Fingerprint Spoofed Detection

Machine Learning and Pattern Recognition

Belloni Sofia 303393 - Ferla Damiano 30593

Politecnico di Torino, 17/07/2023

Contents

1 Introduction

1.1 Abstract

The goal of the project is the development of a classifier to detect whether a fingerprint image represents an authentic or spoofed fingerprint. The fingerprint images are represented by means of embeddings, i.e. low-dimensional representations of images obtained by mapping images to allow dimensional manifold.

The dataset consist of synthetic data, and embeddings had signficantly lower dimensions than real use cases.

We are going to analyze different classifiers in order to understand which model fit better for the dataset, and we are going to compute the cost for each model.

1.2 Problem Overview

The dataset contains synthetic features that represent fingerprint samples. The embeddings are 10-dimensions, continues valued vectors, belonging to either the authentic fingerprint (label=1) or the spoofed fingerprint (label=0) class.

The dataset consist of 10.029 samples, which are divided in 2.325 samples for the training set and 7.704 for the test set. Moreover, the training class is imbalanced. In fact, we have 800 samples which belong to authentic fingerprints and 1525 samples which belong to spoofed fingerprints.

The target application considers an application where prior probabilities of authentic and spoofed classes are the same, but labeling a spoofed fingerprint as authentic has a larger cost due to the security vulnerabilities that such errors would create. The target application working point is therefore defined by the triplet ($\pi_T = 0.5, C_{fn} = 1, C_{fp} = 10$).

1.3 Feature Analysis

In this subsection we are going to dive deeper into training set features, in order to see the distribution over histograms.

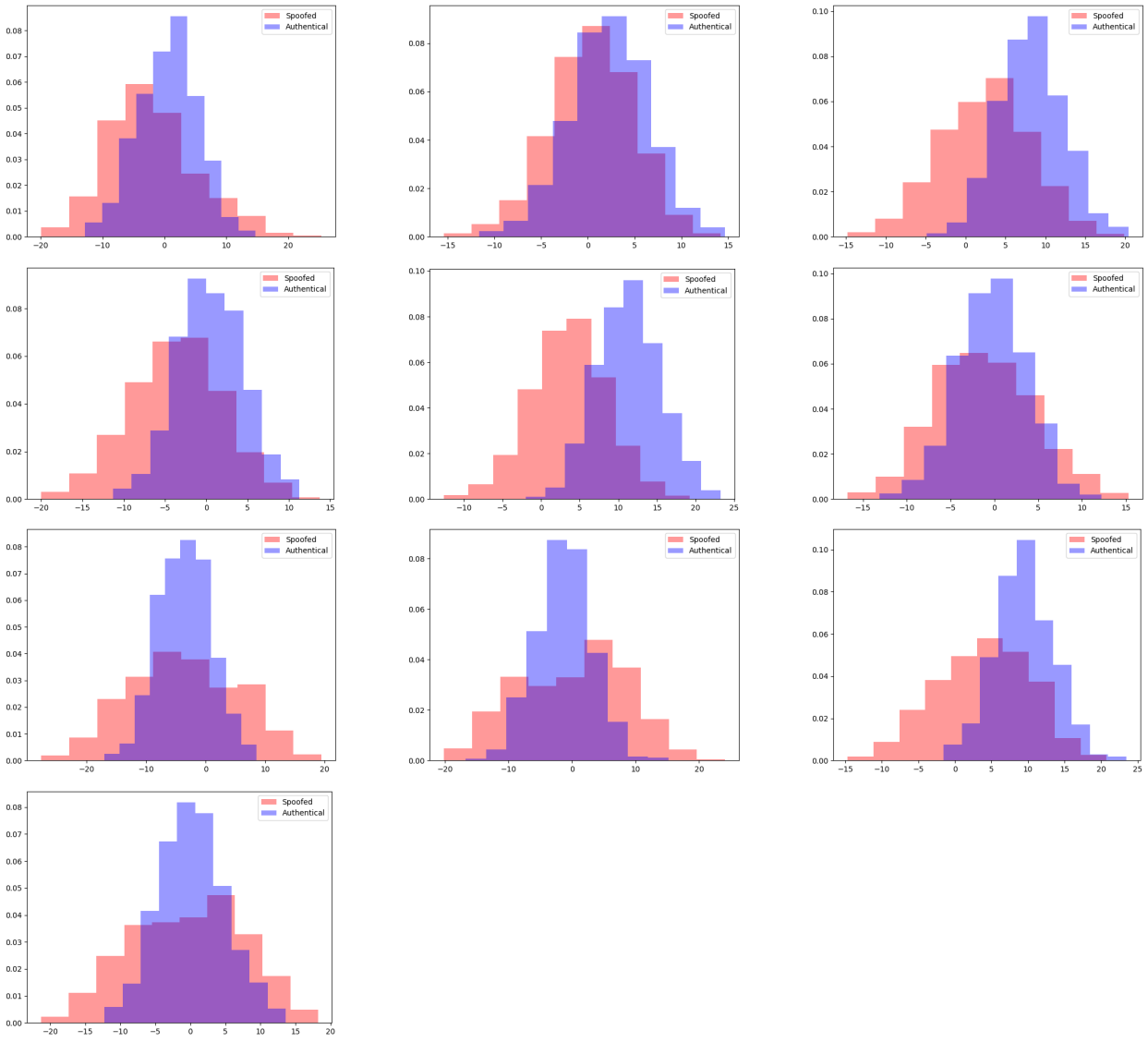

Figure 1: Histograms per feature

As we can see, none of the features seems to be very discriminant for the two classes. An

exception could be the histogram n.5, that is the most discriminant feature for the dataset. Authentical fingerprint class features are referable to Gaussian distributions, while the spoofed fingerprint class features have a different behaviour according to which feature we are going to take in exam.

This statement can be confirmed in the scatter plots of pairs of features, as it can be seen that the samples of the target class maintain the same distribution, whereas the other class varies greatly depending on the analyzed features. Below are some more significant scatter plots.

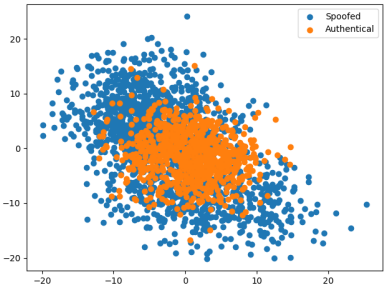

Figure 2: Features 0-7

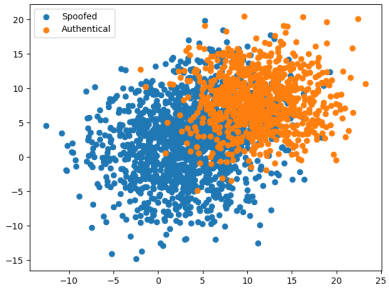

Figure 3: Features 4-2

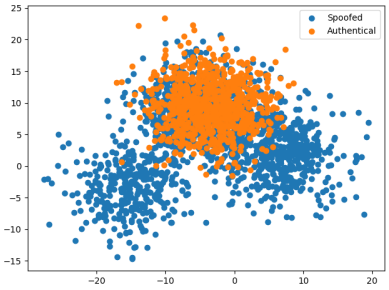

Figure 4: Features 6-8

Now, let's consider the correlation for our dataset. Considering the heatmap of Pearson correlation between features for all the dataset, we can state that features are not very correlated, exception features 7-8 (on the heatmap the point are 6-7 because we start from 0), features 8-10 (7-9). For this reason, we are going to expect that PCA (Principal Component Analysis) will not help so much.

