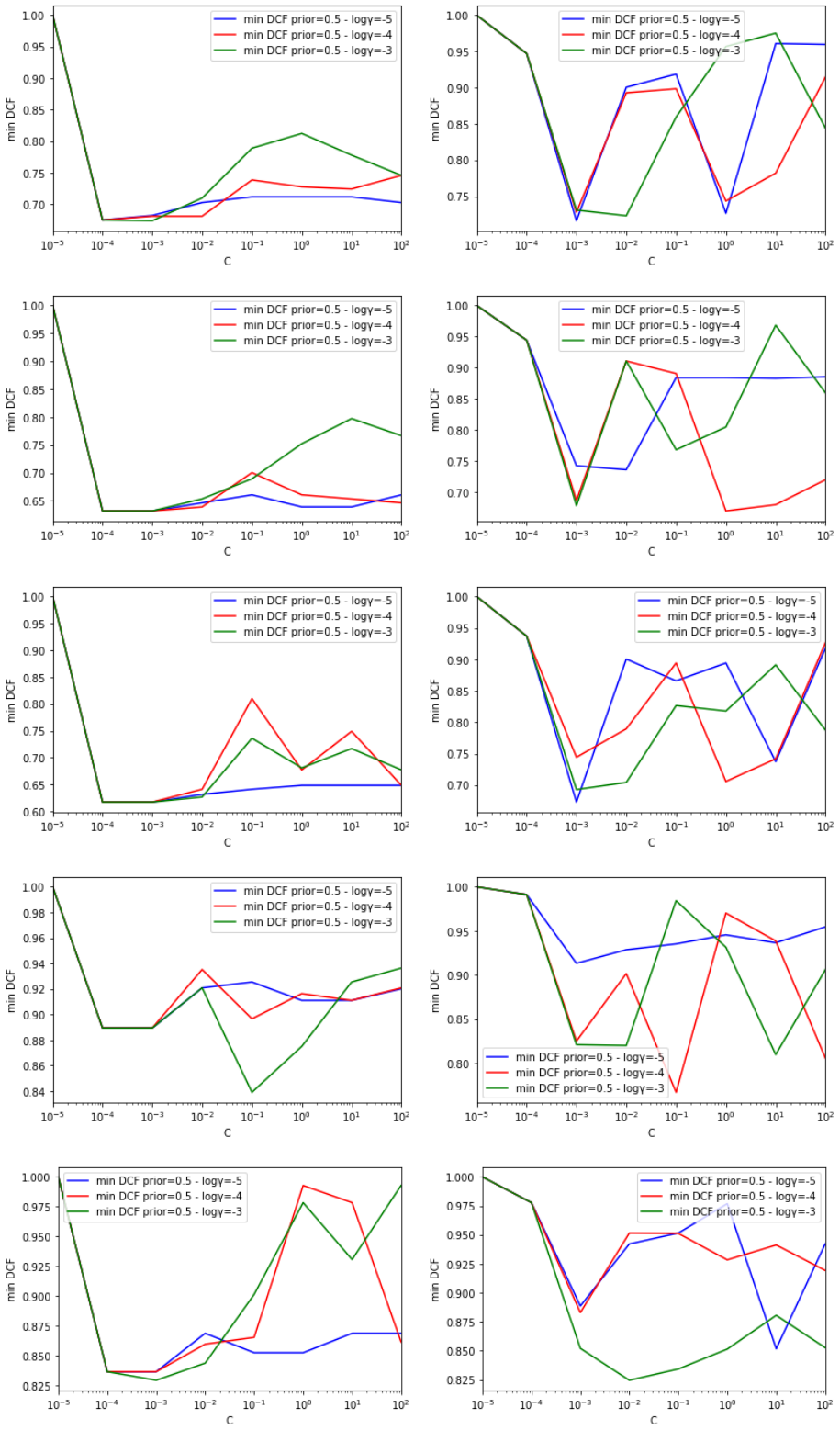


Fig. 11- Plots using RBF *SVM classifier. On the left, a set of curves represents the variation of minDCF with changing C in single-fold mode. On the right, in the k-fold mode with k = 5. (From top to bottom:* no PCA/LDA, *PCA = 8, PCA = 7, LDA, PCA+LDA)*

We have decided to use C = 10^{-4} and $\gamma = 10^{-5}$ because these hyperparameters yield the lowest minDCF value.

