Politecnico di Torino

Language Detection Project

Cappello Alessio s309450, Gennuso Angelo s310082

Year 2022/2023

Abstract

In this report, what we intend to study is the efficiency, performances and results of the application of basic Machine Learning algorithms to some given datasets.

In our case, what we are studying in this context is a simulated dataset that tries to represent as much as possible the distribution of utterances based on language.

The problem is presented as a Binary Classification problem. What we are doing concretely is classifying if a determined utterance is spoken in a determined Language or not. In our specific case, we are separating the target class utterances from all the others.

We will present in order: a brief description of the problem, some features' analysis, the training results with various classification algorithms, an evaluation of some models proposed and finally our conclusions.

This report is written for the Machine Learning and Pattern Recognition course at Politecnico di Torino.

Contents

1 Introduction

1.1 Task explanation

For this project, we have been presented with a Language Detection task. The objective of this classification task is to detect whether a certain utterance is spoken in a pre-determined target language (i.e. if the target language is Italian, we want to check whether an utterance is spoken in Italian or not).

The utterances are represented as language embeddings (i.e. audio speeches mapped to a low-dimensional manifold). In our specific case, which we consider a simple one, the number of dataset dimensions is 6.

We are given two datasets (train and test): these are imbalanced because the target language class has fewer samples than the non-target language one (the latter comprehends various languages and an utterance of this class belongs to one of 25 possible languages).

The following are the considered target applications:

1. $(\pi_T = 0.5, C_{fn} = 1, C_{fp} = 1)$
2. $(\pi_T = 0.1, C_{fn} = 1, C_{fp} = 1)$

To evaluate the models' performance, we define a primary metric C_{prim}, defined as the average minimum cost for the two working points.

2 Data Analysis

2.1 Feature Analysis

As already stated, we are in a simplified context of a real use-case scenario. We are given two datasets:

- Training set samples: 2371
- Test set samples: 4403

The datasets are unbalanced: respectively, they contain 400 and 800 samples belonging to the target class.

2.2 Distribution of Data

To begin with, we visualize how training data is distributed. To do this we construct a histogram and a 2D scatter plot based on the training samples. We apply PCA 2 to construct the scatter plot from a 6-dimensional dataset.

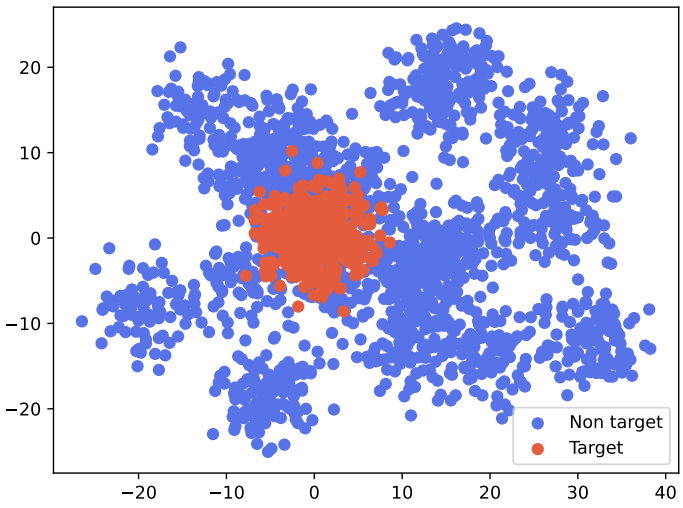

Figure 1: 2D Scatter Plot

As for the histograms, we decide to represent just the 2 dimensions we used for the scatter plot.

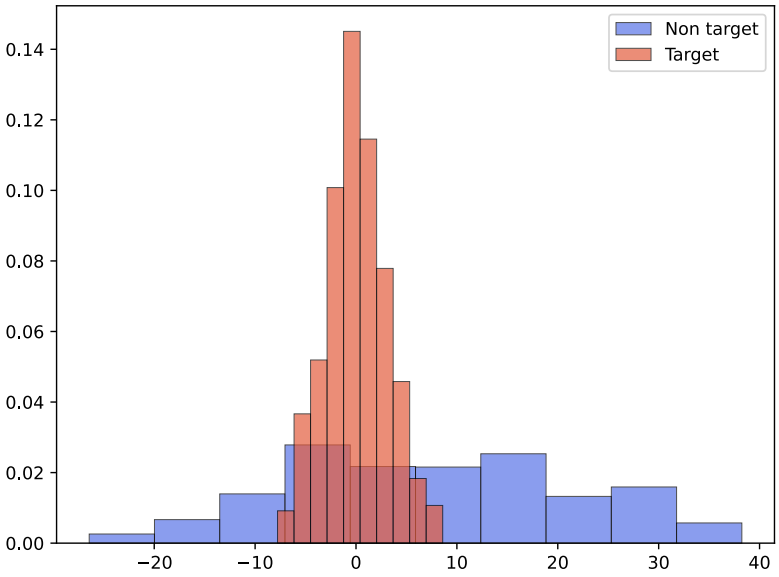

Figure 2: PCA - Main Component

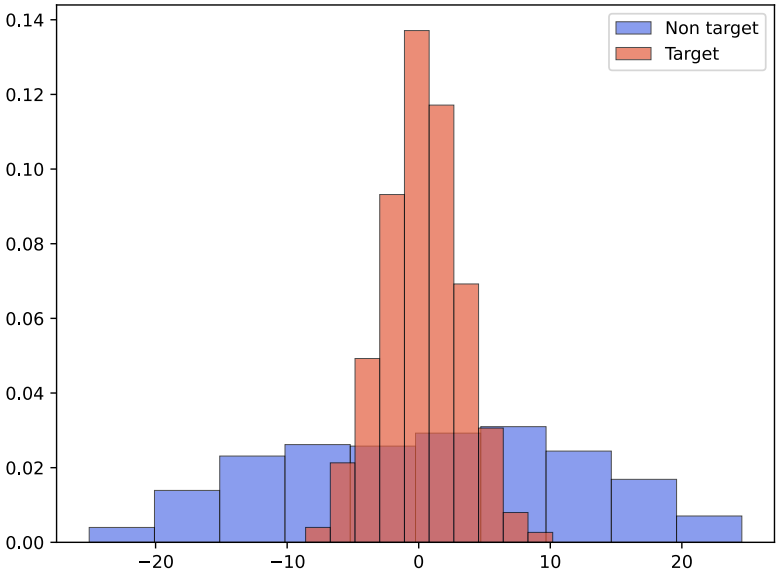

Figure 3: PCA - 2nd Component

Looking at these results, it is clear that the target class is concentrated in the centre of the scatter plot, and we can also notice various clusters in the non-target class. It is also worth stating that the two histograms represent a near-Gaussian Distribution on the target class.

We could also look at the correlation between features. To do this, we first compute the covariance matrix for the dataset using the following formula:

$$CovMatr = \frac{1}{N}\sum_{i=1}^{6}(x_i - \mu)(x_i - \mu)^T$$

We then need to compute the correlation matrix. We calculate the Pearson Correlation Coefficient for each feature pair, using the following formula:

$$Cov_{XY} = \frac{\sigma_{XY}}{\sigma_X \sigma_Y}$$

where X and Y are the considered features.

The output is a 6x6 matrix which we decide to represent as a heatmap. This process can be done also for the target class only and the non-target one: we reported below the results of the process.

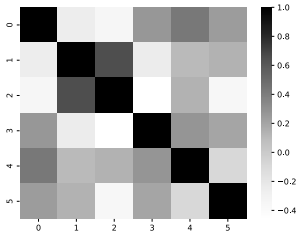

Figure 4: Dataset

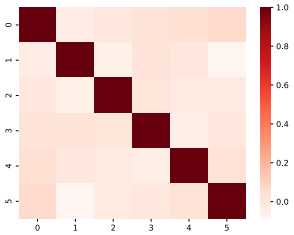

Figure 5: Target Class

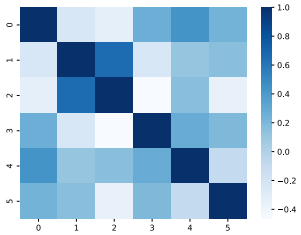

Figure 6: Non-Target Class

The features of the target class are lowly correlated: this could be a sign of the possible effectiveness of the Naive Bayes hypothesis, but it could not work properly on the non-target class. PCA may help in reducing correlation, but we have to pay attention because we are in a low-dimensionality context.

