Italian Language Detection from Utterance Embeddings: A Comparative Study of SVM, Gaussian Models, Logistic Regression, and GMM

Di Fede, Giulia
307746

Dardanello, Leonardo
319060

Abstract

In this work, we present an investigation into the behaviour and performance of some the most common Machine Learning algorithms. In particular, the classifiers will be trained on a dataset consisting of small-dimensional utterance embeddings extracted from audio sources. To have a better and more comprehensive view of the data, our study will set off from data exploration, followed by the analysis of the behaviour of different algorithms in a validation phase. We will also go through the process of score calibration to analyse the goodness of our decisions. Then we will employ Fusion to see how our models would perform together. To complete our study then our models will go through an evaluation phase to select the optimal configuration for our classification task.

Contents

1 Introduction 1

2 Features Analysis 2
- 2.1 Scatter Plots and Histograms 2
- 2.2 Correlation Matrices 2

3 Dimensionality Reduction 3
- 3.1 Principal Component Analysis 3
- 3.2 Linear Discriminant Analysis 3

4 Classification 4
- 4.1 Gaussian Classifiers 4
- 4.2 Logistic Regression 5
- 4.3 Support Vector Machines . . 7
- 4.4 Gaussian Mixture Models . . 11

5 Scores calibration and Fusion 12
- 5.1 Scores calibration 13
- 5.2 Fusion 14

6 Evaluation Phase 15
- 6.1 Gaussian Classifiers 16
- 6.2 Quadratic Logistic Regression 16
- 6.3 Support Vector Machines . . 16
- 6.4 Gaussian Mixture Models . . 18
- 6.5 Fusion Models 18

7 Conclusions 19

1 Introduction

In our increasingly interconnected world, natural language processing and identification has become a crucial task with a growing range of applications and domains, including communication technologies, translation services, voice assistants and social media analysis to name a few.

While the field of speech recognition focuses primarily on converting spoken language into written text, it often lacks of ways to first correctly identify the spoken language which has to be eventually selected by the user. Thus the task of language identification can play an important role in enabling an effective and automatic selection of context, which is often important in Natural Language Processing (NLP) tasks.

In this study we focus mainly on a more specific task, which is the testing of Support Vector Machines, Gaussian Models, Logistic Regression, and Gaussian Mixture Models classifiers on the identification of the target Italian language among a given set of 26 languages. The non-target language samples belong to one of 25 possible languages, but the actual language label is not available, therefore, this study addresses a binary classification problem.

The dataset is composed of synthetic language embeddings representing utterances in 26 different languages, every embedding is represented by a 6-dimensional continuous-valued vector belonging either to the target language (label 1) or to the non-target languages (label 0). The embedding components do not have a physical interpretation.

One of the challenges imposed by the dataset is given by the unbalanced nature of the samples. The training set in fact contains 400 samples from the target class and 1971 samples from the non-target class. The evaluation set instead contains more data, more specifically it is made of 800 samples belonging to the target class and 3603 samples belonging to the non-target class.

We want to focus our analysis on two working points $(0.5, 1, 1)$ and $(0.1, 1, 1)$, therefore our objective is to build a classification system that provides good results for both applications at the same time, and our primary metric will consist on the average of the $minDCF$ of the two working points $minC_{prim}$.

2 Features Analysis

2.1 Scatter Plots and Histograms

To start analyzing our features we consider the scatter plots shown in Figure 30.

It is noticeable from the scatter plots of our data that our classes seem well described by Gaussian densities and we can hypothesize that the non-target class is composed of different sub-classes of similar distributions, each one representing one of the 26 non-target languages. This suggests us that models like the Guassian Mixture Models could produce good results as well as non-linear versions of the classifiers we will compare. However the scatter-plots also suggest how the data is not linearly separable, thus we expect non-linear models to produce a better performance than linear models.

The histograms in Figure 30 (obtained with a reduction to 2 dimensions with the employment of Principal Component Analysis) show how the data is not easily separable. But it is noticeable almost in all features how the non-target class is characterized by a larger variance compared to the target class. This could be given by the fact that the non-target class contains samples from many other different languages and therefore it is natural to expect a wider variability. We can also see the presence of a few outliers, although they are not too relevant.

It is also important to notice how the mean values of the Italian samples are close to zero, but this characteristic is not exclusive to the target samples. The non-target samples also exhibit similar mean values.

2.2 Correlation Matrices

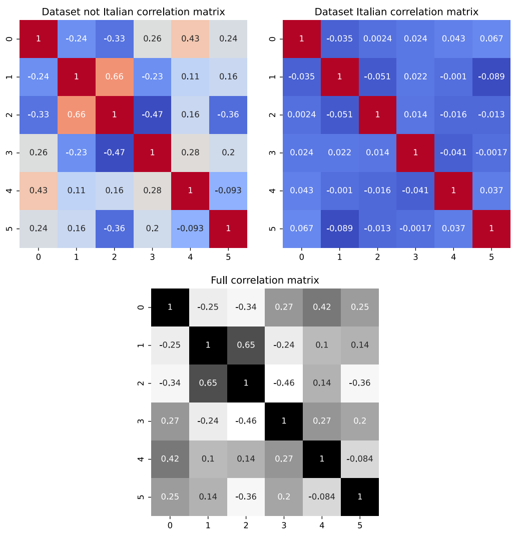


Figure 1: Correlation matrices of the given features.

From the correlation matrix shown in Figure 1, we can observe that some features exhibit a certain degree of correlation in the case of the non-target class.

For example, feature 1 and feature 2 are described by a positive correlation, as feature 4 and feature 0 do. While feature 2 and feature 3 instead share a negative correlation, similarly to feature 2 and feature 5.

Instead, it is evident that there are no significant correlations between the features of the target class, in fact the values shown in the correlation matrix are close to zero.

We note that we might want to consider the employment of techniques of dimensionality reduction because of the nature of our dataset. In fact our dataset is characterized by few samples on the target class. Dimensionality reduction could reduce the effects of the curse of dimensionality, the reason of this is given by the fact that we will have less dimensions to estimate with more data. However as we have seen from the Correlation Matrices the non-target class holds some degree of correlation, therefore a drastic reduction of the dimensionality of our dataset might cause a loss of this information and therefore a poorer performance for some models as the MVG model.

3 Dimensionality Reduction

3.1 Principal Component Analysis

Principal Component Analysis is an unsupervised dimensionality reduction technique that consists in the minimization of the average reconstruction error which translates into finding the directions of maximization of the variance.

We now employ a Principal Component Analysis to better understand by how many dimensions we can reduce our data while keeping an acceptable amount of information. In order to better quantify the amount of information that each feature convey we decided to take into consideration as a metric the fraction of explained variance.

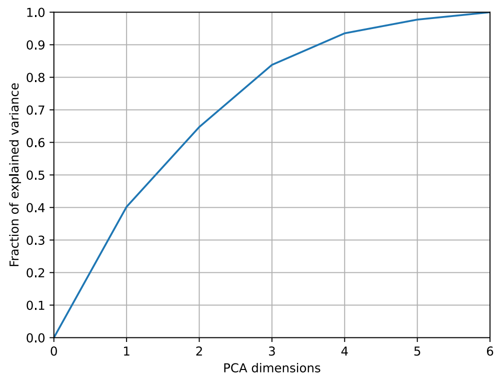


Figure 2: Graph reporting the fraction of explained variance.

From this analysis we can observe that a dimensionality reduction down to four dimensions lets us maintain an acceptable amount of information. Following these result our study will focus on data with a minimum of four dimensions.

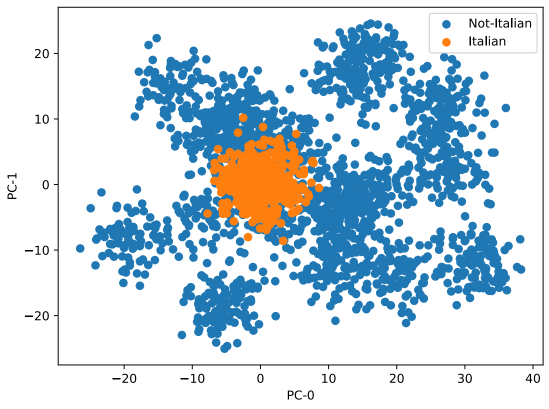


Figure 3: Scatter plot of the data reduced by means of PCA

From the scatter plot in Figure 3 obtained with the use of PCA dimensionality reduction down to 2 dimensions we can see in more detail how the non-target class is characterized by sub-clusters.