Consider the heatmaps for a single class. These two heatmaps are different between them, in fact in the heatmap representing the correletion for the authentical fingerprints the features are basically uncorrelated, while in the heatmap representing the correlation for the spoofed

fingerprints the features are a bit more correlated.

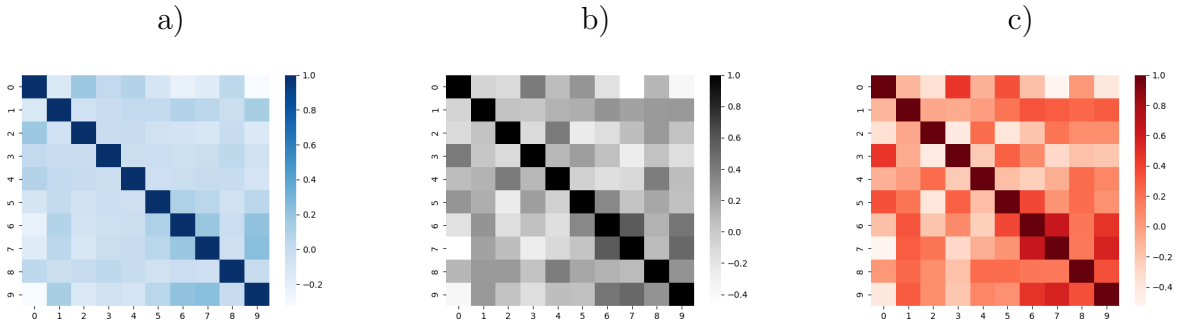

Figure 5: Correlation for a) authentical class; b) whole dataset; c) spoofed class

Representing the explained variance from PCA we can see that for each dimension we reduce, we have a significatly loss of information. However, if we want to reduce the number of dimension anyway, in order to have a slightly lighter model, we can do but we have to pay attention to not go under 90% of explained variance. In shorts, we can reduce 2-3 dimensions safetely. In the next sections, we are going to check later this assumption.

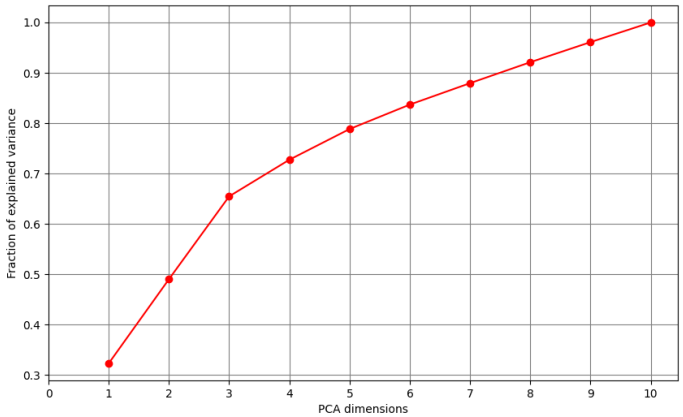

Figure 6: Explained variance

One way to understand if features are linearly discriminable is the LDA (Linear Discriminant Analysis), often used as a preprocessing step. Considering the fact that we have only two classes, we can represent at maximum 1 dimension.

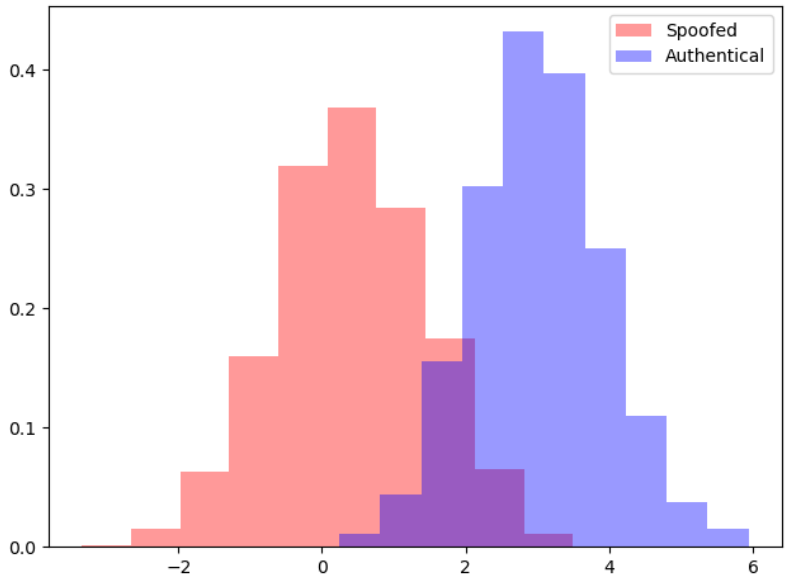

Figure 7: LDA histogram

As we can see from this plot , classes are discriminable but there is a small region in where they are overlapping.

1.4 Training Protocol

We adopt a K-Fold Cross Validation with K=10, and the training set is shuffled with a specific seed in order to reproduce the same results (seed=10).

Our primary metric in order to valuate the performance is **minDCF**, because this metric keeps in account the different costs for a false positive or false negative. As we said before, a misclassification of a spoofed fingerprint labeled as authentic could bring security vulnerabilities. Then, we are going to choose the best model and test it on the test set.

In addiction, we will test our model even in other application, defined by the following triplets:

- $(\pi = 0.5, C_{fn} = 1, C_{fp} = 10)$
- $(\pi = 0.1, C_{fn} = 1, C_{fp} = 10)$
- $(\pi = 0.9, C_{fn} = 1, C_{fp} = 10)$

2 Multivariate Gaussian Classifier

2.1 Overview

We used different models in order to collect different results and choose the model which performs better. In this section, we analize the following generative models:

- Standard Multivariate Gaussian Classifier
- Naive-Bayes Multivariate Gaussian Classifier
- Tied Multivariate Gaussian Classifier

Each of these models were trained and evaluated with a combination of the following settings:

- Raw features
- Normalized features through z-normalization
- Dimensionality reducted features through PCA

2.2 Results

In the following table we report the results for different runs of Gaussian classifier.

	NO PCA					
	RAW features			Z-normalization		
	π = 0.5	π = 0.1	π = 0.9	π = 0.5	π = 0.1	π = 0.9
MVG	0.331	0.608	0.111	0.331	0.608	0.111
MVG-NB	0.475	0.804	0.145	0.474	0.804	0.145
MVG-TIED	0.481	0.710	0.183	0.481	0.710	0.183

Figure 8: MVG Results - NO PCA

PCA=9						
	RAW features			Z-normalization		
	π = 0.5	π = 0.1	π = 0.9	π = 0.5	π = 0.1	π = 0.9
MVG	0.336	0.626	0.111	0.335	0.627	0.111
MVG-NB	0.378	0.745	0.117	0.379	0.745	0.117
MVG-TIED	0.482	0.703	0.181	0.482	0.704	0.181

Figure 9: MVG Results - PCA = 9

PCA=8						
	RAW features			Z-normalization		
	π = 0.5	π = 0.1	π = 0.9	π = 0.5	π = 0.1	π = 0.9
MVG	0.333	0.605	0.110	0.333	0.604	0.110
MVG-NB	0.385	0.750	0.117	0.384	0.751	0.117
MVG-TIED	0.487	0.699	0.182	0.485	0.670	0.182

Figure 10: MVG Results - PCA = 8

PCA=7						
	RAW features			Z-normalization		
	π = 0.5	π = 0.1	π = 0.9	π = 0.5	π = 0.1	π = 0.9
MVG	0.329	0.62	0.112	0.329	0.62	0.111
MVG-NB	0.388	0.747	0.118	0.388	0.747	0.117
MVG-TIED	0.477	0.701	0.181	0.477	0.701	0.180

Figure 11: MVG Results - PCA = 7

2.3 Considerations

Let's start discussing Z-Normalization. The results are the same without it, and this is confirmed by the fact that, as we saw in the previous graphs, the dataset is already centred and, for this reason, we can ignore it.