We also used LDA to see if the classes are linearly separable in order to assess if linear models can be suitable for this task or not.

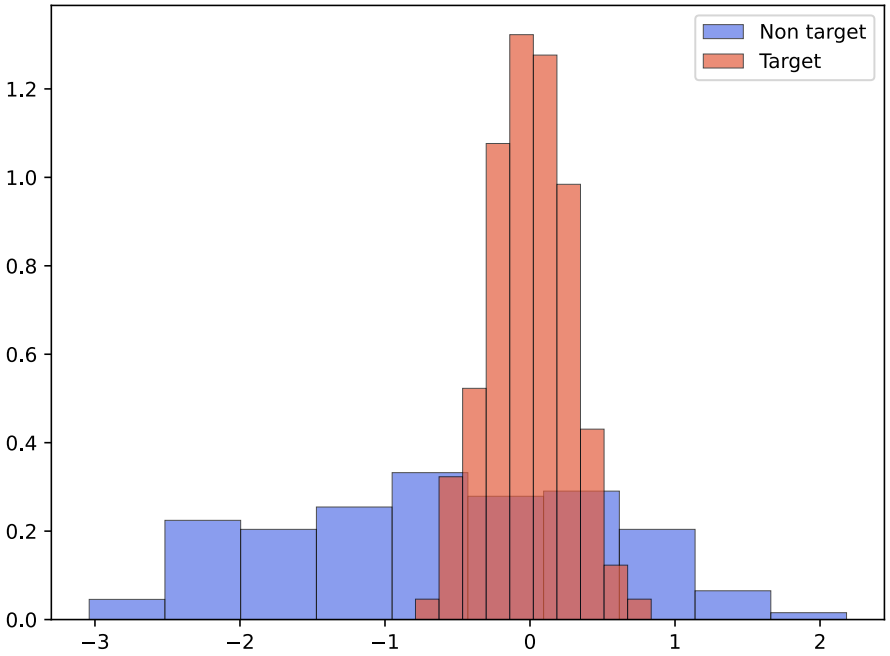

Figure 7: LDA Histogram

As we can see from the histogram, the curves are overlapping: this indicates the poor performance that linear models can have on this task.

2.3 Possible Preprocessing

We now focus on looking at some possible preprocessing techniques we can adopt to treat the data. To start with, we can look at the PCA. The possible range of PCA in our case is from 1 to 6. To evaluate how effective can this strategy be, we can look at the percentage of explained variance for all the various stages.

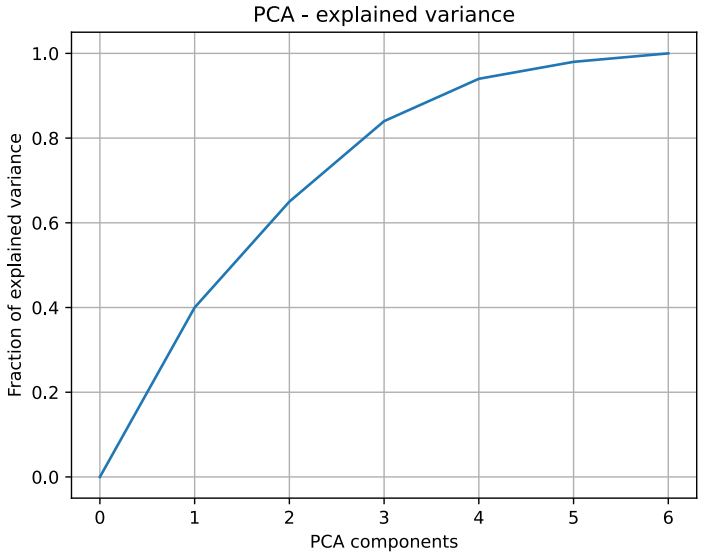

Figure 8: PCA Explained Variance

As we said before, we need to pay attention because we have a low-dimensional dataset, so we should not exaggerate in reducing dimensions in order to not lose lots of information useful to address the classification problem.

We also take into account Z-Normalization, which we try to use in some classification models later on. Given a dataset feature and having computed its mean and its variance vector, we can compute the Normalized feature vector with the subsequent formula:

$$Z = \frac{X - \mu}{\sigma}$$

If applied to all the features in the dataset, we obtained a Standardized dataset.

3 Classification

3.1 Specifics

For the whole process, we will use the K-Fold cross-validation technique with $K = 5$: in this way, we can compensate for the small amount of training data with the usage of a validation set.
As stated in the introduction, we are going to train the application on two working points. Considering this, we will use the $minDCF$ approach, computing its value for each working point and then averaging the two, obtaining our previously defined cost metric C_{prim}.
As for the actual value of the DCF, $actDCF$, we will compute its value and calibrate the scores only after we have chosen a model as the most promising one.

3.2 Gaussian Classifiers

We will analyze the results without PCA and using PCA 5 and PCA 4. For the Naive-Bayes we will also use PCA 6: this is done because the MVG and Tied are invariant to full PCA, while the Naive is not. What we said means that using PCA 6, we would get the same results as No PCA for the first two, whereas we have different results for the latter.
Looking at the distributions of the data from the previous histograms and scatter plots, as already stated, we can see that the data for the target class is following a near-Gaussian distribution, while the data for the non-target class seems to be composed of some clusters. Given this fact, we could expect that the MVG performs better than just a linear classifier (we will discuss the possibility of exploiting the clusters, instead, in the GMM section).
As also seen in heatmaps, the target class features are barely correlated. On the other hand, the non-target class one shows that they are instead. This tells us that the Naive could work better than the standard MVG (even though the non-target class features are more correlated, it may help in reduce overfitting).
As seen in the LDA results and also in the original scatter-plot, the data cannot be considered linearly separable, so we expect the results of the Tied to not be good.

PCA	minDCF ($\tilde{\pi} = 0.5$)	minDCF ($\tilde{\pi} = 0.1$)	min C_{prim}
-	0.1318	0.5023	0.3171
5	**0.1295**	**0.4938**	**0.3117**
4	0.1344	0.5428	0.3386

Table 1: MVG results

PCA	minDCF ($\tilde{\pi} = 0.5$)	minDCF ($\tilde{\pi} = 0.1$)	min C_{prim}
-	0.1379	0.5465	0.3422
6	**0.1292**	0.4847	0.3069
5	0.1301	**0.4727**	**0.3014**
4	0.1318	0.5384	0.3351

Table 2: MVG Naive-Bayes results

PCA	minDCF ($\tilde{\pi} = 0.5$)	minDCF ($\tilde{\pi} = 0.1$)	min C_{prim}
-	0.5157	1.0	0.7578
5	0.5105	1.0	0.7552
4	**0.4831**	1.0	**0.7416**

Table 3: MVG Tied results

As we can see from the results, the PCA-4 preprocessing performs rather poorly. This means that we shouldn't proceed further in reducing the number of dimensions under 5, since it is taking away too much information useful for classification. Furthermore, the Tied model performs quite badly because of the difference between the target class distribution and the non-target class one.

3.3 Logistic Regression

Moving to Logistic Regression models, given the fact that the classes are not linearly separable, we expect the Linear Logistic Regression to perform poorly. Let's start visualizing the effect of preprocessing strategies on the linear model: in our case, we choose PCA5 and Z Normalization. We will conduct our analysis looking for the best value of the hyperparameter λ.