3.2 Linear Discriminant Analysis

While PCA doesn't take into consideration the labels of our features, Linear Discriminant Analysis follows a supervised approach working on finding the direction of largest separation between the means of the classes while trying to obtain the smallest spread of the samples that belong to each class. Because of the nature of the algorithm we can only find $C-1$ discriminant directions, with C being the number of classes. In our case $C = 2$ which means that we can find only one discriminant direction.

Although we have already observed in the features analysis that the distribution of the features doesn't suggest a linear separation rule due the to the presence of overlapping portion in the histograms we try to employ a Linear Discriminant Analysis to find the most discriminant direction.

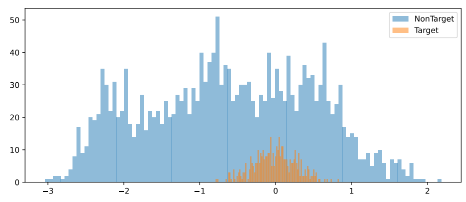


Figure 4: Histogram resulted by the application of Linear Discriminant Analysis.

As we expected the LDA approach can't find a direction in which the data might be linearly separable. Again, given the distribution of the features we expect non-Gaussian models and non-linear models to perform significantly better.

4 Classification

We will now begin our analysis of the performance of different classifiers, for our training method we will apply a K-fold cross-validation approach with $K = 5$. The analysis is expressed by meaning of the minimum Detection Cost Function, a function that measures the cost of our decisions in terms of the prior probability and risks defined in the working points. We will report the result for both working points, $(0.5, 1, 1)$ and $(0.1, 1, 1)$, and in terms of $minCprim$ expressed as the average of the $minDCF$ between the working points.

4.1 Gaussian Classifiers

Our analysis begins with Gaussian classifiers, specifically considering four different variations: MVG Classifier, the MVG classifier with Naive Bayes assumption and, despite being linear classifiers, we also consider the MVG classifiers with tied covariance and the MVG classifier with tied covariance and Naive assumption.

All of the MVG classifiers assume that the data of each class can be modelled by a Multivariate Gaussian distribution, if our data is D-dimensional:

$$\boldsymbol{X}|C = c \sim \mathcal{N}(\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c) \tag{1}$$

$$f(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})} \tag{2}$$

Specifically, the Naive assumption simplifies the estimation assuming that the different components are independent from each other. This simplification brings us to obtain a diagonal covariance matrix $\boldsymbol{\Sigma}_c$ for each class. In this case we lose the information given to us by the correlations which are present in our data, specifically in the non-target class. However we will have less parameters to estimate as we can avoid the estimation of the off-diagonal elements of $\boldsymbol{\Sigma}_c$.

Instead, the MVG classifier with tied covariance assumes that the covariance matrix $\boldsymbol{\Sigma}$ is the same for all classes, starting from the assumption that our data is composed by its mean $\boldsymbol{\mu}_c$ and gaussian distributed noise $\boldsymbol{\epsilon}_i$.

$$\boldsymbol{x}_{c,i} = \boldsymbol{\mu}_c + \boldsymbol{\epsilon}_i \tag{3}$$

Tied covariance models can therefore capture correlations, but as our classes have different distributions we expect it to perform poorly. Furthermore, with the tied covariance model we obtain a linear classification rule which is likely to yield unfavourable outcomes.

However, although the Tied assumptions seem quite inaccurate, Tied and Naive Bayes models may benefit from the lower risk of overfitting and high-dimensionality, so we consider a brief analysis even if we consider the MVG classifier to demonstrate a better performance.

PCA	minDCF($\tilde{\pi}$=0.1)	minDCF($\tilde{\pi}$=0.5)	min C_{prim}
-	0.502	0.131	0.316
5	0.494	0.130	0.312
4	0.543	0.134	0.338

Table 1: MVG classifier minCDF and minCprim (K-Fold=5)

As we expected, PCA does improve the classification for the MVG model. PCA seems slightly effective only by reducing the features dimensionality by one dimension. But this does not bring too much improvement. A reason for this might be the loss of the information about the correlation of our data, however this is compensated by the reduction of dimensionality which brings down the risks posed by high-dimensionality datasets.

PCA	minDCF($\tilde{\pi}$=0.1)	minDCF($\tilde{\pi}$=0.5)	min C_{prim}
-	0.546	0.138	0.342
5	0.475	0.130	0.302
4	0.537	0.131	0.334

Table 2: MVG Naive classifier minCDF and minCprim (K-Fold=5)

As we can see from Table 1 and Table 2 the Naive model performs slightly better than the MVG classifier by applying PCA and reducing the features dimensionality by one dimension. This is probably due to the hypotheses spoken above. In fact we can see that dimensionality reduction does bring visible improvements to our classification.

PCA	minDCF($\tilde{\pi}$=0.1)	minDCF($\tilde{\pi}$=0.5)	min C_{prim}
-	1	0.515	0.757
5	1	0.510	0.755
4	1	0.483	0.741

Table 3: MVG Tied classifier minCDF and minCprim (K-Fold=5)

As we anticipated, the linear model cannot separate the classes and thus has significant worse performance than the MVG classifier, despite prior considerations.

PCA	minDCF($\tilde{\pi}$=0.1)	minDCF($\tilde{\pi}$=0.5)	min C_{prim}
-	1	0.491	0.745
5	1	0.510	0.755
4	1	0.482	0.741

Table 4: MVG Tied Naive classifier minCDF and minCprim (K-Fold=5)

It can be seen from Table 4 how the performance obtained from the MVG classifier with tied covariance and Naive assumption is poor, as we are still considering linear classification rules.

In Table 5 we analyse the effects of z-normalization on the gaussian classifier. We see that this makes our best model perform worse.

PCA	minDCF($\tilde{\pi}$=0.1)	minDCF($\tilde{\pi}$=0.5)	min C_{prim}
-	0.546	0.137	0.342
5	0.563	0.142	0.352
4	0.592	0.152	0.372

Table 5: Best Gaussian classifier (Naive) minCDF and minCprim tested for z-normalization (K-Fold=5)

We conclude that the best model between the Gaussian Classifiers is the Naive MVG classifier with PCA 5. Even if the MVG classifier takes into account the correlations of our dataset, the naive bayes assumption with the help of PCA does simplify the estimation for the reasons reported above.

4.2 Logistic Regression

We now try discriminative classifiers, in particular we will try Prior-Weighted Linear Logistic Regression Models and Prior-Weighted Quadratic Logistic Regression Models. Unlike the Gaussian models these classifiers focus on the direct estimation of the posterior probability of a given sample for a given class.

Considering the definition of the sigmoid function, we define the score for each class as described below.

$$\sigma(s) = \frac{1}{1+e^{-s}} \tag{4}$$

$$P(x_i|\mathcal{H}) = \sigma(\omega^T x_i + b) \tag{5}$$

In order to estimate the model parameters (ω, b) we need to go through the minimization of the objective function:

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

Where the hyper-parameter λ in the regularization term $\frac{\lambda}{2}||\omega||^2$ can help us choose a trade off between over-fitting and a simpler solution, in particular:

1. for values of $\lambda \gg 0$ we obtain a poor class separation but a simpler solutions with a smaller norm of ω.

2. for values of $\lambda \simeq 0$ we may have a high degree of over-fitting but good class separation.

For the Quadratic Logistic Regression we will use the same objective function but we will need to expand the data features using the following transformation:

$$\Phi = \begin{bmatrix} vec(xx^T) \\ x \end{bmatrix} \tag{6}$$

Our first analysis will focus on the models with π_T equal to the Empirical Prior of the traning set, that for our dataset corresponds to $\pi_T \simeq= 0.17$ to observe the model performance without Prior-Weighting.

We start by considering the Linear Logistic Regression although we expect that will perform poorly due to the data distribution.We train our model with and without Z-normalization to observe the effect of the normalization on the regularization term.

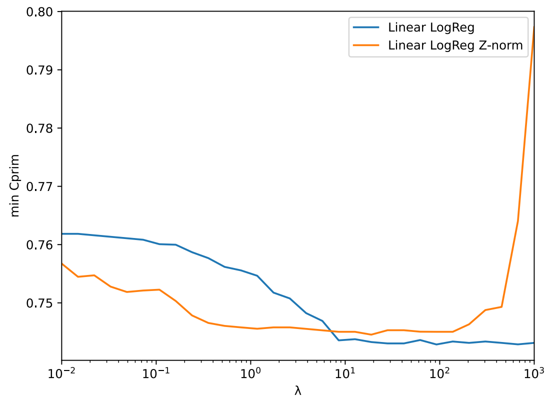


Figure 5: Linear Logistic Regression Classifier with and without Z-normalization, using the empirical prior.