About PCA, as we supposed before, it doesn't bring so much improvement. We can choose a model with PCA anyway, because if the results are the same respect to a model without it, it is better to have a lighter model, due to the fact that the number of parameters are quadratic respect to the number of dimensions.

As we can see from the table, the model which performs better is the Standard MVG. We can expect this behaviour, because the theories behind other models are not verified in our dataset. The Naive-Bayes assumption is that the different components are approximately independent, and that corrisponds to an MVG Classifier with diagonal covariance matrices.

In our case, as we saw in the heatmaps before, the components, in particular such of them belonging to the non-target class, are a bit correlated. About the Tied MVG, the assumption is that the covariance matrices are the same, and, for this reason, we can consider only the covariance within classes matrix. In our dataset this is not verified, because the two covariance matrices are not similar.

Finally, we can conclude that the best generative model for our target application is the **Standard MVG Classifier**, with **m=7** dimensions.

3 Logistic Regression

3.1 Overview

As we did for the MVG, we're going to list all the models that we're going to analyze in this chapter:

- Logistic Regression
- Quadratic Logistic Regression
- Prior weighted Logistic Regression

Moreover, we have to find the best value of λ, that is the regularization term. This term is useful because favors simpler solutions, and allows reducing the risk of over-fitting the training data. If we use a too big value of λ, our model will be too much general and we will have a poor accuracy for unseen data.

3.2 Results & Considerations

Here, we're going to show how the model behave with different values of λ. As we saw for the MVG, PCA doesn't affect our results, and for this reason we are not going to discuss other dimensions again, but we compare our data only between $l = 7$ dimensions and all dimensions (NO PCA).

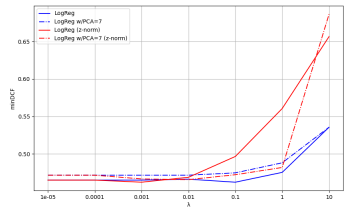

Figure 12: LR with $\pi = 0.5$

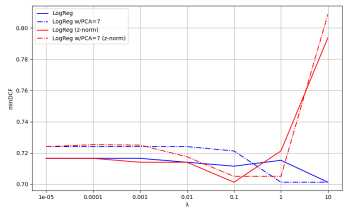

Figure 13: LR with $\pi = 0.1$

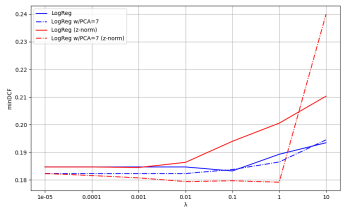

Figure 14: LR with $\pi = 0.9$

The model with the prior $\pi = 0.1$ performs bad because the classes are unbalanced (target class elements are less then non-target class elements), so we can exlude it for the moment. Regarding the other two models, putting all the information inside a table we can select the model that best performs:

With PCA=7	RAW features		Z-normalization	
	π = 0.5	π = 0.9	π = 0.5	π = 0.9
LogReg - λ=0.001	0.471	0.182	0.466	0.180
LogReg - λ=0.1	0.474	0.183	0.472	0.179
Without PCA	**RAW features**		**Z-normalization**	
	π = 0.5	π = 0.9	π = 0.5	π = 0.9
LogReg - λ=0.001	0.464	0.185	0.462	0.184
LogReg - λ=0.1	0.462	0.183	0.496	0.193

Figure 15: Logistic Regression results

So, for the Logistic Regression models, we obtained, even with different combination of parameter λ, more or less the same result. Even in this case, z-normalization doesn't bring significative improvement. We define the best model the **Logistic Regression, with** $\lambda = 0.1$ **and RAW features.**

At this point, we want to analyze the Quadratic Logistic Regression (Q-LogReg), with the same values of λ and π we discussed before. In this case, we ignore the PCA, but we always

want to compare RAW features against z-norm features.

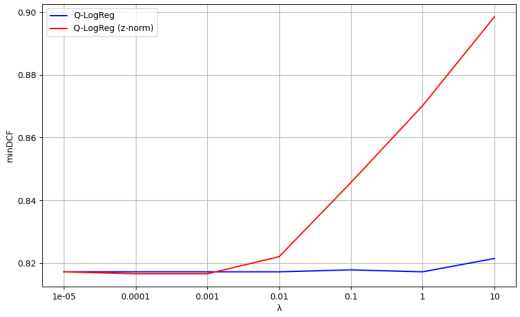

Figure 16: Q-LR with $\pi = 0.5$

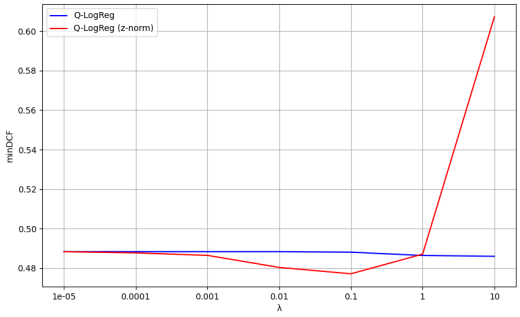

Figure 17: Q-LR with $\pi = 0.9$

The Q-LogReg model has worse perfomance than LogReg. We expected this behaviour because the previous analysis such as scatter plots, histograms and LDA analysis show us that quadratic models do not perform better then the linear one.

Finally, let's consider the Prior-Weighted Logistic Regression and we're going to estabilish which of these discriminative models performs better. Here, we used two different priors for the weights $\pi_{T1} = 0.3$ and $\pi_{T2} = 0.7$.

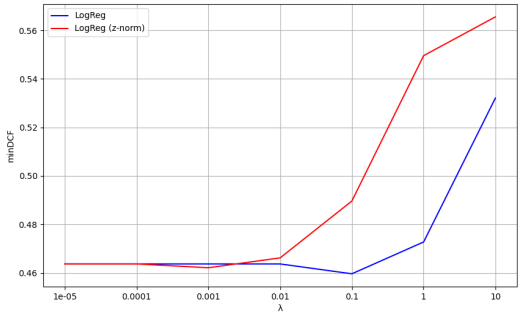

Figure 18: $\pi = 0.5, \pi_T = 0.3$

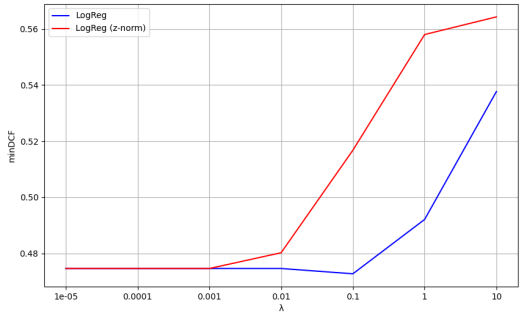

Figure 19: $\pi = 0.5, \pi_T = 0.7$

In this situation, with $\pi = 0.5$, the value of λ which gives the best performance is 0.1