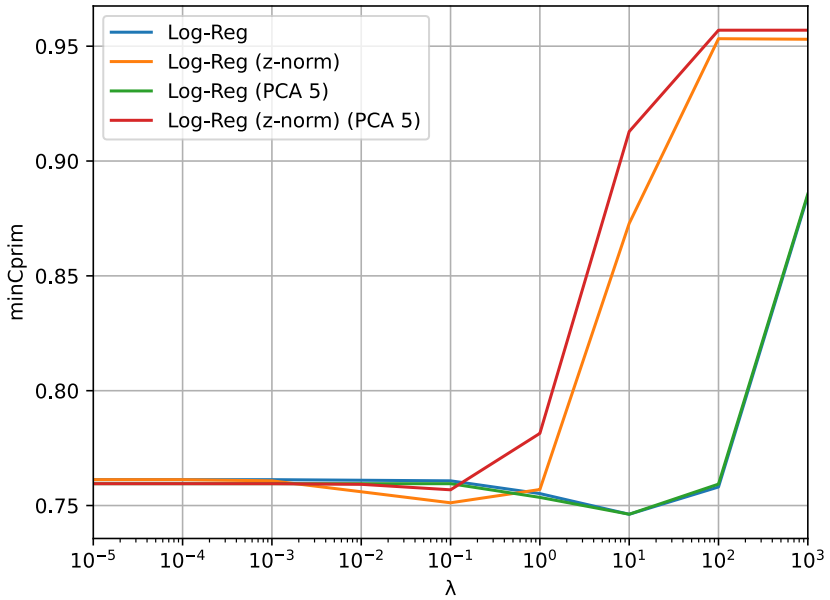

Figure 9: Linear Logistic Regression

As we expected, we can see that the linear model doesn't bring good results. Let's apply a quadratic feature expansion to get more task-suitable separation rules. For the moment, we keep also the Z Normalization.

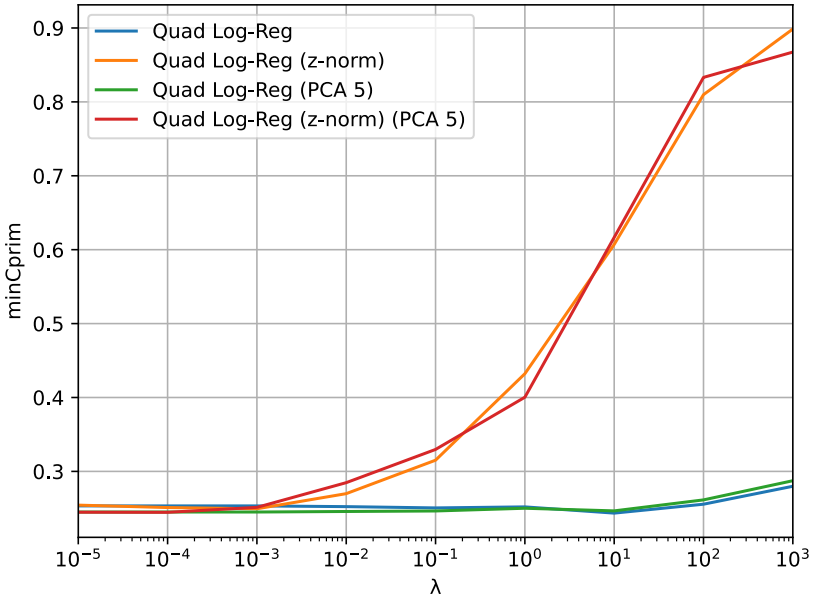

Figure 10: Quadratic Logistic Regression

As we can see, the quadratic version of the Logistic Regression leads to better results. Z Normalization does not look useful, so we discard it from our analysis. Given the results, it may be worth trying a prior-weighted version of the model. The following tables show only the best values of λ for the prior-weighted version, raw and with PCA5.

Prior	λ^*	minDCF ($\tilde{\pi} = 0.5$)	minDCF ($\tilde{\pi} = 0.1$)	min C_{prim}
Emp. (Fold)	10	0.1038	0.3833	0.2435
Emp. (Dataset)	10	0.1027	0.3833	0.2430
0.1	10	0.1031	**0.3809**	**0.2420**
0.2	10	0.1023	0.3833	0.2427
0.5	10	**0.1012**	0.3837	0.2425

Table 4: Quadratic Logistic Regression - No PCA

Prior	λ^*	minDCF ($\tilde{\pi} = 0.5$)	minDCF ($\tilde{\pi} = 0.1$)	min C_{prim}
Emp. (Fold)	10^{-3}	**0.0953**	0.3947	0.2450
Emp. (Dataset)	10^{-3}	**0.0953**	0.3947	0.2450
0.1	10^{-3}	0.0953	0.4142	0.2461
0.2	10^{-1}	0.0990	0.3916	0.2452
0.5	10^{-1}	0.0955	**0.3898**	**0.2427**

Table 5: Quadratic Logistic Regression - PCA 5

PCA5 is able to lead to good results: however, we obtain the best using raw data. So, for this classifier, we decide to select prior-weighted Quadratic Logistic Regression with π_T=0.1 and λ=10 (no PCA).

Prior	λ^*	minDCF ($\tilde{\pi} = 0.5$)	minDCF ($\tilde{\pi} = 0.1$)	min C_{prim}
0.5	10^{-1}	0.1031	0.3809	0.2420

Table 6: Quadratic Logistic Regression - Best configuration (no PCA)

3.4 SVM: Support Vector Machines

As stated previously, the LDA preprocessing made clear that classes are not linearly separable. As seen with Logistic Regression models, we expect the linear SVM to perform poorly. For the entire SVM analysis, we fix the value of K to 1, while we search for the best value of the hyperparameter C. We report the plot of the Linear SVM, combined with PCA 5 and Z Normalization.

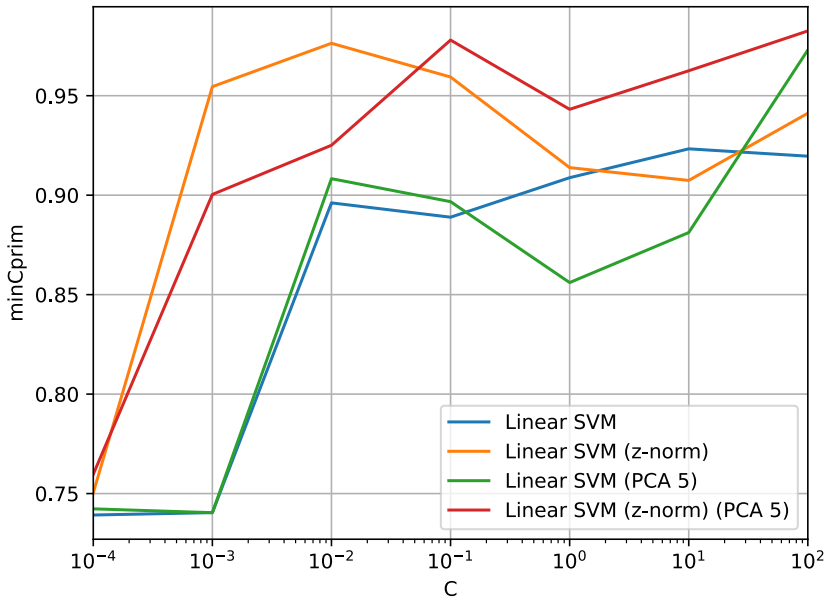

Figure 11: Linear SVM

As we expected, we don't get impressive results due to the class being non-linearly separable. For the SVM we will then apply some kernel methods to bring forth our analysis:

- Polynomial Kernel
- Radial Basis Function (RBF)

Starting from the Poly Kernel, we fix c to 1. We start considering a quadratic kernel (d=2). Let's start again using the same preprocessing strategies as above.