It can be observed that the Z-norm does contribute to the obtainment of a lower $minCprim$ for values of $\lambda < 10^1$. However the best value is obtained with no normalization.

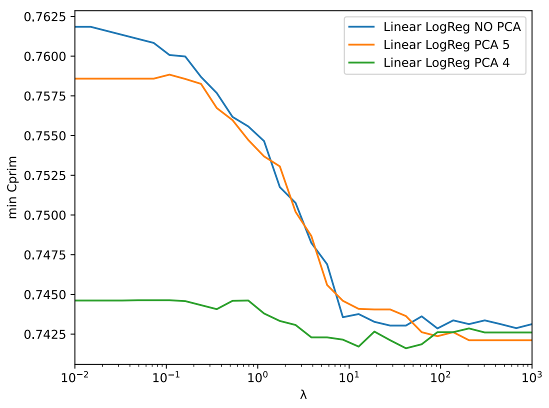


Figure 6: Linear Logistic Regression Classifier with the application of PCA.

Dimensionality reduction via PCA slightly improves the outcome but the model still performs poorly, so we conclude our analysis on the linear version of Logistic Regression and shift our focus towards the Quadratic Logistic Regression that we expect to perform much better.

As we did with the Linear Logistic Regression we start considering the Z-normalization.

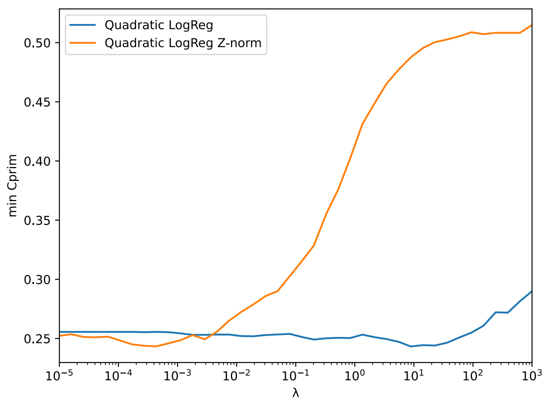


Figure 7: Quadratic Logistic Regression Classifier with and without Z-normalization, using the empirical prior.

For Quadratic Logistic Regression Z-normalization is still giving a good contribution for values of $\lambda < 2 \times 10^{-3}$, however we reach the best values of $minCprim$ without the employment of Z-normalization.

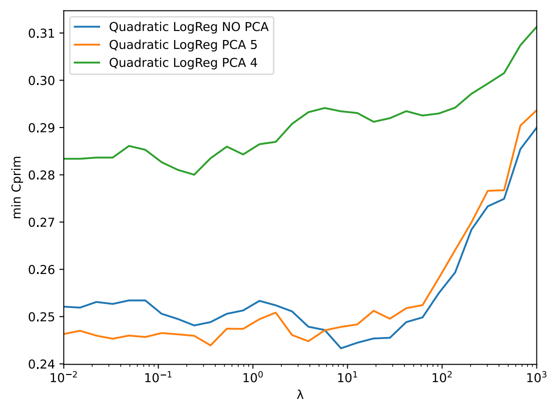


Figure 8: Quadratic Logistic Regression Classifier with the application of PCA.

For values of $\lambda < 6 \times 10^0$ we observe that the model obtained with the use of 5-dimensional PCA seems to perform better but, as mentioned earlier, a value of $\lambda \simeq 0$ can lead to a higher amount of over-fitting. Meanwhile the model obtained without any application of PCA performs slightly better for values of $\lambda \simeq 10$ and would probably yield less overfitting, so we chose this configuration for our next analysis.

To further analyze the effects of Z-normalization, we decide to check if it is able to provide a better outcome by using the best two configuration in terms of PCA dimensions.

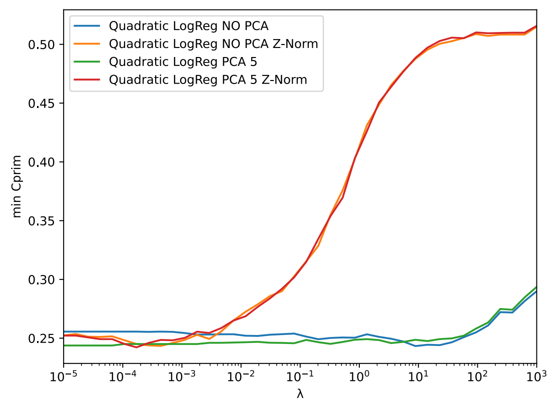


Figure 9: Quadratic Logistic Regression Classifier with the application of PCA = 5 and Normalization.

Which brings us to the conclusion that Z-normalization does bring to some improvements, however our best model is given by the QLR model with $\lambda \simeq 10$.

Therefore we decide to focus on the Quadratic Logistic Regression Model with $\lambda \simeq 10$ without any application of PCA.

Our second analysis of the models will focus on changing the Prior-Weighting π_T to better fit our main goal of evaluating the data in two different working points. So we consider values of $\pi_T = [0.1, 0.5]$, in particular we are interested in testing $\pi_T = 0.1$ and $\pi_T = 0.5$ due to our working points but also $\pi_T = 0.2$ due to our Empirical Prior being close to 0.2. For this analysis we won't consider anymore the Linear Logistic Regression Model given its poor precedent results.

π_T	minDCF($\tilde{\pi}$=0.1)	minDCF($\tilde{\pi}$=0.5)	min C_{prim}
π_T^{emp}(0.17)	0.386	0.104	0.245
0.1	0.380	0.103	0.242
0.2	0.383	0.102	0.243
0.5	0.384	0.102	0.243

Table 6: Prior-Weighting of the Quadratic Logistic Regression Classifier.

From Table 5 we observe that Prior-Weighting with different priors doesn't changes significantly the performance of the model. We chose as our best configuration for the Quadratic Logistic Regression the model with: NO PCA,$\pi_T = 0.1$ and $\lambda \simeq 10$.

4.3 Support Vector Machines

We now consider the Support Vector Machines, a classifier that allows us to obtain scores that have a geometrical interpretation, which however makes them not directly related to class posteriors.

Considering the Primal form of the objective function with soft margin shown below, our results depend on the value of the regularization term C.

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \frac{1}{2} \|\boldsymbol{w}\|^2 + C \sum_{i=1}^{n} \xi_i \qquad (7)$$

For different values of C:

1. Higher values of C will penalize solutions in which we accept errors or points in the margin.

2. Lower values of C will let us have a larger margin and achieve more generalization, but have more errors.

To solve the SVM problem we consider its Lagrangian Formulation, from which we obtain the dual SVM problem. The dual objective function can be expressed in matrix form as:

$$L_D(\boldsymbol{\alpha}) = \boldsymbol{\alpha}^T \mathbf{1} - \frac{1}{2}\boldsymbol{\alpha}^T \boldsymbol{H}\boldsymbol{\alpha}$$

$$s.t\ 0 \leq \alpha_i \leq C, \forall i \in \{1, \dots, n\}, \sum_{i=1}^{n} \alpha_i z_i = 0$$

Such formulation depends on the matrix $\boldsymbol{H}$ whose elements can be defined as:

$$\boldsymbol{H}_{ij} = z_i z_j \boldsymbol{x}_i^{\boldsymbol{T}} \boldsymbol{x}_j \tag{8}$$

Since the dual formulation consists of both box constraints and an additional constraint $\sum_{i=1}^{n} \alpha_i z_i = 0$, the L-BFGS algorithm is unable to incorporate the latter constraint. Therefore we need to slightly modify the SVM problem as to make the constraint disappear.

Using the transformation below

$$\Phi = \begin{bmatrix} \boldsymbol{x}_i \\ K \end{bmatrix} \tag{9}$$

With $K = 1$, the matrix $\boldsymbol{H}$ is transformed into:

$$\boldsymbol{H}_{i,j} = z_i z_j (\boldsymbol{x}_i^T \boldsymbol{x}_j + 1) \tag{10}$$

Given the results obtained with the use of the Quadratic Logistic Regression, we will also try to analyse the non-linear formulations of the SVM models. In such models, non-linearity is easily obtained without doing any features expansion, as the dual formulation depends on the data only through the dot-products in matrix H. For this reason a non-linear transformation only requires dot-products in the expanded space.