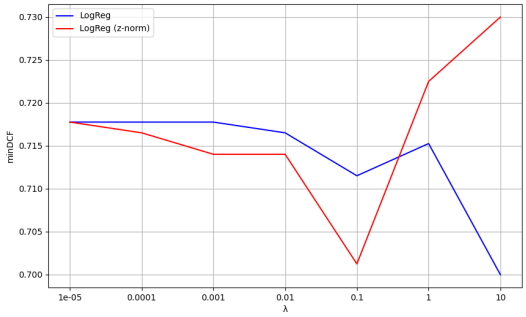

Figure 20: $\pi = 0.1, \pi_T = 0.3$

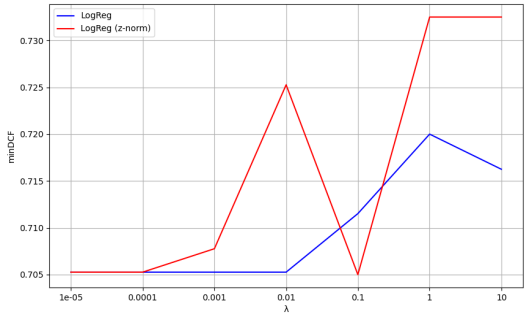

Figure 21: $\pi = 0.1, \pi_T = 0.7$

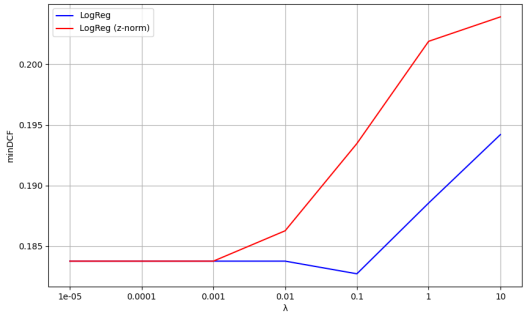

Figure 22: $\pi = 0.9, \pi_T = 0.3$

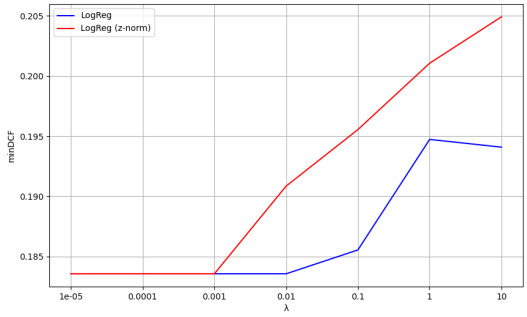

Figure 23: $\pi = 0.9, \pi_T = 0.7$

In the following, we report a comparison between PW-Logistic Regression and the "Classical" one for our target application.

π = 0.5			
Model	Lambda considered	RAW features	Z-normalization
PW-LogReg - π_t = 0.3	λ=0.1	0.460	0.490
PW-LogReg - π_t = 0.7	λ=0.1	0.473	0.517
LogReg	λ=0.001	0.465	0.462

Figure 24: Overall results - LogReg vs PW-LogReg, $\pi = 0.5$

Even if the are few differences between these two models, considering the weight $\pi_t = 0.3$, we can affirm that the **Prior Weighted** model performs sligthly better.

4 Support Vector Machines

4.1 Overview

In this chapter, we are going to analyze the Support Vector Machine model and its variants. In particular, we are going to analyze:

- **Linear Support Vector Machines**: obtained solving the primal problem, where the objective consists in minimizing the

$$J(w,b) = \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n} max(0, 1 - z_i(\mathbf{w^T x_i} + b))$$

  where n is the number of samples, z_i is the class of the i-th sample (its values are 1 for target class and 0 for non-target class).

- **Quadratic (Polynomial with grade $d = 2$) Support Vector Machines**: obtained solving the dual problem, where the objective consist in maximes the:

$$\alpha^T \mathbf{1} - \frac{1}{2}\alpha^T \mathbf{H} \alpha$$

  In the computation of $\mathbf{H}$, we are going to use the polynomial kernel of degree d: $k(\mathbf{x_1}, \mathbf{x_2}) = (\mathbf{x_1^T x_2} + c)^d$

- **SVM with Radial Basis kernel Function (RBF)**: obtained as the dual problem, but the kernel function used will be different:

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

  where γ is an hyperparameter, and defines the width of the kernel.

As our dataset is unbalanced, we used different values of C for different classes. For class balancing, we set $C_T = C\frac{\pi_T}{\pi_T^{emp}}$ and $C_F = C\frac{\pi_F}{\pi_F^{emp}}$

Each model will be evaluated with:

- RAW features
- Normalized features with Z-Score
- PCA used, with $l = 7$ dimensions: we discovered that it is the optimum number of dimensions, as the following plot suggests.

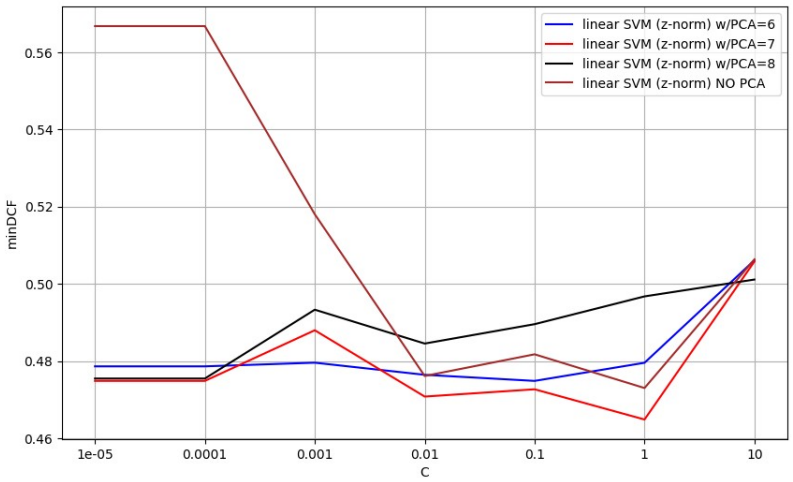

Figure 25: Linear SVM - Comparing PCAs with $\pi = 0.5$ and $\pi_T = 0.5$

4.2 Results & Considerations

In the following paragraphs, we're going to show our runs and our results.

4.2.1 Linear Support Vector Machines

As we saw that linear models performs well, we're going to expect this behaviour even in this case. All the test we made are performed with different values of the hyperparameter C. In particular, we considered an interval of values between 10^{-4} and 10^{2}. In general, C represents the regularization parameter and it controls the trade-off between maximizing the margin and minimizing the classification error. A smaller value of C allows for a larger margin but may lead to more misclassified training examples. A larger value of C reduces the margin to classify more training examples correctly. As we expected, this linear classifier gives us results comparable with the previous analyzed models. Despite other models, in this case, we can notice a relevant improved of performance thanks to normalized features. This is related to the fact that with z-score we centered data with unit variance.