RBF SVM RAW					
	no PCA/LDA	PCA (m=8)	PCA (m=7)	LDA(m=2)	PCA+LDA
Single-fold	0.662	0.655	0.601	0.834	0.677
K-fold (K = 5)	0.946	0.943	0.938	0.976	0.953

RBF SVM Z-normalized					
	no PCA/LDA	PCA (m=8)	PCA (m=7)	LDA(m=2)	PCA+LDA
Single-fold	0.674	0.617	0.631	0.889	0.836
K-fold (K = 5)	0.947	0.937	0.944	0.911	0.977

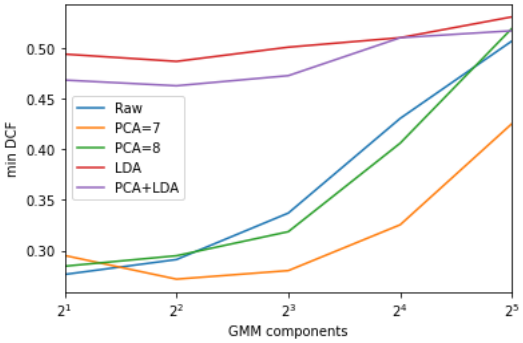
Although Support Vector Machine (SVM) is a powerful and versatile classifier, it may not be suited for this particular scenario. SVM's limitations arise in cases where data is highly complex, non-linear, or has a large number of features. Additionally, SVM can be computationally expensive for large datasets

3.4 GMM

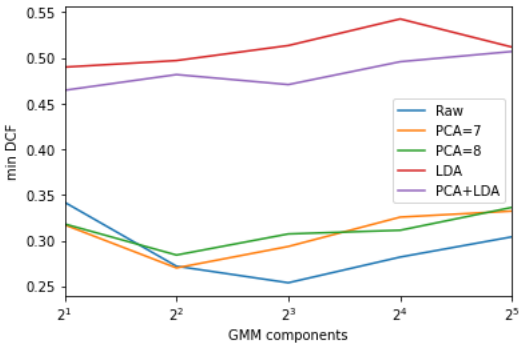
We will now examine the final model, which is a generative model constructed by training a Gaussian Mixture Model (GMM) using the data from both classes. Our analysis will encompass three variations: the full covariance model, the diagonal model, the model with covariance tying and the Naive Bayes tied covariance. In the case of the tied covariance model, covariance tying occurs at the level of each class, resulting in distinct covariance matrices for different classes.

Considering that Naive Bayes Tied Cov is closely related to Naive Bayes, we will opt for the latter due to its simplicity.