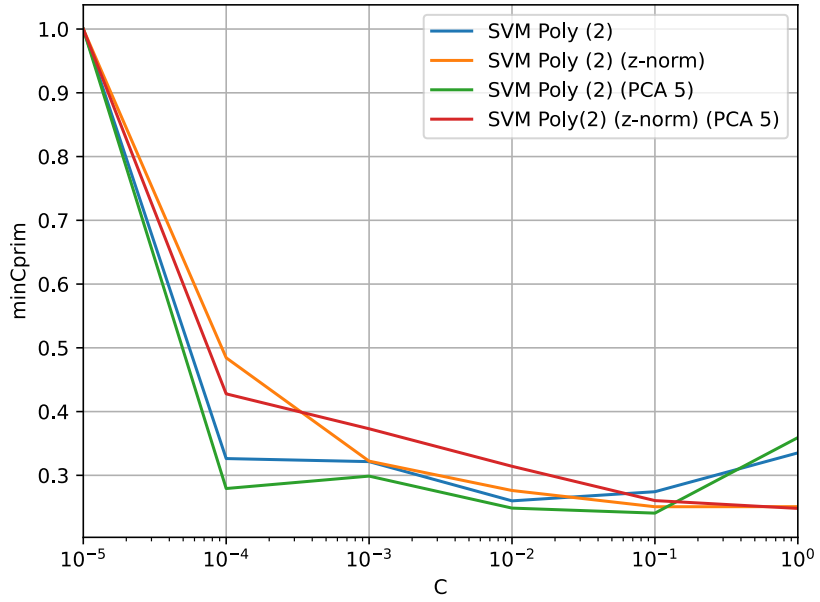

Figure 12: SVM - Kernel Poly ($d = 2$)

Again, we got better results using quadratic separation rules.
Let's give a glance also to d equal to 3. We conducted a brief search only on the most promising values of C.

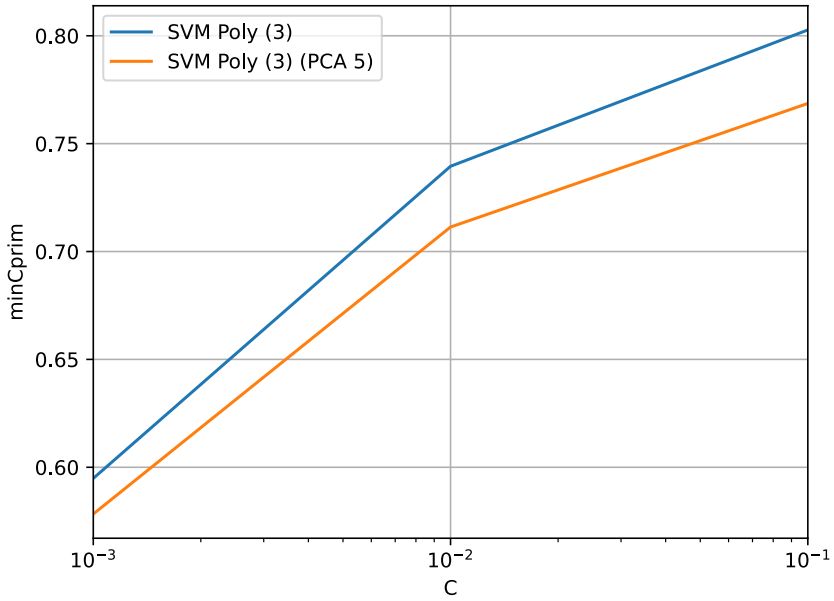

Figure 13: SVM - Kernel Poly ($d = 3$)

The results are way worse than the Quadratic Kernel ones, probably due to overfitting, so we won't consider this case in our analysis.

Moving to RBF Kernel, we need to look for the best value of γ, which indicates the strength of the kernel. We started again considering the usual preprocessing strategies that characterized our analysis up to now.

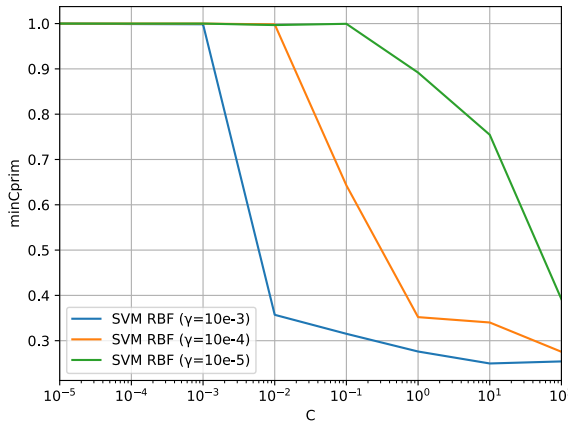

Figure 14: RBF (no PCA)

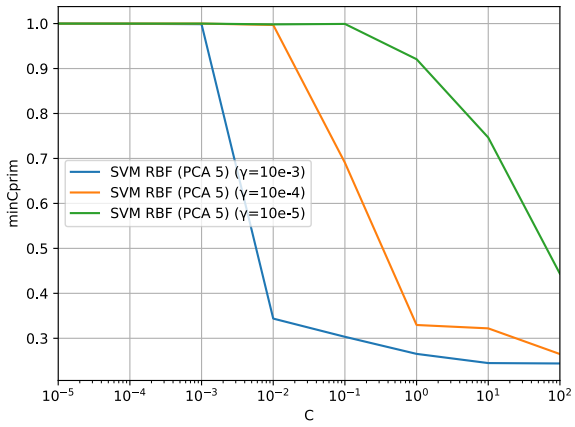

Figure 15: RBF (PCA 5)

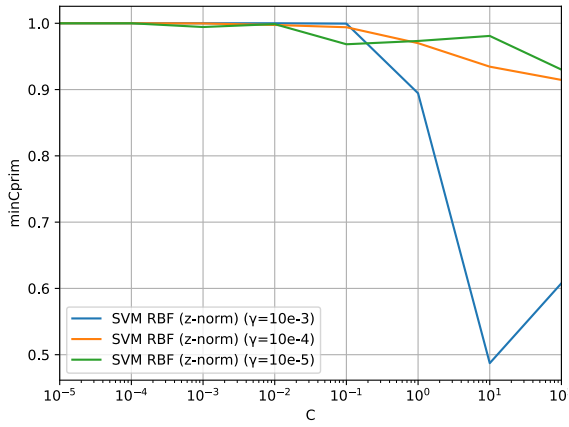

Figure 16: RBF (Z Norm.)

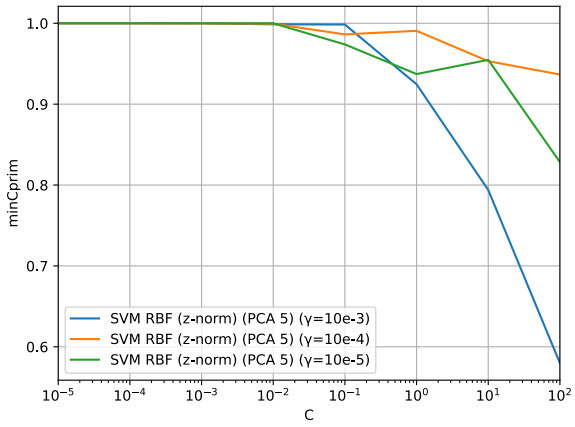

Figure 17: RBF (PCA 5 + Z Norm.)

From these graphs, we can state that PCA is more effective than Z Normalization for RBF kernel. The two considered kernel functions look promising: after a more or less complex grid search, we report some of the best results we got trying different configurations of preprocessing and hyperparameters.

Pre-processing	Kernel	C*	minDCF ($\tilde{\pi} = 0.5$)	minDCF ($\tilde{\pi} = 0.1$)	min C_{prim}
No PCA	Poly ($d = 2$)	$10^(-2)$	0.1092	0.4107	0.2599
PCA 5	Poly ($d = 2$)	$10^(-1)$	0.1033	**0.3781**	**0.2407**
No PCA + Z-Norm	Poly ($d = 2$)	$10^(-1)$	0.1029	0.3985	0.2507
No PCA	RBF ($\log \gamma = -3$)	10	0.0974	0.4020	0.2497
PCA 5	RBF ($\log \gamma = -3$)	10^2	**0.0954**	0.3925	**0.2440**

Table 7: SVM - Poly and RBF Kernel results

Given the closeness of the two best results, we decide to investigate more on them. As we did previously, we move to prior-weighted versions of these: we report just the best value of C we got for each considered prior.