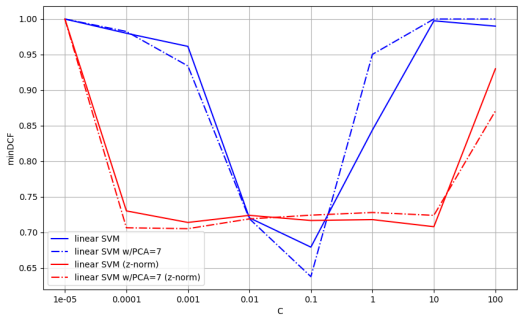

Figure 26: $\pi = 0.1, \pi_T = 0.3$

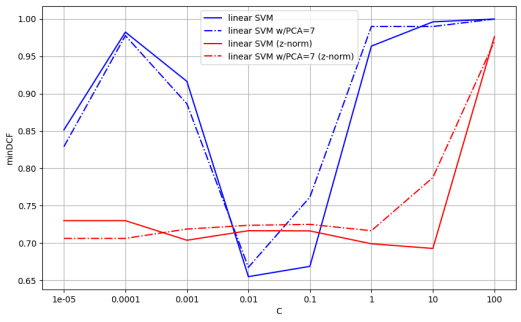

Figure 27: $\pi = 0.1, \pi_T = 0.7$

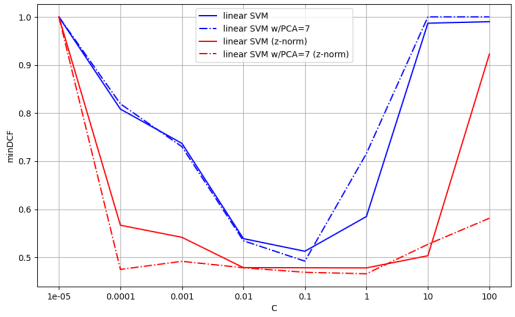

Figure 28: $\pi = 0.5, \pi_T = 0.3$

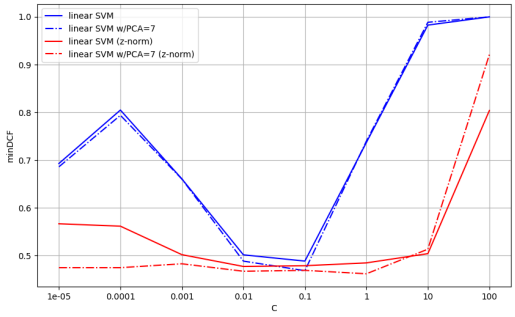

Figure 29: $\pi = 0.5, \pi_T = 0.7$

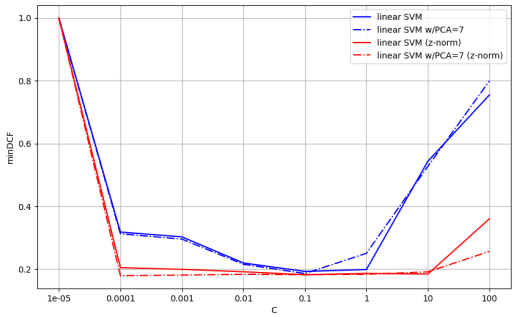

Figure 30: $\pi = 0.9, \pi_T = 0.3$

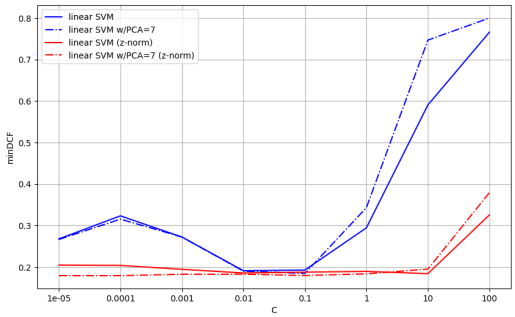

Figure 31: $\pi = 0.9, \pi_T = 0.7$

As we can see, using different values of π_T does not affect the results significantly: even with $\pi_T = 0.5$, results are more or less the same, with slightly improvement. In the following, we report the best result in details. We chose $C = 0.1$, as it optimizes different applications.

$\pi_T = 0.5$ & $C = 0.1$	Overall Linear Support Vector Machines		
Model	**$\pi = 0.5$**	**$\pi = 0.1$**	**$\pi = 0.9$**
SVM RAW feature	0.495	0.692	0.188
SVM Z-NORM feature	0.482	0.700	0.187
SVM RAW feature w/PCA=7	0.480	0.649	0.179
SVM Z-NORM feature w/PCA=7	0.473	0.719	0.182

Figure 32: Overall results - Linear SVM

As we supposed, the score obtained is comparable with the previous models. Looking the target application prior $\pi = 0.5$, all of obtained results are slightly different, but the Linear SVM, z-normalized, with 7 dimensions (PCA) is a bit better than the others.

4.2.2 Quadratic Support Vector Machines

Even in this case, we used the same range of values for C.

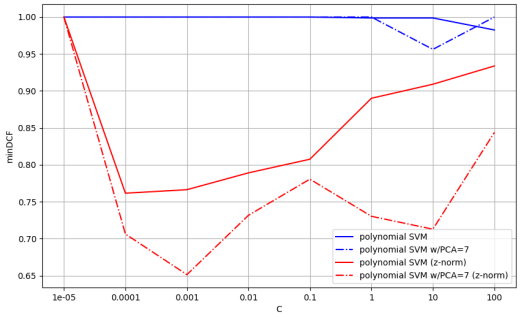

Figure 33: $\pi = 0.1, \pi_T = 0.3$

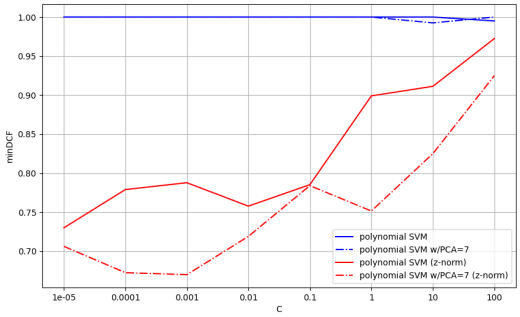

Figure 34: $\pi = 0.1, \pi_T = 0.7$

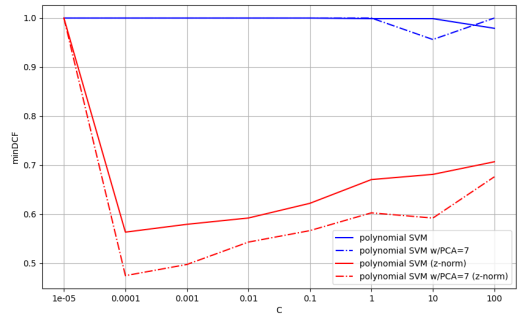

Figure 35: $\pi = 0.5, \pi_T = 0.3$

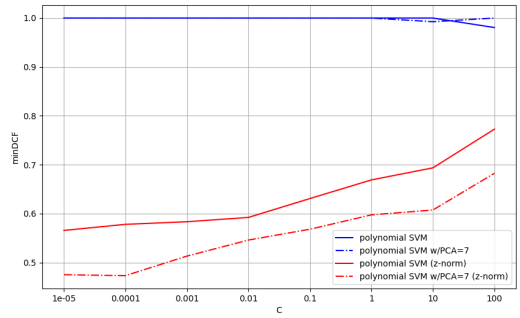

Figure 36: $\pi = 0.5, \pi_T = 0.7$

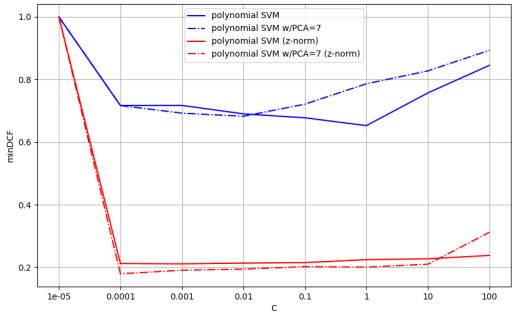

Figure 37: $\pi = 0.9, \pi_T = 0.3$

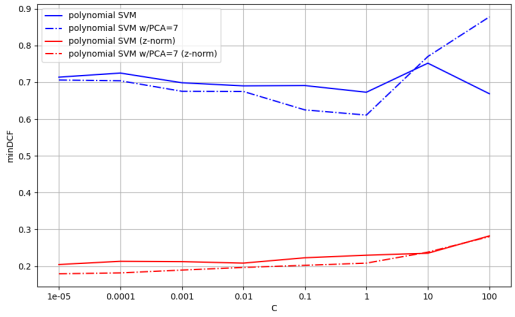

Figure 38: $\pi = 0.9, \pi_T = 0.7$

Even in this case, different values of π_T bring more or less the same performance, and, for this reason, we're going to present the overall table with $\pi = 0.5$. The best value of C, taken thanks to the cross validation, is $C = 0.0001 = 10^{-4}$, and we are going to consider only this one.