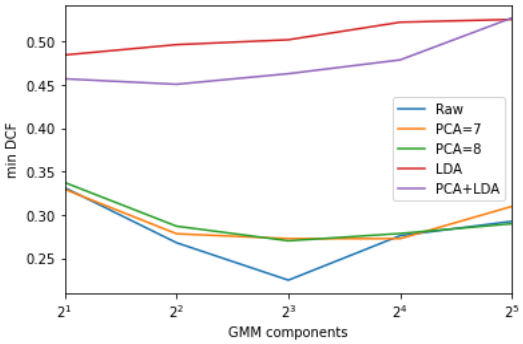
K-fold Full Cov Z-Normalized



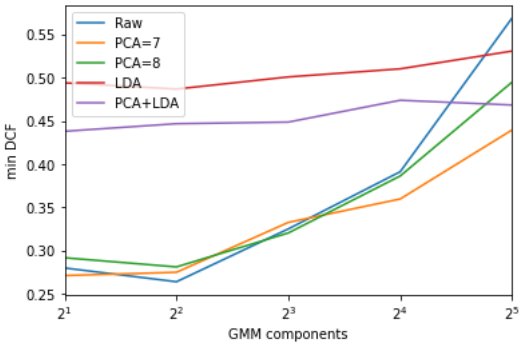
K-fold Naive Bayes Z-Normalized



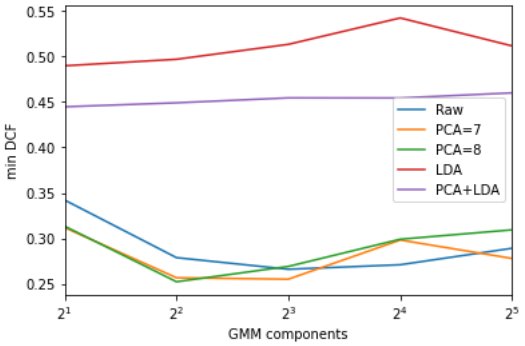
K-fold Tied Cov Z-Normalized



K-Fold Full Cov Raw



K-fold Naive Bayes Raw



K-fold Tied Cov Raw

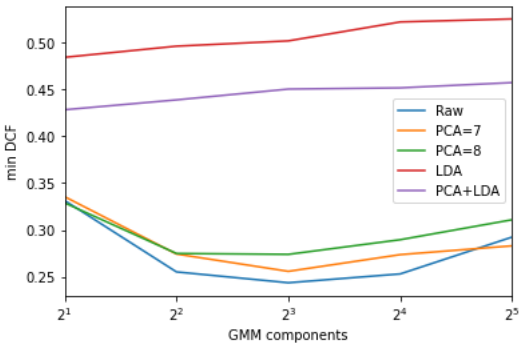


Fig.12- Plots using GMM classifiers.

We report in the following table, only the best models and we do not consider the gaussians with one component because they are already described above.

When evaluating various models, it was observed that the performance deteriorates as the number of components exceeds 8. Additionally, single-fold analysis yielded worse results, prompting us to solely consider K-fold evaluations for a more reliable assessment. It is essential to conduct K-fold cross-validation to obtain a more accurate and representative evaluation of the models' performance, especially when dealing with a limited dataset.

Full Cov K-Fold (K=5) Raw					
	No PCA/LDA	PCA (m=8)	PCA (m=7)	LDA(m=2)	PCA+LDA
2 components	0.280	0.291	0.271	0.494	0.438
4 components	0.266	0.280	0.275	0.487	0.447
8 components	0.324	0.281	0.332	0.500	0.448

Naive Bayes K-Fold (K=5) Raw					
	No PCA/LDA	PCA (m=8)	PCA (m=7)	LDA(m=2)	PCA+LDA
2 components	0.342	0.313	0.312	0.490	0.445
4 components	0.279	0.252	0.257	0.497	0.449
8 components	0.266	0.269	0.255	0.512	0.455

Tied Cov K-Fold (K=5) Raw					
	No PCA/LDA	PCA (m=8)	PCA (m=7)	LDA(m=2)	PCA+LDA
2 components	0.331	0.335	0.329	0.484	0.428
4 components	0.262	0.271	0.270	0.496	0.439
8 components	0.245	0.273	0.253	0.503	0.436

Full Cov K-Fold (K=5) Z-Normalized					
	No PCA/LDA	PCA (m=8)	PCA (m=7)	LDA(m=2)	PCA+LDA
2 components	0.276	0.284	0.295	0.494	0.468
4 components	0.291	0.295	0.272	0.487	0.463
8 components	0.344	0.319	0.280	0.501	0.473

Naive Bayes K-Fold (K=5) Z-Normalized					
	No PCA/LDA	PCA (m=8)	PCA (m=7)	LDA(m=2)	PCA+LDA
2 components	0.342	0.318	0.317	0.490	0.464
4 components	0.272	0.284	0.270	0.497	0.482
8 components	0.254	0.307	0.294	0.512	0.471

Tied Cov K-Fold (K=5) Z-Normalized					
	No PCA/LDA	PCA (m=8)	PCA (m=7)	LDA(m=2)	PCA+LDA
2 components	0.331	0.337	0.329	0.484	0.457
4 components	0.266	0.287	0.278	0.496	0.450
8 components	0.225	0.270	0.275	0.503	0.462

We once again employ the 5-folds approach to determine the optimal number of components and compare various models.
For our target application, the Tied-Cov K fold model with 8 components and without PCA and LDA, yields the most promising results.

The results obtained with LDA, whether used alone or in conjunction with PCA, are rather disappointing. Despite initially exhibiting a clear separation of data belonging to the two different classes during the feature analysis, the performance decreases significantly when utilized with more complex models such as GMM. This can be attributed to the data being overly simplified, resulting in the classifier lacking sufficient information to perform accurate classification.

In the final analysis phase, where the Test set will be employed, it is expected that models utilizing GMM with a high number of components will experience a decline in performance. This is due to the Test set being larger than the Train set, increasing the risk of overfitting with the use of a more complex model like GMM and employing the K-Fold technique.