Prior	C*	minDCF ($\tilde{\pi} = 0.5$)	minDCF ($\tilde{\pi} = 0.1$)	min C_{prim}
Polynomial (degree 2) Kernel - PCA = 5				
Emp. (fold)	10^{-1}	0.1030	0.3780	0.2405
Emp. (dataset)	10^{-2}	0.1000	0.4025	0.2513
0.1	10^{-1}	0.1067	0.3889	0.2478
0.2	10^{-2}	0.1001	0.3850	0.2426
0.5	10^{-1}	0.0997	0.3813	0.2405
RBF ($\log \gamma = -3$) Kernel - PCA = 5				
Emp. (fold)	10^2	0.0951	0.3934	0.2443
Emp. (dataset)	10^2	0.0949	0.3959	0.2454
0.1	10^2	0.0993	0.3909	0.2451
0.2	10	**0.0926**	0.3929	0.2428
0.5	10	0.0966	**0.3710**	**0.2338**

Table 8: Prior-weighted SVM - Poly and RBF Kernel results

Since the distance between the two best results is higher than in the previous case, and given the similarity between Quadratic Logistic Regression and SVM with Quadratic Kernel, we got the best configuration for SVM.

Kernel	Prior	C*	minDCF ($\tilde{\pi} = 0.5$)	minDCF ($\tilde{\pi} = 0.1$)	min C_{prim}
RBF ($\log \gamma = -3$)	0.5	10	0.0966	0.3710	0.2338

Table 9: SVM - Best configuration (PCA 5)

3.5 GMM: Gaussian Mixture Models

We now can exploit the data distributions we have visualized in the data analysis section.

We will consider models with different numbers of components both for the target and non-target class: we expect that the target class can be described with fewer components than the non-target class because its distribution already resembles a Gaussian curve.

The first step to take is to check the effect of increasing K: we started using only Full Covariance matrices combined with different preprocessing strategies. Due to the subclusters, visible in Figure 1, that constitute the non-target class, we expect that an increasing number of components for it can have beneficial effects; however, we need to pay attention in order to not stumble in overfitting.

Target	Non-Target	minDCF ($\tilde{\pi} = 0.5$)	minDCF ($\tilde{\pi} = 0.1$)	min C_{prim}
GMM - No PCA				
1	2	0.1227	0.4651	0.2939
1	8	0.0941	0.4260	0.2600
1	16	0.0914	0.3976	0.2445
GMM - PCA 5				
1	2	0.1192	0.4318	0.2755
1	8	0.0965	0.4241	0.2603
1	16	**0.0853**	0.4022	0.2438
GMM Z-Norm				
1	2	0.1299	0.4847	0.3073
1	8	0.0919	0.3780	0.2349
1	16	0.0880	**0.3693**	**0.2287**

Table 10: GMM - Full Covariance results

As we expected, increasing the number of non-target components leads to improved performance.

We now want to analyze some combinations of covariance matrix transformations based on distributions of target and non-target classes' data. We will mainly transform the covariance matrices in the subsequent ways:

- F: Full Covariance
- D: Diagonal
- T: Tied
- DT: Diagonal Tied

Since the number of possible combinations of the parameters in this model can be practically unbounded, we decided to stop at K = 32 for the non-target class: going over this threshold could lead, as previously said, to overfitting, other than being computationally expensive. For the target class, as said before, we noticed more regularity: a fewer number of components should be sufficient (we tried up to 4 components). We report only some of the best results we got.

Target	Non-Target	minDCF ($\tilde{\pi} = 0.5$)	minDCF ($\tilde{\pi} = 0.1$)	min C_{prim}
GMM - No PCA				
2 (D)	32 (D)	**0.0838**	0.3370	0.2104
4 (D)	32 (D)	0.0839	0.3406	0.2123
GMM - PCA 5				
2 (T)	32 (T)	0.0896	**0.3305**	**0.2101**
4 (F)	32 (DT)	0.0915	0.3408	0.2161
GMM Z-Norm				
1 (D)	32 (D)	0.0869	0.3365	0.2117

Table 11: GMM - Combinations results

We may argue about which model to select between the NoPCA-Diagonal (T2 - NT32) and the PCA5-Tied (T2 - NT32) since they differ very slightly on C_{prim}, but we decided to carry out our analysis by picking the latter: we want to see if the hypothesis of having the same spread for the subclusters of the non-target class can lead to better results. Plus, up to now, PCA has behaved well and we decide to use it again.

Target	Non-Target	minDCF ($\tilde{\pi} = 0.5$)	minDCF ($\tilde{\pi} = 0.1$)	min C_{prim}
2 (T)	32 (T)	0.0896	0.3305	0.2101

Table 12: GMM - Best configuration (PCA 5)

3.6 Score Calibration & Fusion

To sum up, the models we chose are reported in the following table. (note: from now on, we will refer to the prior-weighted Quadratic Logistic Regression chosen as candidate simply as Quadratic Logistic Regression).

	minDCF ($\tilde{\pi} = 0.5$)	minDCF ($\tilde{\pi} = 0.1$)	min C_{prim}
Q-Log-Reg λ=10, π_T=0.1, No PCA	0.1031	0.3809	0.2420
SVM RBF $\log \gamma = -3$, π_T=0.5, C=10, PCA 5	0.0966	0.3710	0.2338
GMM Tied (T2 - NT32), PCA 5	0.0896	0.3305	0.2101

Table 13: Candidates models

We can compare the performance of these models through a DET plot.

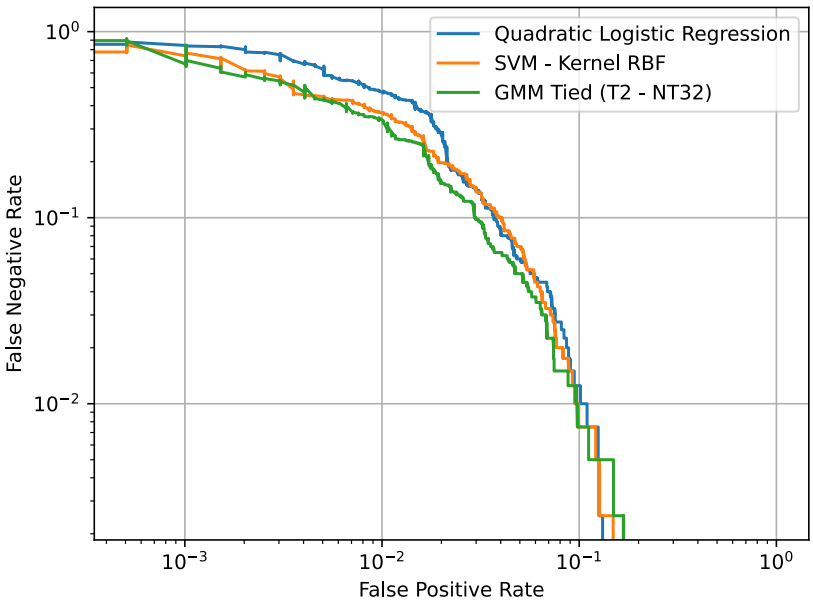

Figure 18: DET - Training

It appears that, for very low values of FPR, SVM is the best model. As we go along with increasing FPR, we get that overall GMM is the best approach, and at the very end, the Quadratic Logistic Regression is the better (so for higher values of FPR). We get how the GMM is the best model on our working points, especially on $\pi = 0.1$, while it's just slightly better on $\pi = 0.5$.
We now move to analyze whether these models are well-calibrated and if their fusion at the score level could lead to additional improvements. We report the actual and minimum costs for our models.

Model	minDCF ($\tilde{\pi} = 0.5$)	minDCF ($\tilde{\pi} = 0.1$)	min C_{prim}	actDCF ($\tilde{\pi} = 0.5$)	actDCF ($\tilde{\pi} = 0.1$)	C_{prim}
Weighted Q-Log-Reg	0.1031	0.3809	0.2420	0.2033	1.0	0.6017
SVM RBF	0.0966	0.3710	0.2338	0.1026	0.8121	0.4574
GMM	0.0896	0.3305	0.2101	0.0955	0.3331	0.2143

Table 14: Candidates models - Minimum and actual costs

The GMM looks well-calibrated for our working points, while SVM and Quadratic Logistic Regression don't seem so. To confirm this fact, we report the Bayes Error Plot for the scores as seen up to now.