$\pi_T = 0.5$ & $C = 0.0001$	Overall Polynomial Support Vector Machines		
Model	$\pi = 0.5$	$\pi = 0.1$	$\pi = 0.9$
Poly-SVM RAW feature	0.577	0.982	0.320
Poly-SVM Z-NORM feature	0.566	0.730	0.205
Poly-SVM RAW feature w/PCA=7	0.577	0.985	0.315
Poly-SVM Z-NORM feature w/PCA=7	0.480	0.706	0.179

Figure 39: Overall results - Polynomial SVM

This table shows how the Polynomial with $d = 2$ (Quadratic) performs worse then Linear SVM, as we expected. Even in this case, RAW features performs extremely bad respect their Z-Normalization, and, if we consider also the dimensionality reduction with the PCA, the difference between them increases.

4.2.3 RBF Support Vector Machines

With RBF SMV we can tune two hyperparameters at the same time (C and γ). So, we are going to plot the behaviour of the model with different combinations of C and γ. In the following plot, we want to find the optimum value of γ, given $\pi_T = 0.5$ and $\pi = 0.5$:

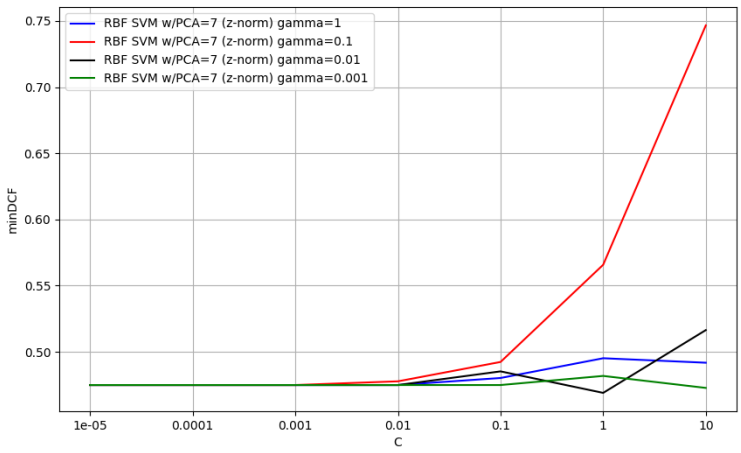

Figure 40: γ and C

So, we can establish that the optimum value of γ is 0.01, and, with it, the others values of C does not affect the score too much.

Starting from now, we're going to test the models with $\gamma = 0.01$. In the following plot, we show the behaviour of RBF SVM for our three different application ($\pi = 0.5$, $\pi = 0.1$ and $\pi = 0.9$). We are going to use feature dimensions reduced by PCA ($l = 7$).

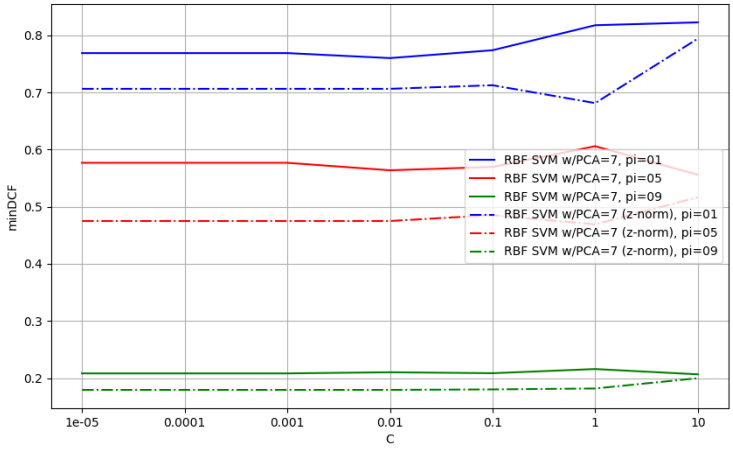

Figure 41: Comparison for different runs of C, with $\gamma = 0.01$

We can notice that RBF SVM behaves differently from other SVM tested models. In fact, the model maintains more or less the same score even for different values of C. Here, we present them in details, setting the value of $\pi_T = 0.5$, $\gamma = 0.01$ and $C = 0.01$:

$\pi_T = 0.5$ & C = 0.01 & γ = 0.01	Overall RBF Support Vector Machines		
Model	π = 0.5	π = 0.1	π = 0.9
RAW feature w/PCA=7	0.563	0.760	0.210
Z-NORM feature w/PCA=7	0.475	0.706	0.180

Figure 42: Overall results - RBF SVM

Considering our target application prior, we can notice that RBF model behaves slightly better than quadratic one, even if it is not good as the linear one.

4.2.4 Results

Let's conclude our analysis regarding SVM, comparing the best results for each model. We are going to assume these parameters as fixed: $\pi_T = 0.5$, dimensionality reduced features

($l = 7$),

	Overall conclusion - Support Vector Machines		
Model	π = 0.5	π = 0.1	π = 0.9
Linear - RAW feature w/ C=0.1	0.480	0.649	0.179
Linear - Z-NORM feature w/C=0.1	0.473	0.719	0.182
Poly - Z-NORM feature w/ C = 0.0001	0.480	0.706	0.179
RBF Z-NORM w/ C = 0.01 & γ = 0.01	0.475	0.706	0.180

Figure 43: Overall results - SVM models

For the SVMs, we can conclude that, for our target application, the best model is **Linear SVM, with z-normalized features and C=0.1, and l=7 dimensions considered with PCA**.

5 Gaussian Mixture Models

5.1 Overview

In this chapter, we are going to analyze the last classifier: Gaussian Mixture Models. The hyperparameter that we are going to tune is the number of Gaussian components, for both type of features (raw and z-normalized). In particular, we're going to test with $N = 2, 4, 8$ components. We can't consider an higher number of features due to the very expensive computational cost.

We're going to analyze different combination of models and covariance matrices:

- Mode: **Full** && Cov: **Tied**
- Mode: **Full** && Cov: **Untied**
- Mode: **Diagonal** && Cov: **Tied**
- Mode: **Diagonal** && Cov: **Untied**

5.2 Results

Given MVG's results, we expect this classifier to perform very well. We analyzed all 4 versions both on raw features and on features on which PCA = 7 was applied.
Due to our dataset characteristics, tied and untied model perform in the same way, so we're going to report only "Full tied" and "Diagonal tied" results.
Firstly, let's start to check if PCA (with $l = 7$)is needed or not.