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \Phi(\boldsymbol{x}_i)^T \Phi(\boldsymbol{x}_j) \tag{11}$$

Our *Kernel Functions* efficiently compute the dot-products in the expanded space.

$$\boldsymbol{H}_{i,j} = z_i z_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) \tag{12}$$

In particular we will test the following kernel functions:

1. 2-Polynomial Kernel $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\boldsymbol{x}_i^T \boldsymbol{x}_j + c)^d$. where d is the degree of the polynomial function. We note here that higher values of d might cause problems of over-fitting.
2. Radial Basis Function (RBF) Kernel $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = e^{-\gamma ||\boldsymbol{x}_i - \boldsymbol{x}_j||}$. where γ is a hyperparameter.

Specifically for the RBF kernel we note the behaviours:

1. If γ is large (Narrow Kernel), each Support Vector has a very small influence on points that are far from it. So for example if $\gamma >> 0$ the influence will be dominated by the closest Support Vector.

2. If γ is small (Wide Kernel), then each Support Vector will influence points that are much further away.

And in order to add a regularized bias in the non-linear SVM, it is sufficient to add a constant value $\xi = K^2$ to the kernel function:

$$\hat{k}(\boldsymbol{x}_i, \boldsymbol{x}_j) = k(\boldsymbol{x}_i, \boldsymbol{x}_j) + \xi \tag{13}$$

We start our analysis of the Support Vector Machines Models with the Linear classification rule.

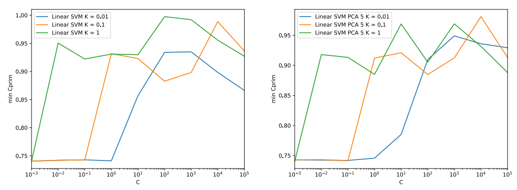


Figure 10: SVM Linear Classifier minCprim, varying values of K (K-Fold=5)

By examining the graphs in Figure 10, it becomes evident that it is difficult to identify the optimal value for K, while, the best values for C would fall in the interval $C = [10^{-2}, 10^{0}]$. However, it is clear that the performance of the Linear SVM model is consistently poor, therefore we turn our attention to the non-linear models.

Starting with the 2-Polynomial kernel, we obtain the following results.

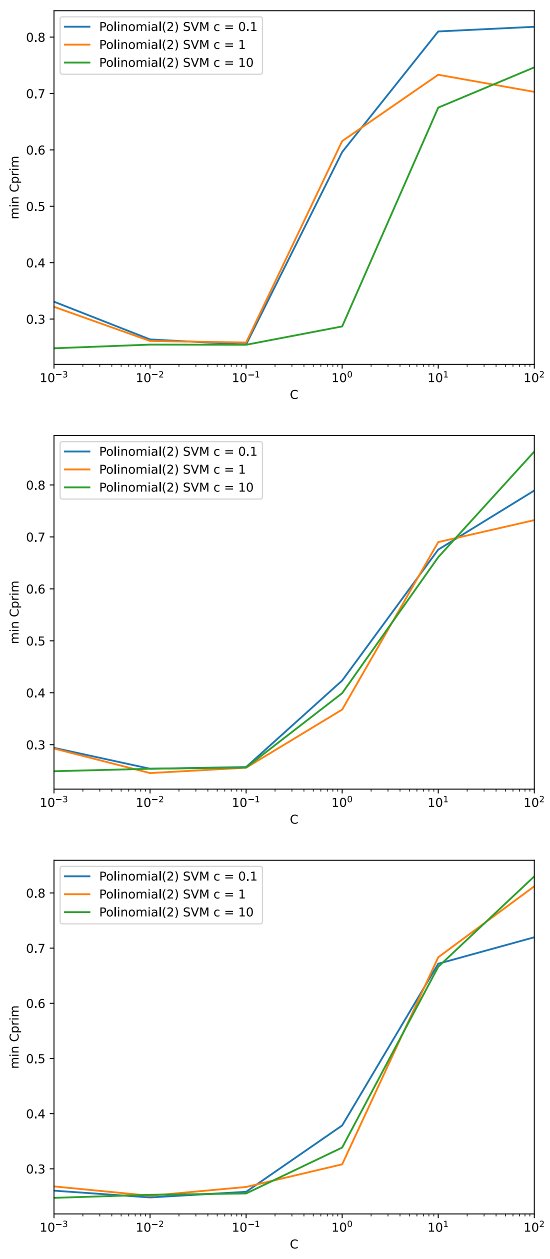


Figure 11: SVM 2-Polynomial Classifier minCprim, varying values of K (K = 1 Top, K = 10 Middle, K = 100 Bottom) (K-Fold=5)

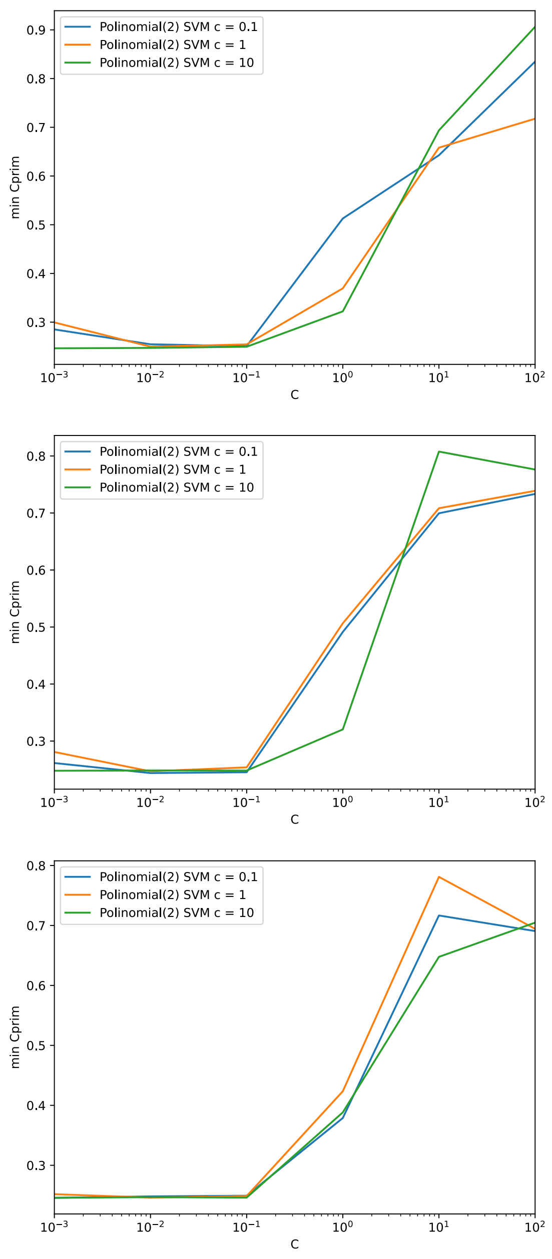


Figure 12: SVM 2-Polynomial Classifier minCprim with the application of PCA (5), varying values of K (K = 1 Top, K = 10 Middle, K = 100 Bottom) (K-Fold=5)

The best values of $minCprim$ are obtained mainly with values of C in the interval $C < 10^{-1}$, and for the considered values of c the models perform similarly. However the best parameters for this model that were experimentally individuated are $(c = 0.1, C = 0.01, K = 10)$.

Before taking another kernel in consideration, we first test a PCA dimensionality reduction to 5 features on our 2-Polynomial model.

The results are comparable to the ones obtained without applying any dimensionality reduction. Our best model selected in this case is characterized by the parameters $c = 0.1, K = 10, C = 0.1$.

We now try exploring SVM with a Polynomial Kernel degree higher than 2. We compare the models using the best Polynomial SVM configuration found so far.

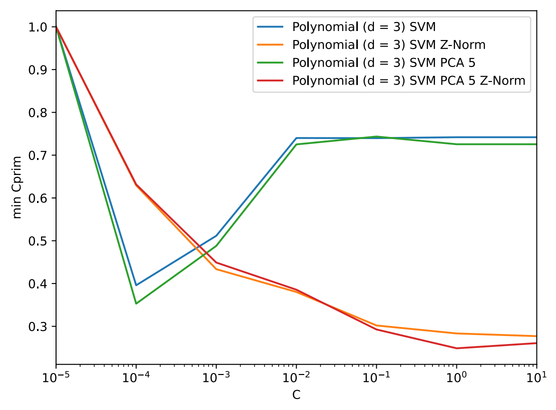


Figure 13: Comparison of SVM Poly-3 with the best parameters configurations with and without the application of PCA and Z-normalization.