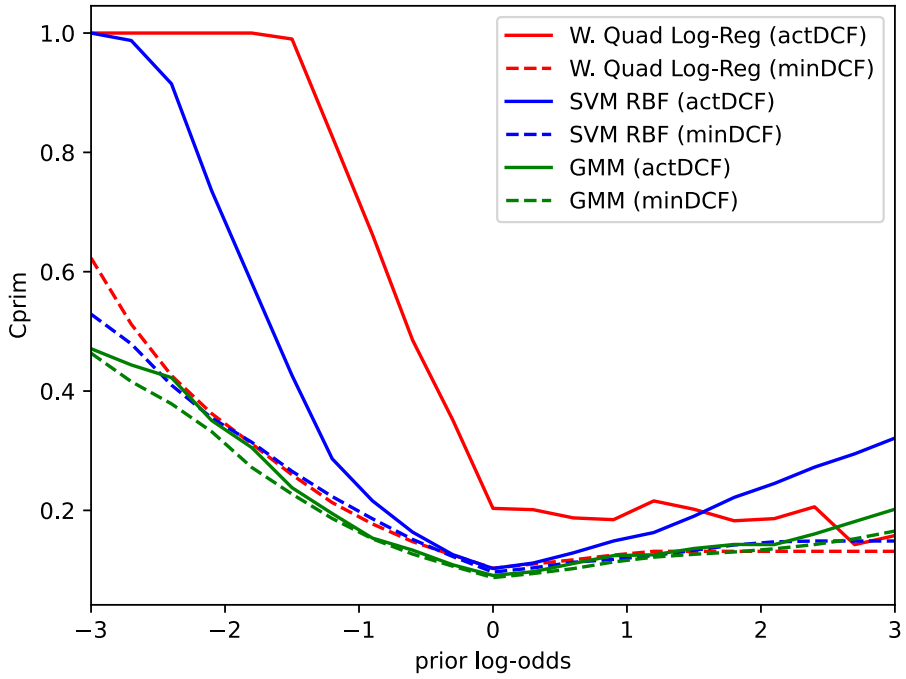

Figure 19: Bayes Error Plot - Uncalibrated scores

It's clear that a score calibration could improve our performance, so we will train a prior-weighted Logistic Regression over the scores to re-calibrate the models. We are looking for optimizing the performance on two working points (and we also considered an intermediate point): so we will try all these priors to train a Logistic Regression over the scores (so we will use as priors 0.1, 0.2 and 0.5). We adopt, again, a K-Fold approach. In the following, we report both the minimum and actual costs for calibrated scores.

Model	Prior	minDCF ($\tilde{\pi} = 0.5$)	minDCF ($\tilde{\pi} = 0.1$)	min C_{prim}	actDCF ($\tilde{\pi} = 0.5$)	actDCF ($\tilde{\pi} = 0.1$)	C_{prim}
Q-Log-Reg	0.1	0.1047	0.3834	0.2441	0.1150	0.3879	0.2515
Q-Log-Reg	0.2	0.1047	0.3834	0.2441	0.1145	0.3904	0.2525
Q-Log-Reg	0.5	0.1047	0.3813	0.2430	0.1160	0.3929	0.2545
SVM RBF	0.1	0.0997	0.3848	0.2423	0.1029	0.4037	0.2533
SVM RBF	0.2	0.0981	0.3848	0.2415	0.1029	0.4037	0.2533
SVM RBF	0.5	0.0976	0.3823	0.2400	0.1044	0.3987	0.2516
GMM	0.1	0.0921	**0.3410**	**0.2166**	0.0949	**0.3489**	0.2219
GMM	0.2	0.0924	**0.3410**	**0.2167**	**0.0944**	**0.3489**	**0.2217**
GMM	0.5	**0.0920**	0.3437	0.2178	**0.0944**	0.3510	0.2227

Table 15: Minimum and actual costs - Calibrated scores

Since there are no substantial differences among the priors, we select $\pi_T = 0.5$ (the one working better for the more uncalibrated models). We report the Bayes Error Plot for the three models after having employed a Logistic Regression score calibration with $\pi_T = 0.5$.

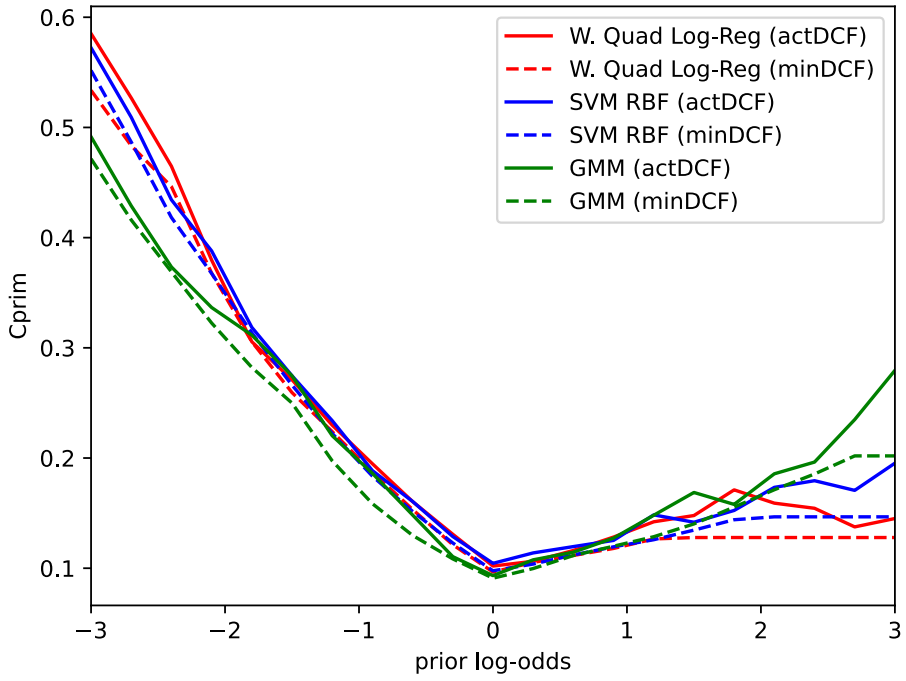

Figure 20: Bayes Error Plot - Log-Reg calibrated ($\pi_T = 0.5$)

We can notice still some miscalibration in the right part of the plot, so for high priors: however, our target working points have taken advantage of the calibration process. We notice how, in particular, SVM and Quadratic Logistic Regression have reduced drastically the gap between actual and minimum costs.

Finally, we consider the fusion of the models employing again a Logistic Regression with $\pi_T = 0.5$.

Model	minDCF ($\tilde{\pi} = 0.5$)	minDCF ($\tilde{\pi} = 0.1$)	min C_{prim}	actDCF ($\tilde{\pi} = 0.5$)	actDCF ($\tilde{\pi} = 0.1$)	C_{prim}
[1] W. Q-Log-Reg	0.1047	0.3813	0.2430	0.1160	0.3929	0.2545
[2] SVM RBF	0.0976	0.3823	0.2400	0.1044	0.3987	0.2516
[3] GMM	0.0920	0.3437	**0.2178**	0.0944	0.3510	0.2227
[1] + [2]	0.0991	0.3797	0.2394	0.1054	0.3987	0.2521
[1] + [3]	0.0893	0.3405	**0.2149**	0.0969	0.3622	**0.2296**
[2] + [3]	0.0893	0.3542	0.2218	0.0934	0.3738	0.2336
[1] + [2] + [3]	0.0925	0.3530	0.2228	0.0964	0.3784	0.2374

Table 16: Minimum and actual costs - Calibrated scores (Fusion)

The fusion approach apports some improvements. The best fusion is [1]+[3], the one which combines the Quadratic Logistic Regression with the GMM, but the performance is similar to that obtained with only the GMM.