4. Score Calibration

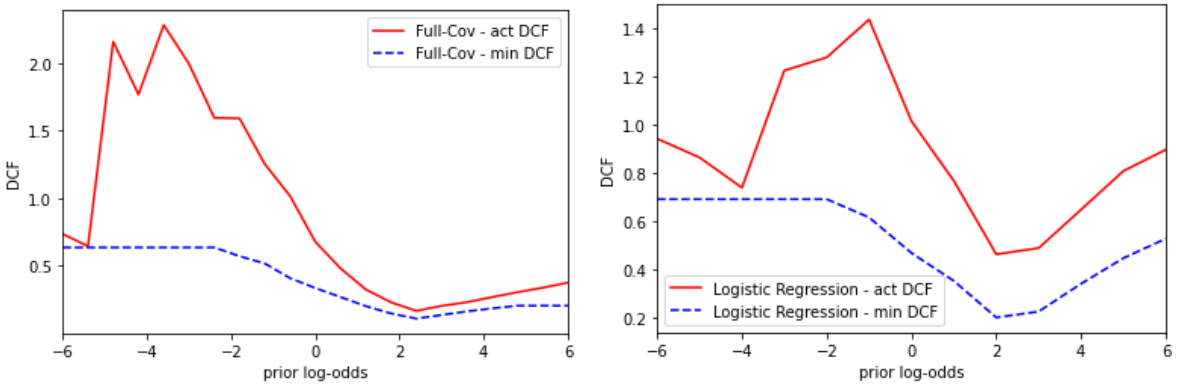
Up until now, we have only focused on the minimum DCF metric. As we've discussed before, the minimum DCF measures the expense we would face if we made the best decisions for the validation set using the scores of the recognizer.
However, the actual cost we incur depends on how well we choose the threshold for class assignment.
Therefore, let's now consider the actual DCFs.
If the scores are properly calibrated, the optimal threshold that minimizes the Bayes risk is the theoretical threshold, which is calculated as t=-log1- .

Now, let's evaluate the actual DCF to determine how well the models would perform if we were using the theoretical threshold for each application, assuming that the scores are already well-calibrated.
(We excluded the SVM classifier from the analysis due to its poor performance.)

	minDCF	actDCF
MVG, Full Cov, PCA m=7 - Raw	0.329	0.678
Log Reg, ($\lambda=10^{-2}$), no PCA/LDA - Raw	0.421	1.014
GMM TiedCov, 8 components, no PCA/LDA - Z-Norm	0.225	0.487
GMM Naive Bayes, 4 components, PCA m=8 - Raw	0.252	0.562

We can observe that all the prediction models used that yielded the best results produce poorly calibrated outcomes, especially for Logistic Regression.

This analysis can also be verified through Bayes error plots, which show the DCFs for different applications.



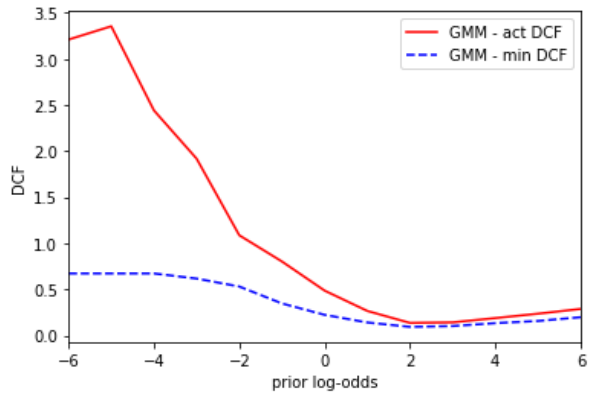


Fig. 13- Bayes error plots, which show the actual DCF e min DCF for different applications (from top to bottom: MVG Full Cov, PCA m=7 - Raw, Log Reg, ($\lambda=10^{-2}$), no PCA/LDA - Raw, GMM TiedCov, 8 components, no PCA/LDA - Z-Norm)

First, we aim to improve the calibration of all three models by employing score calibration techniques. The general approach involves computing a transformation function, denoted as f(s), which maps the scores.We assume that function f is linear in s:

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

This function 'f(s)' can be interpreted as the log-likelihood ratio for the two class hypotheses, denoted as 'HT' and 'HF':

$$f(s) = log\frac{f_{S|C}(s \mid HT)}{f_{S|C}(s \mid HF)} = \alpha s + \beta$$

so the class posterior probability for a prior π' can be written as:

$$f(s) = log\frac{P(C=HT \mid s)}{P(C=HF \mid s)} = \alpha s + \beta + log(\frac{\pi'}{1-\pi'})$$

We can interpret the scores as features so that this expression will be similar to the log posterior ratio of the Logistic Regression model. If we rewrite:

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

By specifying a certain prior (0.5 in our case), we effectively optimize the calibration for this specific application. Nevertheless, this approach provides good calibration for different applications as well.