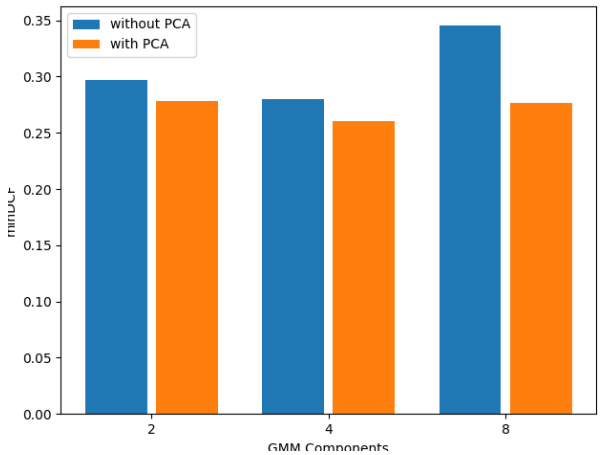

Figure 44: PCA Comparison - full tied

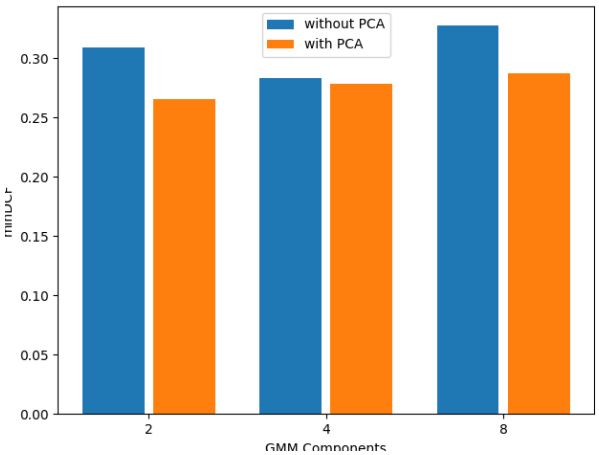

Figure 45: PCA Comparison - znorm full tied

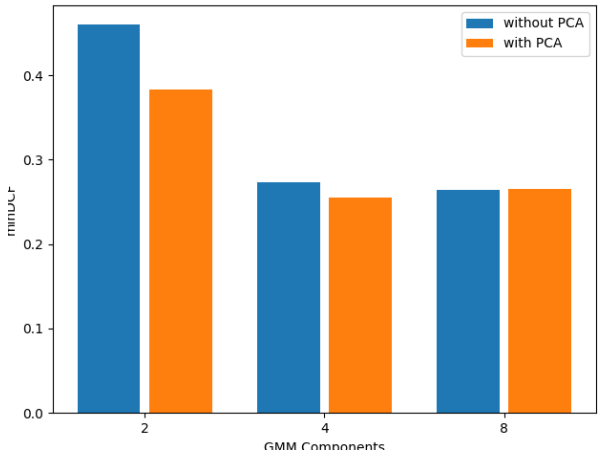

Figure 46: PCA Comparison - diagonal tied

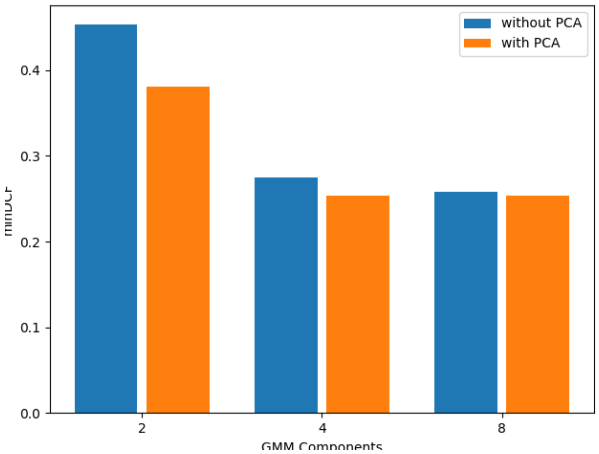

Figure 47: PCA Comparison - znorm diagonal tied

From the previous plots, we can state that PCA is slighter useful but the z-normalization is useless. For this reason, we're going to consider only models with PCA but without z-normalization.

5.3 Considerations

The best model is **Diagonal tied, with number of GMM components N=4 and PCA=7**. We're going to present them in details even for different applications.

	Overall GMM w/ N=2 & PCA=7		
Model	π = 0.5	π = 0.1	π = 0.9
Full	0.278	0.456	0.098
Diagonal	0.383	0.734	0.110

Figure 48: Overall GMM with N=2

N=4 & PCA=7	Overall GMM w/ N=4 & PCA=7		
Model	π = 0.5	π = 0.1	π = 0.9
Full	0.260	0.452	0.091
Diagonal	0.254	0.500	0.088

Figure 49: Overall GMM with N=4

N=8 & PCA=7	Overall GMM w/ N=8 & PCA=7		
Model	π = 0.5	π = 0.1	π = 0.9
Full	0.276	0.568	0.096
Diagonal	0.265	0.495	0.086

Figure 50: Overall GMM with N=8

6 Calibration

6.1 Best evaluation models

In the following table, we're going to present the top three models that we encountered during our validation phase.

	Validation - TOP 3		
Model	π = 0.5	π = 0.1	π = 0.9
MVG - Standard & PCA=7	0.329	0.62	0.112
PW-LogReg & π_t = 0.3 & λ=0.1	0.460	0.702	0.187
GMM Diagonal - N=4 & PCA=7	0.265	0.495	0.086

Figure 51: Top 3 models

Finally, we can state that the best model among all the others is **Diagonal-Tied GMM with PCA=7 and N=4 components**

6.2 Score calibration

Minumum detection cost function (minDCF) results depends on the used treshold. In order to understand if the treshold is the theoritical one we can choose the metric known as **actual detection cost function (actDCF)**. In order to understand this, we can compare these models with actDCF and minDCF: if they're equals, we can say the score is calibrated. Otherwise, we need a calibration.

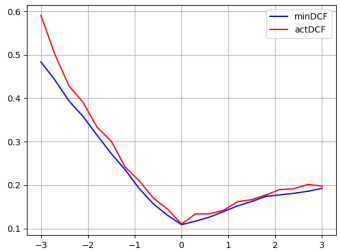

Figure 52: Bayes error plot - MVG

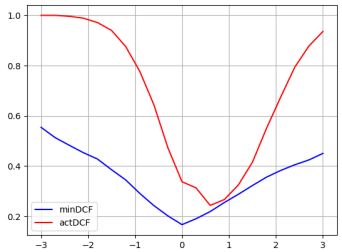

Figure 53: Bayes error plot - PW LOGREG

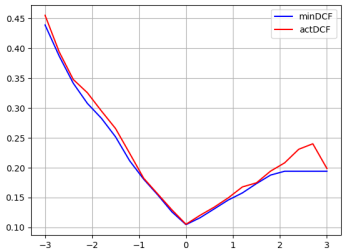

Figure 54: Bayes error plot - GMM

As we can see from the plots, only the Prior Weighted Logistic Regression is mis-calibrated. For this reason, we're going to use **Prior Weighted Logistic Regression** as a Calibration approach.

In the following, we're going to present the Bayes error plot, after we calibrated the score with the technique we mentioned before, using $\lambda = 0.01$

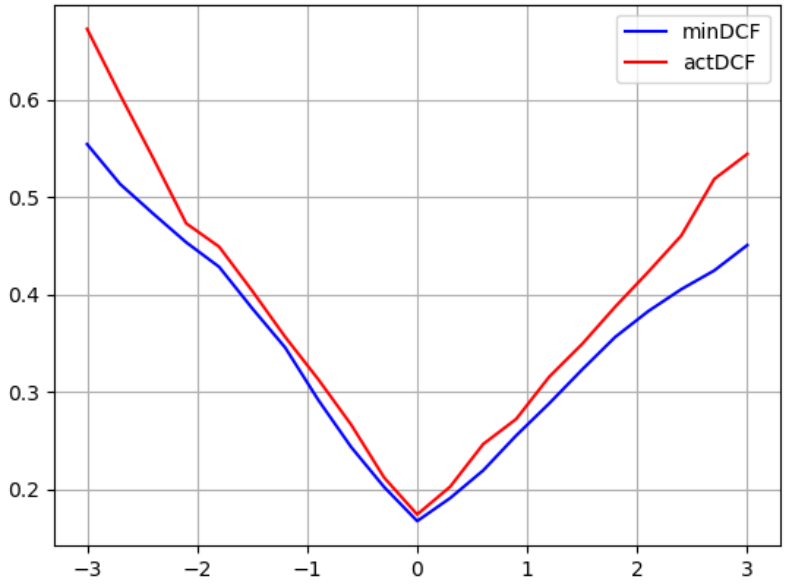

Figure 55: LOGREG after calibration

7 Evaluation

Now we're going to test the previous presented models, using the whole dataset and its original suddivision. We will yet again consider minDCF measures in order to verify whether the proposed solution can indeed achieve the best accuracy. We can expect the new minDCF measures to be slightly different than the previous ones, since the distribution of the data is not exactly the same in the training and test set. If the distribution is similar enough and our assumptions were correct, though, the difference should be very small.