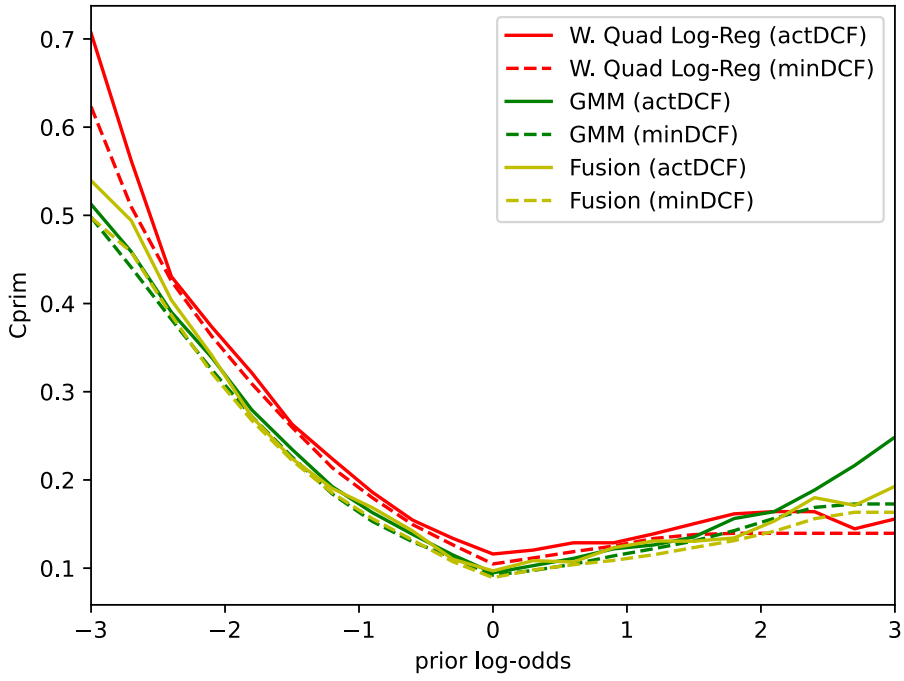

Figure 21: Bayes Error Plot - Fusion [1] + [3]

In conclusion, this fusion leads to better performance, if compared with Quadratic Logistic Regression or SVM. Furthermore, we can see that the fusion is calibrated in our range of interest, while it shows some miscalibration for increasing priors.

4 Evaluation

We move now to analyze the performances of the best models and their fusions on the evaluation set. We compare the results we previously got on the training set with the ones we got after simply evaluating, as said, on the test set.

	Validation Set		Evaluation Set	
Model	min C_{prim}	C_{prim}	min C_{prim}	C_{prim}
[1] Weighted Q-Log-Reg	0.2430	0.2545	0.2646	0.6147
[2] SVM RBF	0.2400	0.2516	0.2495	0.4388
[3] GMM	0.2178	**0.2227**	0.2441	0.2497
[1] + [2]	0.2394	0.2521	0.2506	0.2563
[1] + [3]	**0.2149**	0.2296	0.2407	0.2441
[2] + [3]	0.2218	0.2336	**0.2362**	**0.2425**
[1] + [2] + [3]	0.2228	0.2374	0.2375	0.2431

Table 17: Minimum and actual costs - Evaluation set

The best model seems to be [2]+[3]: this is, indeed, the best fusion since it does not involve the Quadratic Logistic Regression (which shows higher miscalibration than the other models). We notice a confirmation in the relative performance of the pure models (so GMM is still the best pure model).

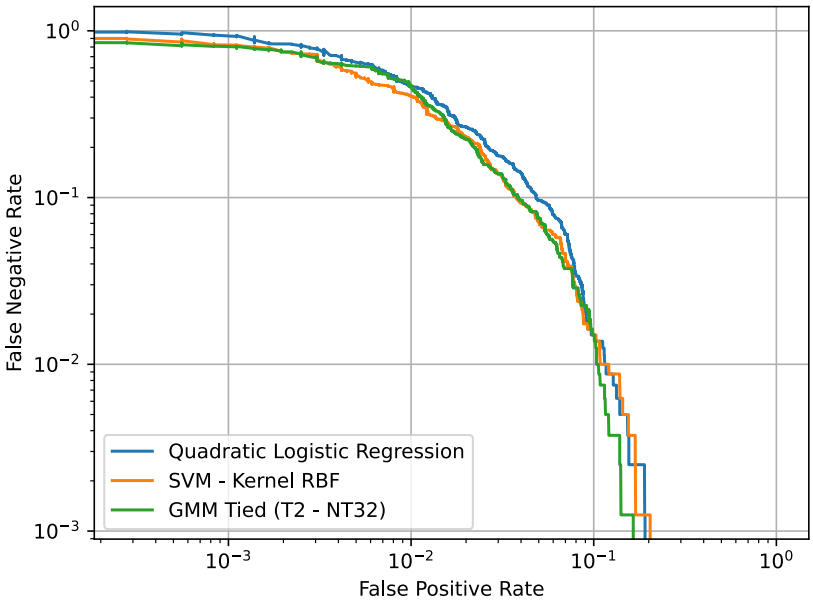

Figure 22: DET - Evaluation

The DET plot confirms that, overall, GMM is the best model (even if for some ranges the SVM performs better). Plus, the plot confirms that the Quadratic Logistic Regression is the one performing worse. This confirms more or less what we saw in the training phase.

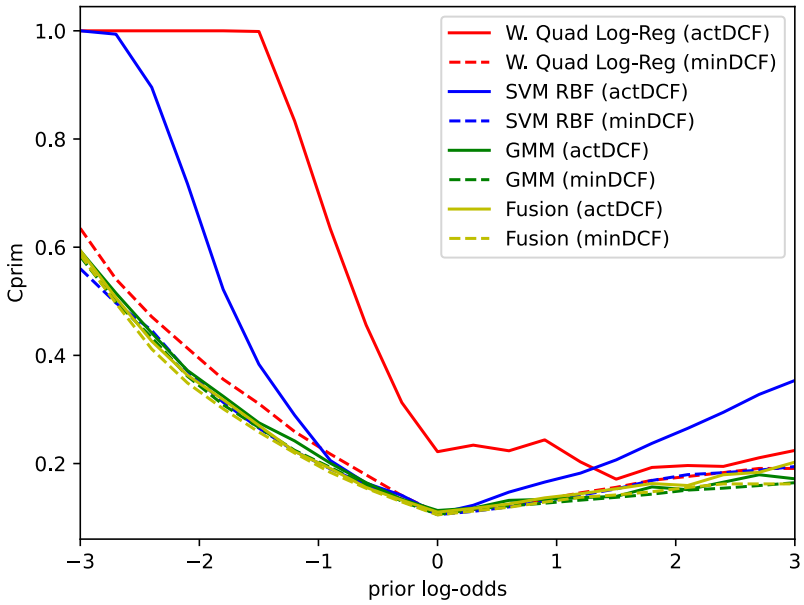

Figure 23: Bayes Error Plot - Evaluation

The Bayes Error confirms what we have stated: the best fusion ([2]+[3], which is the one shown in the plot) is the model which performs best on the evaluation set, even though its performance is almost identical to the GMM one. Quadratic Logistic Regression and SVM show miscalibration.

We now can analyze our choices: we selected three candidate models, with some hyperparameters configuration, based on their performance on the training set. We will conduct our analysis by comparing the minimum costs obtained on the training set and on the evaluation set.

We start with the Quadratic Logistic Regression models: we conduct brief research on what could have been a better value for λ. We report the graphs for both training and evaluation sets, without PCA.

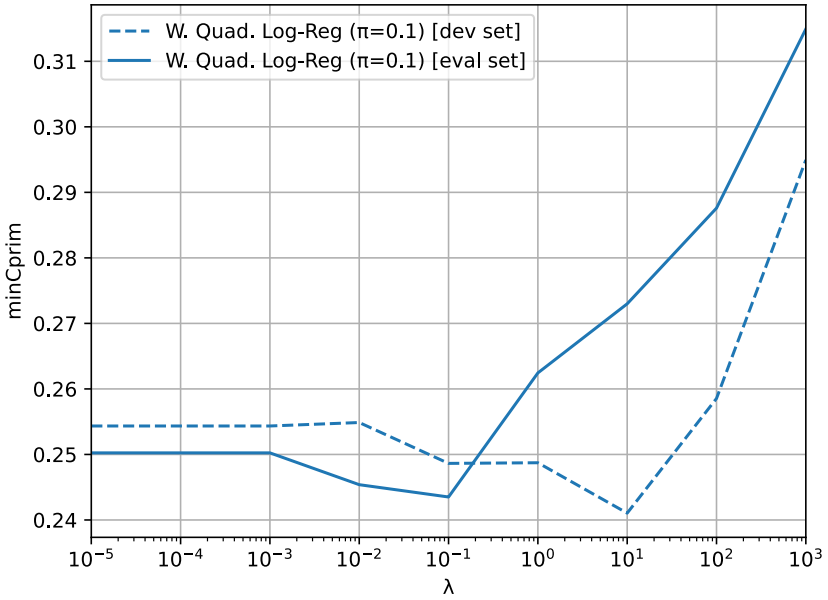

Figure 24: Q-Log-Reg (No PCA, $\pi_T = 0.1$)

As we can see, the best value for the evaluation set would have been $\lambda = 10^{-1}$, instead of 10. We now consider also different priors: we show the main results in the table below.