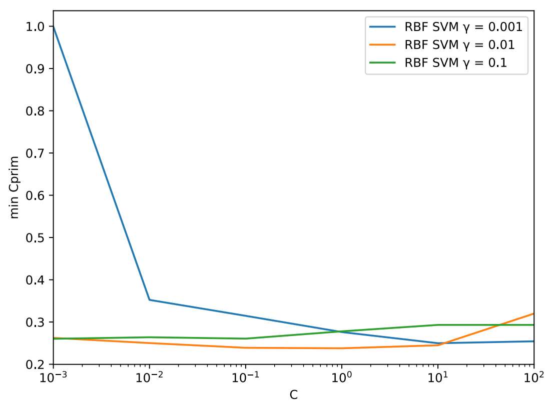


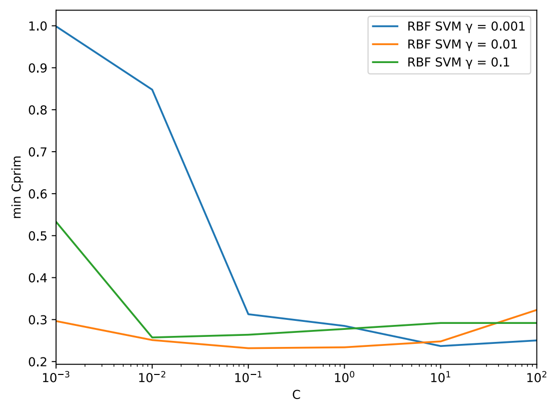
Figure 14: SVM RBF Classifier minCprim, varying values of K (K = 0.01 Top, K = 0.1 Middle, K = 1 Bottom) (K-Fold=5)

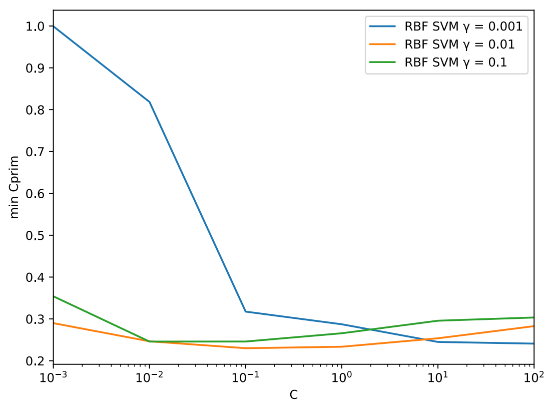
As can be seen in figure 13 the Polynomial Kernel SVM with degree 3 shows a high amount of over-fitting compared to the Polynomial Kernel SVM with degree 2. We can also notice that normalization does bring some improvement of the performance of these models, but only for $C > 10^{-3}$.

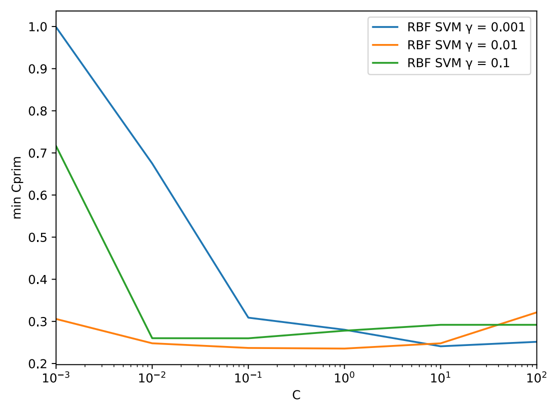
Now we consider the RBF kernel for different values of K.

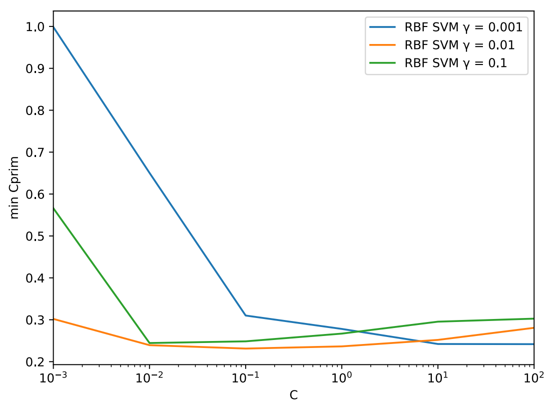
From the graphs shown in Figure 14 we can assume that the optimal values for C fall in the interval $[10^0, 10^2]$ mostly for all considered values of γ. However our best performance is obtained with the combination $(\gamma = 0.01, C = 0.1, K = 0.01)$.

We now try to see the effects of dimensionality reduction on the use of the RBF kernel.









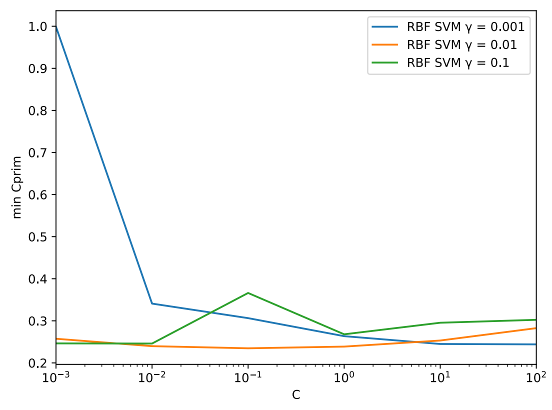


Figure 15: SVM RBF Classifier minCprim with PCA 5, varying values of K (K = 0.01 Top, K = 0.1 Middle, K = 1 Bottom) (K-Fold=5)

Reading the graphs we cannot see any visible improvement in the performance of our RBF SVM Model, but still we obtain very good results. The best model is considered to be obtained with the combination $(\gamma = 0.01, C = 0.1, K = 0.01)$ which is the same combination found above.

Kernel	PCA	K	C	minDCF($\tilde{\pi}$=0.1)	minDCF($\tilde{\pi}$=0.5)	min C_{prim}
Linear	5	1	0.001	1	0.481	0.740
2-Poly (c=0.1)	5	10	0.01	0.386	0.101	0.243
RBF (γ=0.01)	5	0.01	0.1	0.366	0.093	0.230

Table 7: Best SVM results

Before making a choice on our best SVM model we decide to test if Z-normalization improves our results with the best values of K found so far.

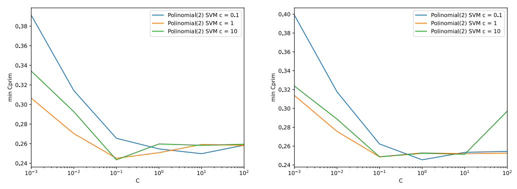


Figure 16: Polynomial(2) with $K = 10$ SVM with the application of Z-Normalization. In the right without the application of PCA and in the left with the application of PCA(5).

For the Polynomial-2 model we note a similar behaviour, however the z-normalization doesn't bring any improvement.

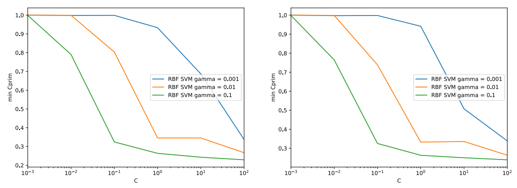


Figure 17: RBF with $K = 0.01$ SVM with the application of Z-Normalization. In the right without the application of PCA and in the left with the application of PCA(5).

For the RBF model Z-normalization brings our model still close to our last chosen model.

Given that our best model seems to be the RBF SVM with parameters $(\gamma = 0.01, C = 0.1, K = 0.01)$ without the application of Z-normalization, as we did for the Logistic Regression model, we will try to consider the balanced version of this SVM classifier, accounting for the unbalanced nature of the dataset, so we can account for different costs. In this case, balancing is executed modifying the box constraint of the dual formulation:

$$C_i = \begin{cases} C\frac{\pi_T}{\pi_T^{emp}} & if\ i = T \\ C\frac{\pi_F}{\pi_F^{emp}} & if\ i = F \end{cases} \tag{14}$$

Our results are shown in the table below:

PCA	π_T	minDCF($\tilde{\pi}$=0.1)	minDCF($\tilde{\pi}$=0.5)	min C_{prim}
5	0.17	0.384	0.102	0.243
5	0.1	0.384	0.100	0.242
5	0.2	0.391	0.101	0.246
5	0.5	0.390	0.106	0.708

Table 8: Balanced 2-Polynomial models with different priors

PCA	π_T	minDCF($\tilde{\pi}$=0.1)	minDCF($\tilde{\pi}$=0.5)	min C_{prim}
5	0.17	0.373	0.094	0.233
5	0.1	0.374	0.092	0.233
5	0.2	0.371	0.088	0.230
5	0.5	0.388	0.094	0.241

Table 9: Balanced RBF models with different priors

Given our results, we conclude that our best SVM model so far is the RBF model with parameters $(\gamma = 0.01, C = 0.1, K = 0.01)$ with prior $\pi_T = 0.2$ after the application of PCA-5.