Considering that there are no substantial disparities between the two calibrations conducted with λ = 10^−3 and λ = 10^−2, we can confidently opt for λ = 10^-2 once again. Moving forward, we will reassess the actual DCFs after calibration. As the final scores need to be properly calibrated, the actual DCFs will be calculated using the theoretical threshold.

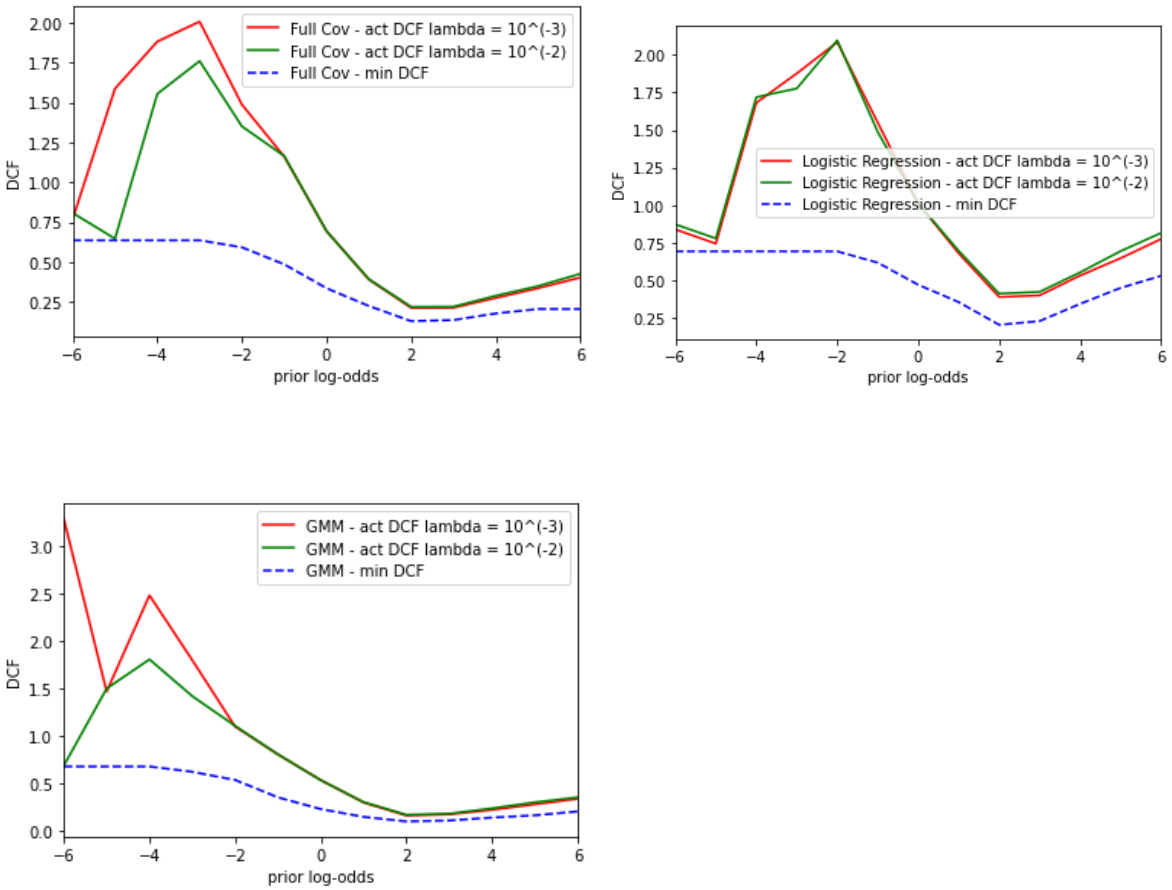


Fig. 14- Bayes error plots after calibration with 5- folds approach and prior-weighted logistic regression.

Miin DCFs and actual DCFs after score calibration, 5-folds approach

	minDCF	actDCF
MVG, Full Cov, PCA m=7 - Raw	0.329	0.693
Log Reg, ($\lambda=10^{-2}$), no PCA/LDA - Raw	0.421	1.010
GMM TiedCov, 8 components, no PCA/LDA - Z-Norm	0.225	0.466
GMM Naive Bayes, 4 components, PCA m=8 - Raw	0.252	0.568

Looking at the calibration results, it is evident that there are no significant improvements observed.