Prior	λ^*_{DEV}	min C_{prim}	λ^*_{EVAL}	min C_{prim}
Emp. (dataset)	10	0.2724	10^{-1}	**0.2429**
0,1	10	**0.2718**	10^{-1}	0.2450
0,2	10	0.2728	10^{-1}	0.2435
0,5	10	0.2761	10^{-1}	0.2473

Table 18: Q-Log-Reg (No PCA) - Evaluation costs

As we can see, the best choice would have been using the dataset empirical prior with $\lambda = 10^{-1}$. There is some difference between the two configurations, so in this case, our choice was sub-optimal.

We can also analyze whether PCA would have improved the performance. Consider so the best configuration (empirical prior and $\lambda = 10^{-1}$). We reported minimum costs for PCA and also for the combination of PCA and Z Normalization.

PCA	PCA - min C_{prim}	PCA + Z-Norm - min C_{prim}
-	0.2429	0.3227
5	0.2452	0.3280
4	0.3049	0.3658

Table 19: Q-Log-Reg - Minimum costs (Evaluation set)

Our intuition was correct: using PCA would have led to sub-optimal choices, especially if combined with Z Normalization.
The optimal configuration we found is $\lambda = 10^{-1}$ using the dataset empirical prior.

We now consider the SVM RBF: starting from our configuration, we consider also other possible values of γ using PCA 5 and $\pi_T = 0.5$.

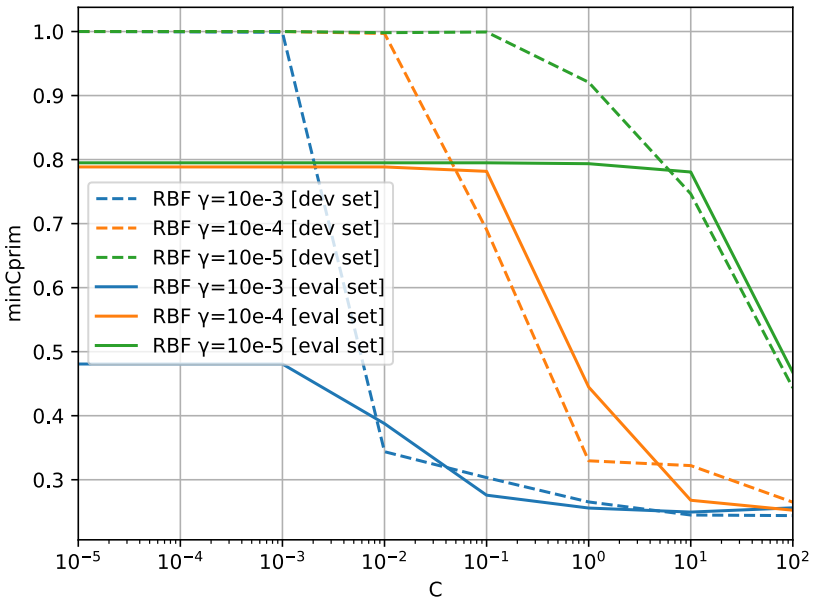

Figure 25: SVM RBF (PCA 5, $\pi_T = 0.5$) - Dev. vs Eval. costs

As we can see, here the situation looks better: for the chosen configuration ($C = 10$, $\gamma = 10^{-3}$), we notice that the gap between minimum cost on the training set and minimum cost on evaluation set is almost unnoticeable.
We investigated more and decided also to conduct a brief research for other possible values of the prior: we reported the best results we got in the table below.

Type	min C_{prim} (Dev)	min C_{prim} (Eval)	min C_{prim} (Eval - Z-Norm)
PCA-5, $\log \gamma = -3$, $C = 10$, $\pi_T = 0.5$	**0.2338**	0.2495	0.7757
PCA-5, $\log \gamma = -3$, $C = 10^2$, $\pi_T = 0.5$	0.2461	0.2561	0.3523
PCA-5, $\log \gamma = -4$, $C = 10^2$, $\pi_T = 0.5$	0.2606	0.2564	0.7876
PCA-5, $\log \gamma = -3$, $C = 10$, $\pi_T = 0.1$	0.2480	**0.2411**	0.3887

Table 20: SVM RBF - Minimum costs (Evaluation set)

We also reported the evaluation for the combination of PCA 5 and Z Normalization and, once again, we can see that the Z Normalization isn't effective. Our choice, even if sub-optimal, leads still to good results: the optimal choice, however, would have been using the same as the one chosen, but with $\pi_T = 0.1$.

Finally, we treat what we have done with GMM. As we said during the training phase, the number of possible combinations for GMM can be extremely high. We reported in the table below some of the best configurations we discovered on the evaluation set, plus our chosen model.

	PCA	min C_{prim} (Dev)	min C_{prim} (Eval)
2 (D) - 32 (D)	-	0.2104	**0.2375**
4 (D) - 32 (D)	-	0.2122	0.2376
1 (FC) - 8 (FC)	-	0.2600	0.2602
4 (FC) - 32 (DT)	PCA-5	0.2161	0.2773
1 (FC) - 32 (DT)	PCA-5	0.2152	0.2398
Chosen Configuration			
2 (T) - 32 (T)	PCA-5	0.2101	0.2441

Table 21: GMM - Minimum costs (Evaluation set)

Even though our model was indeed the best on the training set, we got slightly worse results on the evaluation set. We should have accepted imperceptible worse performance in order to perform better on the evaluation set. Maybe our hypothesis about the common spread over the clusters wasn't entirely correct: however, we still got an acceptable result. The best choice for the GMM would have been the first reported in the table: so Diagonal(T2 - NT32) without PCA.

To sum up, we report for each chosen configuration the respective found optimal one.

Model	Chosen conf.	Optimal conf.
[1] Weighted Q-Log-Reg	$\pi_T = 0.1$, $\lambda = 10$, No PCA	$\pi_T = empirical$, $\lambda = 10^{-1}$, No PCA
[2] SVM RBF	$\pi_T = 0.5$, $C = 10$, $\log \gamma = -3$, PCA 5	$\pi_T = 0.1$, $C = 10$, $\log \gamma = -3$, PCA 5
[3] GMM	Tied(T2 - NT32), PCA 5	Diagonal(T2 - NT32), No PCA

Table 22: Chosen and optimal configurations

In the following summarising table, we show the actual and the minimum costs for the models, both for our choices and for optimal configurations and considering also the fusions.

	Chosen conf.	Optimal conf.
Model	min C_{prim}	min C_{prim}
[1] Weighted Q-Log-Reg	0.2646	0.2429
[2] SVM RBF	0.2495	0.2441
[3] GMM	0.2441	0.2375
[1] + [2]	0.2506	0.2443
[1] + [3]	0.2407	**0.2318**
[2] + [3]	**0.2362**	0.2325
[1] + [2] + [3]	0.2375	0.2364

Table 23: Minimum cost comparison between chosen and optimal configurations - Calibrated scores

The fusion is slightly effective: on the evaluation set, the one performing better is [1] + [3], but these are virtually not relevant.

In the end, our approach led to close-to-optimal models: some adjustments have been needed due to some mismatches between training and evaluation data. The higher cost difference from one of our configurations and the optimal is for the Quadratic Logistic Regression, but it is still good since it is under around 0.023. The fusions have been proven to be useful but with limited benefits.