4.4 Gaussian Mixture Models

Now we move our attention to Gaussian Mixture Models. GMMs assume that the observed data is generated from a mixture of several Gaussian distributions, each representing a different cluster or component within the data. These Gaussian distributions are combined using a set of weights, which determine the contribution of each component to the overall distribution.

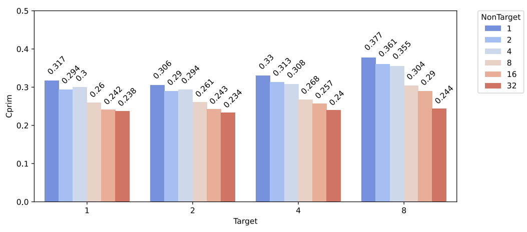


Figure 18: GMM Full Covariance Cprim (K-Fold=5)

As we can see from Figure 18, in the case of the Full Covariance model, the best number of components for the non target class seems to be 2^5 and 2 for the target class. One interesting observation is that the performance of our model improves as we increase the number of components dedicated to the non-target class.

We now try to consider PCA for GMMs, the results are shown in Table 10.

PCA	Tar K	Non Tar K	minDCF($\tilde{\pi}$=0.1)	minDCF($\tilde{\pi}$=0.5)	min C_{prim}
-	2	32	0.374	0.093	0.234
5	2	32	0.390	0.095	0.242
4	2	32	0.443	0.106	0.275

Table 10: GMM Full Covariance minCDF and Cprim (K-Fold=5)

Dimensionality reduction doesn't bring any improvement to our system so we stop considering it for our GMM models.

Considering our analysis of the Gaussian models, we decide to also analyse the behaviour of Diagonal GMM models, since diagonal Gaussian models gave us better results than Multivariate Gaussian models.

Tar K	Non Tar K	min DCF ($\tilde{\pi}$=0.1)	min DCF ($\tilde{\pi}$=0.5)	min C_{prim}
2 (FC)	32(FC)	0.374	0.093	0.234
2 (FC)	32(D)	0.341	0.091	0.216
2 (FC)	32(FC-T)	0.332	0.085	0.208
2 (FC)	32(D-T)	0.346	0.086	0.216
4 (D)	32 (FC)	0.371	0.092	0.231
1 (D)	32 (D)	0.337	0.083	0.210
1 (D)	32 (FC-T)	0.336	0.082	0.209
2 (D)	32 (D-T)	0.348	0.081	0.215

Table 11: GMM Best Model configurations for Diagonal and Full Covariance type

As we expected, considering the target class a few components are enough for most of our models.

For the non-target class our models require a higher number of components.

We can notice that even if we supposed the diagonal model to be have a higher performance, the tied covariance model performs slightly better, probably due to the fact that the different sub-classes of the non-target class might have a similar distribution, as we supposed earlier.

To further analyse the effects of z-normalization on the GMM systems we tested our best GMM model on normalized data, which lead to slight a degradation of the $minCprim$ value, this is shown below in table 12.

Tar K	Non Tar K	min DCF ($\tilde{\pi}$=0.1)	min DCF ($\tilde{\pi}$=0.5)	min C_{prim}
2 (FC)	32(FC)	0.364	0.101	0.232
2 (FC)	32(D)	0.349	0.095	0.222
2 (FC)	32(FC-T)	0.357	0.100	0.229
2 (FC)	32(D-T)	0.361	0.087	0.224
4 (D)	32 (FC)	0.337	0.085	0.211
1 (D)	32 (D)	0.344	0.082	0.213
1 (D)	32 (FC-T)	0.369	0.093	0.231
2 (D)	32 (D-T)	0.347	0.080	0.214

Table 12: GMM Best Model configurations for Diagonal and Full Covariance type

Therefore the model $[2(FC), 32(FC-T)]$ is our best GMM model so far.

5 Scores calibration and Fusion

Summarizing, our best models are listed in Table 13, we have decided to consider only the best three models.

Model	min DCF ($\tilde{\pi}$=0.1)	min DCF ($\tilde{\pi}$=0.5)	min C_{prim}	actDCF ($\tilde{\pi}$=0.1)	actDCF ($\tilde{\pi}$=0.5)	C_{prim}
GMM	0.332	0.085	0.208	0.349	0.086	0.217
RBF SVM	0.371	0.088	0.230	1	0.135	0.567
QLR	0.380	0.103	0.242	1	0.203	0.601

Table 13: C_{prim} and $minC_{prim}$ values for each best model.

Before diving into the next section, we compare the performance of our three models using a DET (Detection Error Trade-Off) plot.

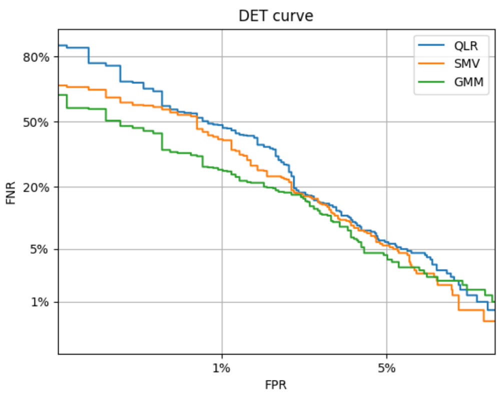


Table 14: DET curves for the best systems.

We can notice how the GMM model seems to work best is most regions of the curve.

5.1 Scores calibration

Up until now in our study we have considered the $minDCF$ to evaluate the models results for our working points. The $minDCF$ metric computes the classification cost that we would pay if we made the optimal decisions for the test set using the scores of the model.

To be more specific we obtain the $minDCF$ through a normalization and a minimization of the Empirical Bayes risk(unnormalized DCF) where C_{fn}, C_{fp} represents the false negative and false positive costs.

$$DCF_u(C_{fn}, C_{fp}, \pi_T) = \pi_T C_{fn} FNR + (1-\pi_T) C_{fp} FPR \qquad (15)$$

However in a real case scenario the classification cost that we pay due to a misclassification depends on the threshold that we use to make class assignments. To represent this phenomena we define the $actualDCF$ as the metric obtained by using the theoretical threshold that optimizes the Bayes risk

$$t = -log\frac{\tilde{\pi}}{1-\tilde{\pi}} \qquad (16)$$

This threshold is only theoretical and doesn't work in situations where the scores are obtained with non-probabilistic model or there is an unbalanced distribution of the classes between the training set and the test set.

The calibration strategy we used for our analysis consists in a K-fold (5 folds) approach consisting in two steps:

1. The model training is done through a K-Fold approach which outputs the scores of our model.

2. The scores of our model are calibrated with our Logistic Regression model with again a K-Fold approach which lets us obtain the calibrated scores.

The scores obtained through the first K-fold will be used to obtain a calibration model C_F model that will be used to calibrate the scores obtained in the Evaluation Phase.

However we must note that the $minDCF$ value may be subject to variations since our scores and labels must go through a reshuffling phase for the obtainment of the calibration set.

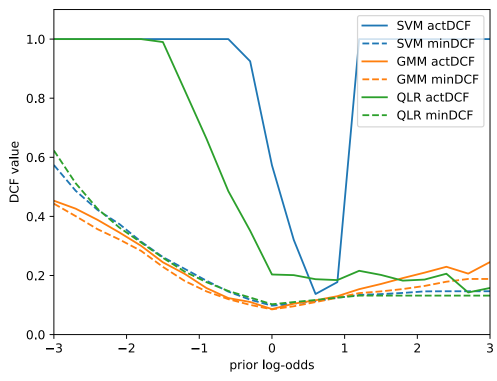


Figure 19: Bayes error plots without calibration.

Taking a look at our plots we can see that the GMM model seems already well calibrated, while the Quadratic Logistic Regression and the RBF Support Vector Machine model need calibration. In Table 13 we can see the values of C_{prim} and $minC_{prim}$ corresponding to our plots.