The lack of improvement in data calibration despite the significant difference between the actual DCF (Detection Cost Function) and the minimum DCF can be attributed to various

factors. Data calibration in machine learning aims to align the predicted probabilities or scores with the actual outcomes to achieve better calibration and calibration-related metrics. However, even if the actual DCF and minimum DCF differ substantially, the calibration process may not yield significant improvements for several reasons:

1) Calibration Issues: The models used may already be well-calibrated, meaning the predicted probabilities or scores are already aligned with the actual outcomes. In such cases, additional calibration may not lead to significant improvements. This isn't our case.

2) Limited Sample Size: If the dataset used for calibration is relatively small or unrepresentative of the overall population, the calibration process may not effectively capture the true calibration patterns, leading to negligible improvements.

3) Model Limitations: The models themselves may have inherent limitations or biases that prevent them from achieving better calibration, even when there are noticeable differences between the actual DCF and minimum DCF.

4) Feature Relevance: The features used for calibration might not be strongly correlated with the calibration metrics, causing the calibration process to have minimal impact on improving performance.

5. Experimental results

Let's analyze the different models' performances on the test set in terms of minimum Detection Cost Function (DCF) values.
After conducting tests on our models using various dimensional reduction strategies, we noticed a significant decline in performance when applying PCA. As a result, we chose to present only the best results without any preprocessing involved.

	No PCA Z-Norm	No PCA Raw
MVG Full Cov	0.258	0.260
MVG Naive Bayes	0.323	0.331
MVG Tied Cov	0.469	0.455
Logistic Regression	0.462	0.455
Linear SVM	0.502	0.495
Polynomial SVM	0.420	1.000
RBF SVM	0.626	0.606
GMM Full Cov 2 components	0.248	0.249
GMM Full Cov 4 components	0.281	0.273
GMM Full Cov 8 components	0.332	0.291
GMM Naive Bayes 2 components	0.291	0.260
GMM Naive Bayes 4 components	0.271	0.262
GMM Naive Bayes 8 components	0.264	0.256
GMM Tied Cov 2 components	0.258	0.260
GMM Tied Cov 4 components	0.267	0.262
GMM Tied Cov 8 components	0.263	0.256

The results are consistent with the ones obtained in the previous analysis and the best models for our target application remain MVG Full Cov and the GMM models, in particular MVG with few components. This is because when working with large datasets and employing complex models that require intensive computations, there is an increased risk of overfitting, especially when utilizing the K-Fold technique.

The close similarity observed between the validation and evaluation results indicates that there are no significant distinctions between the evaluation population and the training population.

We can now move forward with score calibration and implement the same process described before. This will enable our system to utilize the theoretical threshold for optimal decision-making. To compare our classifiers, we can utilize a ROC plot. The classifiers with the highest AUC (Area Under Curve) are considered the best performers. In our case, the GMM Tied-Cov 8 components stands out as the top classifier.

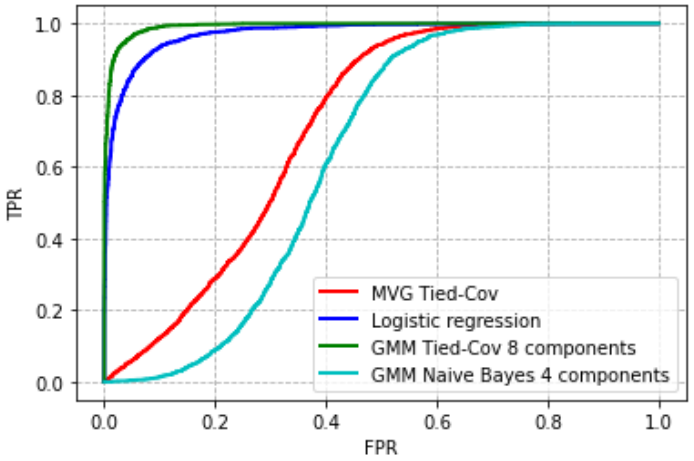


Fig. 15 - ROC plot for models comparison

References

[1] C. Kanmani Pappa, T. Kavitha, I. Rama Krishna, V. Venkata Lokesh & A. V. L. Narayana in Generalization of Fingerprint Spoof Detector.