7.1 Gaussian Mixture Model

We start our analysis with the model that has the best performance.

Even if we obtained the best performance with **Diagonal-Tied GMM with PCA=7 and N=4 components**, we evaluate the model even with z-normalization and without PCA.

	Evaluation - GMM		
Model	π = 0.5	π = 0.1	π = 0.9
Raw Features	0.218	0.502	0.0779
Raw Features & PCA = 7	0.229	0.431	0.0764
Znorm	0.227	0.501	0.0766
Znorm & PCA=7	0.229	0.431	0.0765

Figure 56: Evaluation - MVG

From the table, we can see how z-normalized features doesn't bring improvement.

Unlike the validation, the best result is obtained without PCA for the target application. If we want to consider different applications, the usage of PCA confirms a slightly improvement.

7.2 Multivariate Gaussian Model

Our best model was **MVG Standard with PCA=7**, but we even tested the **Naive-Bayes** with and without PCA.

	Evaluation - MVG		
Model	**π = 0.5**	**π = 0.1**	**π = 0.9**
Standard Raw Features	0.276	0.545	0.082
Standard Raw Features & PCA = 7	0.272	0.580	0.083
NB Raw Features	0.352	0.718	0.110
NB Raw Features & PCA = 7	0.312	0.631	0.086

Figure 57: Evaluation - MVG

As we can see from the table, even in the test phase, PCA brings an improvement.
During the evaluation phase, the standard model behaves even better than the same model in the validation phase probably due to the dimension of the training set.

7.3 Logistic Regression

During our validation phase, we discovered that "Classic" Logistic Regression and Prior-Weighted Logistic Regression with $\pi_T = 0.3$ had very similar results. For this reason, we tested both, with a λ value between 10^{-4} and 10^{-1}. We considered even the case without PCA and with z-normalized features.

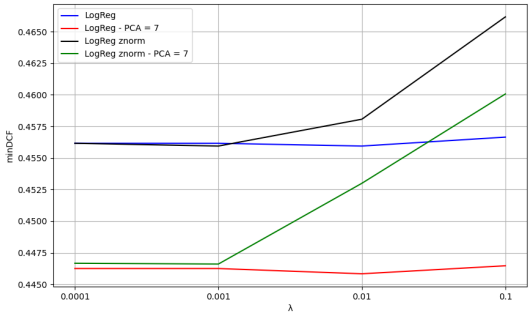

Figure 58: Logistic Regression - $\pi = 0.5$

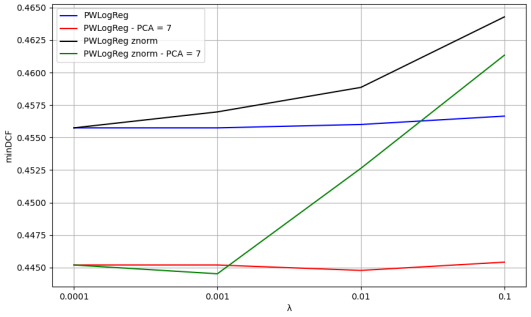

Figure 59: PW-Logistic Regression - $\pi = 0.5$

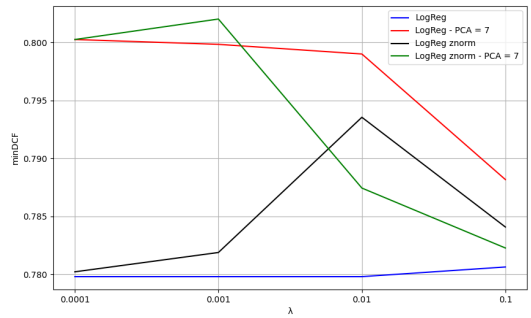

Figure 60: Logistic Regression - $\pi = 0.1$

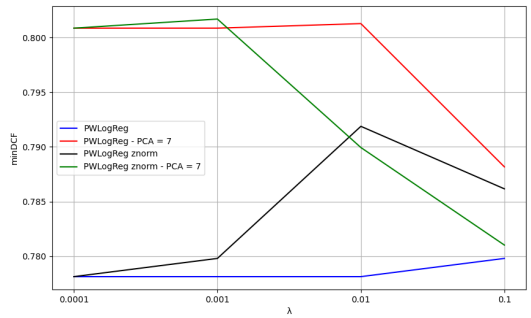

Figure 61: PW-Logistic Regression - $\pi = 0.1$

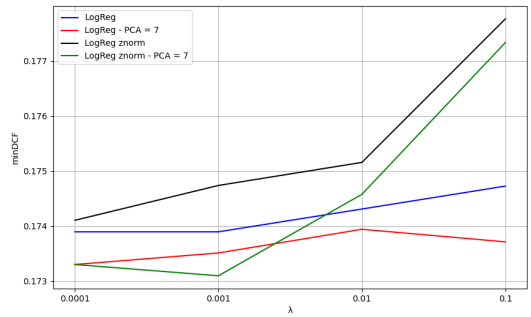

Figure 62: Logistic Regression - $\pi = 0.9$

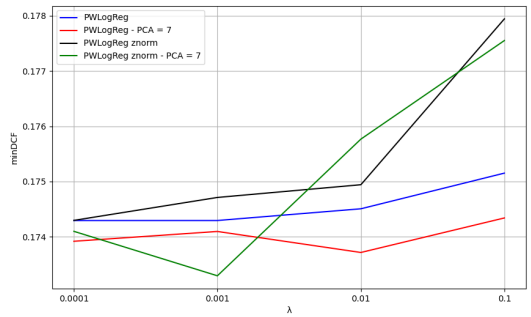

Figure 63: PW-Logistic Regression - $\pi = 0.9$

For a better comparison, we report the results in a table.

	Evaluation - LogReg		
Model with λ=0.001	π = 0.5	π = 0.1	π = 0.9
LogReg w/Raw Feat. & PCA=7	0.4462	0.7998	0.1735
LogReg w/znorm & PCA=7	0.4466	0.8020	0.1730
PWLogReg w/Raw Feat. & PCA=7	0.4451	0.8008	0.1740
PWLogReg w/znorm & PCA=7	0.4445	0.8016	0.1732

Figure 64: Evaluation - LOGREG

Even in the evaluation phase, the Prior Weighted model performs better than the "Classic" one, even with different value of $\lambda = 0.001$, and with z-normalization and PCA.

8 Conclusion

The following table reports an overview of the results during the evaluation, considering all the models presented before.

	Evaluation - OVERALL		
Model	π = 0.5	π = 0.1	π = 0.9
Multivariate Gaussian Model	0.272	0.580	0.083
PW Logistic Regression	0.445	0.8016	0.1732
Gaussian Mixture Model	0.218	0.502	0.0779

Figure 65: Evaluation - Overall models

The choices and strategies used during the training phase have proven effective when applied to test data.

In conclusion, we can state that the best model is: **Gaussian Mixture Model with N=4 components, no PCA, no Z-Normalization**, with a **minDCF** score of 0.218.