To execute score calibration we will employ the prior-weighted Logistic regression model to learn the model parameters α and β over our training scores. Then to recover the calibrated scores $f(s)$ we will need to compute:

$$f(s) = \alpha s + \beta - \log\frac{\tilde{\pi}}{1-\tilde{\pi}} \qquad (17)$$

Model	π_T	min DCF ($\tilde{\pi}$=0.1)	min DCF ($\tilde{\pi}$=0.5)	min C_{prim}	actDCF ($\tilde{\pi}$=0.1)	actDCF ($\tilde{\pi}$=0.5)	C_{prim}
GMM	π_T^{emp}(0.17)	0.345	0.087	0.216	0.491	0.141	0.316
	0.1	0.345	0.088	0.216	0.570	0.201	0.386
	0.2	0.345	0.087	0.216	0.453	0.128	0.291
	0.3	0.347	0.087	0.217	0.385	0.092	0.239
	0.4	0.347	0.087	0.217	0.357	0.090	0.223
	0.5	0.348	0.087	0.217	0.354	0.091	0.222
RBF SVM	π_T^{emp}(0.17)	0.371	0.089	0.230	0.526	0.184	0.355
	0.1	0.371	0.091	0.231	0.611	0.247	0.429
	0.2	0.371	0.091	0.231	0.498	0.169	0.334
	0.3	0.371	0.091	0.231	0.435	0.127	0.281
	0.4	0.371	0.091	0.231	0.415	0.112	0.263
	0.5	0.371	0.091	0.231	0.376	0.108	0.242
Q-Log-Reg	π_T^{emp}(0.17)	0.384	0.102	0.243	0.675	0.144	0.410
	0.1	0.381	0.102	0.242	0.858	0.186	0.522
	0.2	0.384	0.101	0.242	0.611	0.122	0.366
	0.3	0.385	0.101	0.243	0.511	0.110	0.311
	0.4	0.385	0.101	0.243	0.432	0.107	0.269
	0.5	0.385	0.102	0.243	0.392	0.112	0.252

Table 15: C_{prim} and $minC_{prim}$ values for each best model after the application of score calibration varying the values of π_T for the Logistic Regression Model.

From Table 15 we can notice that the calibration actually produces different results with different values of π_T. In this case, we decide to choose the models trained with $\pi_T = 0.5$ which gives us the best calibration results overall, even if there is still some miscalibration for some working points. Note that because of the mono-dimensional nature of the scores we do not need any regularization term λ, therefore our best calibration model is characterized by the parameters $\pi_T = 0.5$ and $\lambda = 0$.

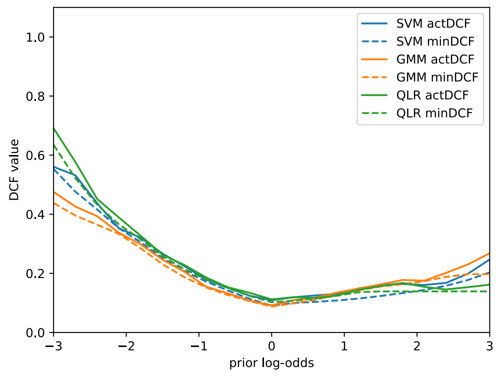


Figure 20: Bayes error plots with calibration done with $\pi_T = 0.5$.

Based on the findings presented in Figure 20, we can say that they are consistent with our expectations, score calibration has been shown to be effective for the Quadratic Logistic Regression Model and the Support Vector Machine model. The GMM model, on the other hand, was already producing well-calibrated scores without the need for additional calibration, in fact the performance of our GMM model deteriorates with calibration.

We report in the Table 16 the results of our calibration:

Model	min DCF ($\tilde{\pi}$=0.1)	min DCF ($\tilde{\pi}$=0.5)	min C_{prim}	actDCF ($\tilde{\pi}$=0.1)	actDCF ($\tilde{\pi}$=0.5)	C_{prim}
GMM	0.348	0.087	0.217	0.354	0.091	0.222
RBF SVM	0.371	0.091	0.231	0.376	0.108	0.242
QLR	0.385	0.102	0.243	0.392	0.112	0.252

Table 16: C_{prim} and $minC_{prim}$ values for each best model after the application of score calibration.

5.2 Fusion

After managing the calibration of our models, we proceed with score fusion. Given that fusion takes into consideration the results of different systems, it might help us obtain a lower degree of over-fitting.

Score fusion is a technique which takes as input the scores produced by two or more different systems and uses them as input for a logistic regression model. The result of training is a set of coefficients $\boldsymbol{w}$ which lets us compute a linear weighting of the scores from multiple systems. Then these are used to fuse and calibrate the scores of the two different systems:

$$f(s) = w_1 s_1 + w_2 s_2 + ... + w_n s_n + b \quad (18)$$

Where $s_1...s_n$ are the scores of the n^{th} system and $w_1...w_n$ are the weights produced by the logistic regression training.

The logistic regression models used for the fusion of our models have been chosen through testing different values of λ and π_T, we have specifically chosen $\lambda = 0.001$ and $\pi_T = 0.4$.

Model	min DCF ($\tilde{\pi}$=0.1)	min DCF ($\tilde{\pi}$=0.5)	min C_{prim}	actDCF ($\tilde{\pi}$=0.1)	actDCF ($\tilde{\pi}$=0.5)	C_{prim}
[1] GMM	0.348	0.087	0.217	0.354	0.091	0.222
[2] RBF SVM	0.371	0.091	0.231	0.376	0.108	0.242
[3] Q-Log-Reg	0.385	0.102	0.243	0.392	0.112	0.252
[1] + [2]	0.344	0.084	0.214	0.375	0.088	0.232
[1] + [3]	0.364	0.085	0.225	0.383	0.091	0.237
[2] + [3]	0.375	0.096	0.235	0.417	0.101	0.259
[1] + [2] + [3]	0.351	0.085	0.218	0.391	0.090	0.240

Table 17: Values of $minCprim$ and $Cprim$ for the best chosen models and their fusions.

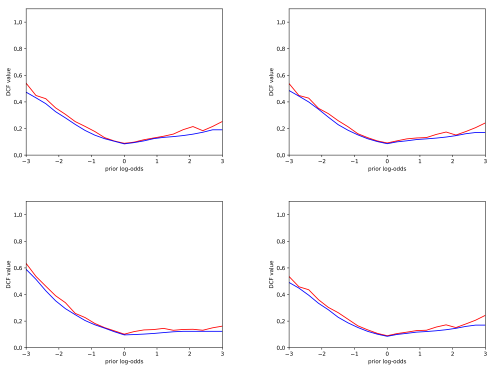


Figure 21: Bayes error plots for the fusion models [1] + [2] (Top-Left), [1] + [3] (Top-Right), [2] + [3] (Bottom-Left), [1] + [2] + [3] (Bottom-Right), showing the *minCprim* plots in blue and the *Cprim* plots in red.

From table 17 and the error plots shown in figure 21 we can asses that the fusion of our models seems to produce good results, with GMM + RBF SVM as the best system which slightly improves the results compared to the GMM model.

6 Evaluation Phase

We now analyse the performance on the evaluation set. At this stage, we do not train any any new model, but we simply go through the re-evaluation of the best models we have trained on the training set as a whole. However, we won't consider linear models as we have assessed that they are not suited to our target applications.

Before going through the re-analysis of all our models we report the performance of our chosen configurations on the evaluation set.

We note that calibration for the scores in the evaluation set was executed with a calibrator C_F trained on the calibration set.

Model	Validation Set		Evaluation Set	
	min C_{prim}	C_{prim}	min C_{prim}	C_{prim}
[1] GMM	0.217	0.222	0.256	0.262
[2] RBF SVM	0.231	0.242	0.270	0.288
[3] Q-Log-Reg	0.243	0.252	0.264	0.284
[1] + [2]	0.214	0.232	0.250	0.264
[1] + [3]	0.225	0.237	0.239	0.272
[2] + [3]	0.235	0.259	0.260	0.276
[1] + [2] + [3]	0.218	0.240	0.236	0.264

Table 18: Comparison of the values of *minCprim* and *Cprim* for the best models and their fusions on the Validation Set and on the Evaluation Set.

From table 18 we can observe that costs are slightly higher than on the validation set, this might be caused by over-fitting, because as we can see Fusion now behaves better than the chosen models.

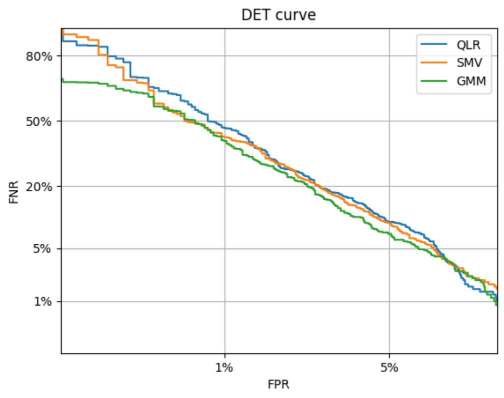


Figure 22: DET curves for the best configurations found in the validation phase with evaluation data.

The DET plot shown in figure 22 still confirms that the GMM is the best model in a wide range of applications, but the SVM model could be competitive in a small higher region, while the Q-Log-Reg in a final lower region.

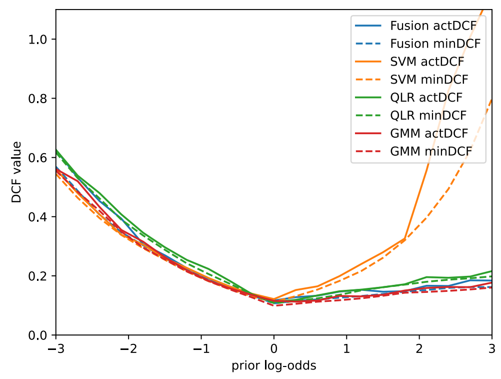


Figure 23: Bayes Plot curves for the best systems in evaluation.

The Bayes Error plots in figure 23 confirms our findings read from the DET plot above, our models still present some degree of mis-calibration. However, the best model taken between the chosen models still proves to be the GMM model, while the best model overall is still the Fusion of QLR + GMM + SVM.

Since we will analyze our models in terms of *minCprim* to find the optimal configuration, we won't consider calibration anymore in our analysis as we have already seen it is effective.

6.1 Gaussian Classifiers

Just for the sake of analysis, we're presenting the results of Gaussian models on the evaluation set. The table below showcases the top-performing result for each model version.

Model	PCA	Evaluation Set min C_{prim}
MVG	5	0.324
NAIVE	5	0.325

Figure 24: Best systems for the Gaussian models on the evaluation set.

From the table above we can see that the best system is given by the Multivariate Model used on data reduced to 5 dimensions via PCA. However this best configuration is found to be barely better than our best configuration found in validation (Naive model with 5-dimensional PCA).

6.2 Quadratic Logistic Regression

As in the Validation phase, unbalanced logistic regression has been tested. Here we plot the *minCprim* value varying λ and the dimensionality of PCA.

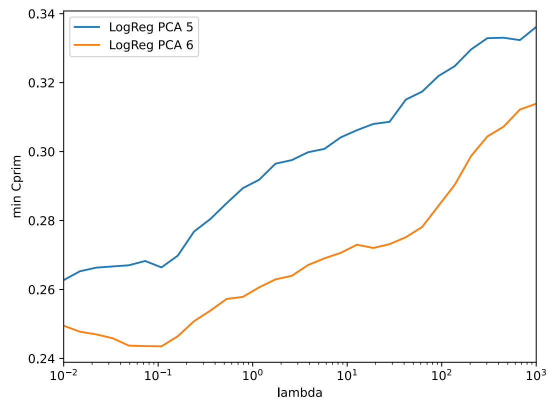


Figure 25: *minCprim* value plotted for various values of *lambda*.

From the graph we can note that the best models are obtained with $\lambda = 0.1$ and without any employment of PCA.

Now, we will apply prior weighting on our logistic regression model with the best value of λ found so far.

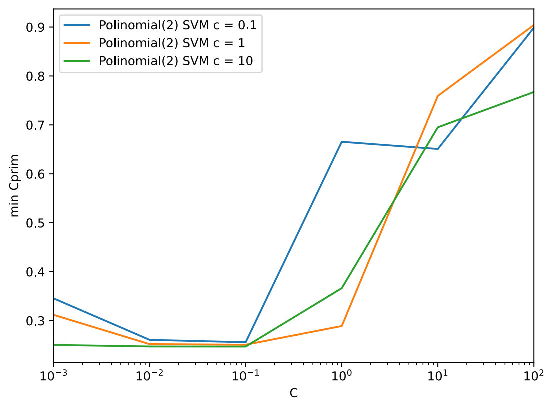
π_T	λ	Validation Set min C_{prim}	λ	Evaluation Set min C_{prim}
$\pi_T^{emp}(0.17)$	10	0.245	0.1	0.243
0.1	10	0.242	0.1	0.245
0.2	10	0.243	0.1	0.243
0.5	10	0.243	0.1	0.247

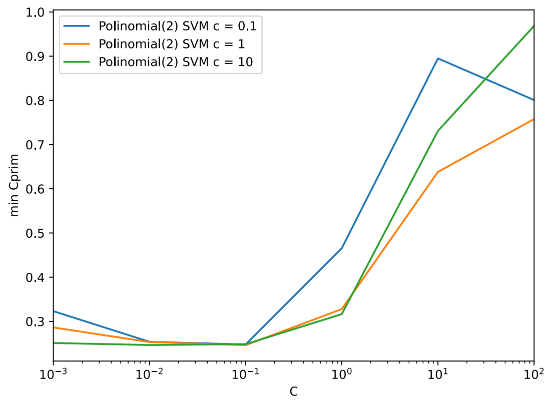
Table 19: *minCprim* value varying the values of π_T.

As we can see, the optimal configuration is still different than the chosen configuration. In fact the optimal configuration seems to be given by $\pi_T = \pi_T^{emp}$, $\lambda = 0.1$ without any application of PCA.

6.3 Support Vector Machines

As for Support Vector Machines, we reconsider the Polynomial quadratic model to see if it would perform better than our chosen RBF configuration. Below in figure 26 we report the value of *minCprim* varying C and c with and without the use of PCA.





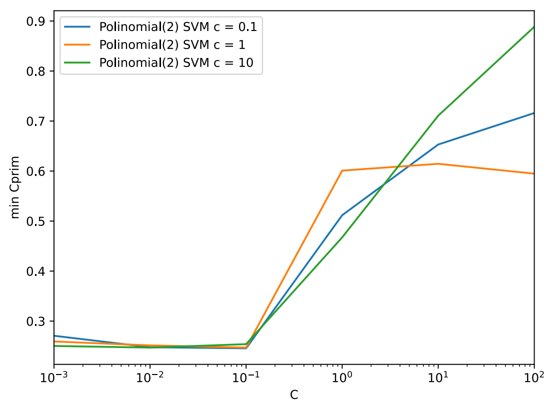


Figure 26: SVM Polynomial(2) Classifier minCprim, varying values of K (K = 1 Top, K = 10 Middle, K = 100 Bottom) (Evaluation)

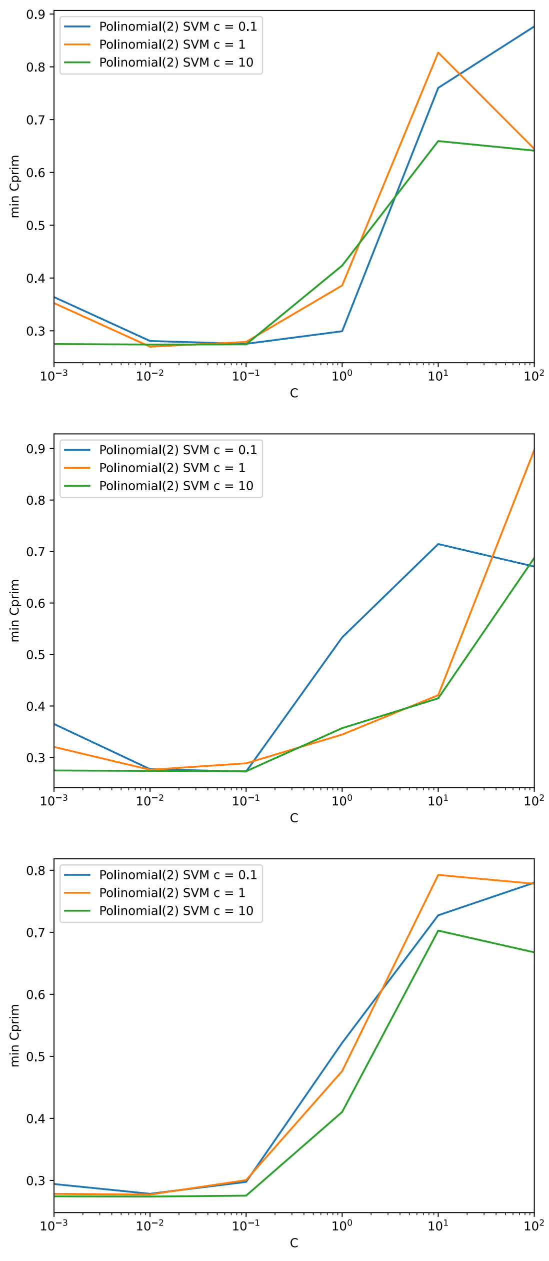


Figure 27: SVM Polynomial(2) Classifier minCprim, varying values of K (K = 1 Top, K = 10 Middle, K = 100 Bottom) (Evaluation) with the use of PCA (5)

We conclude that the best system is given by $K = 100$, $C = 0.1$, $c = 0.1$ with $minCprim = 0.245$ which however doesn't perform better than our chosen RBF model.

Now we analyze the performance of the RBF variant of our SVM models in the figures below.

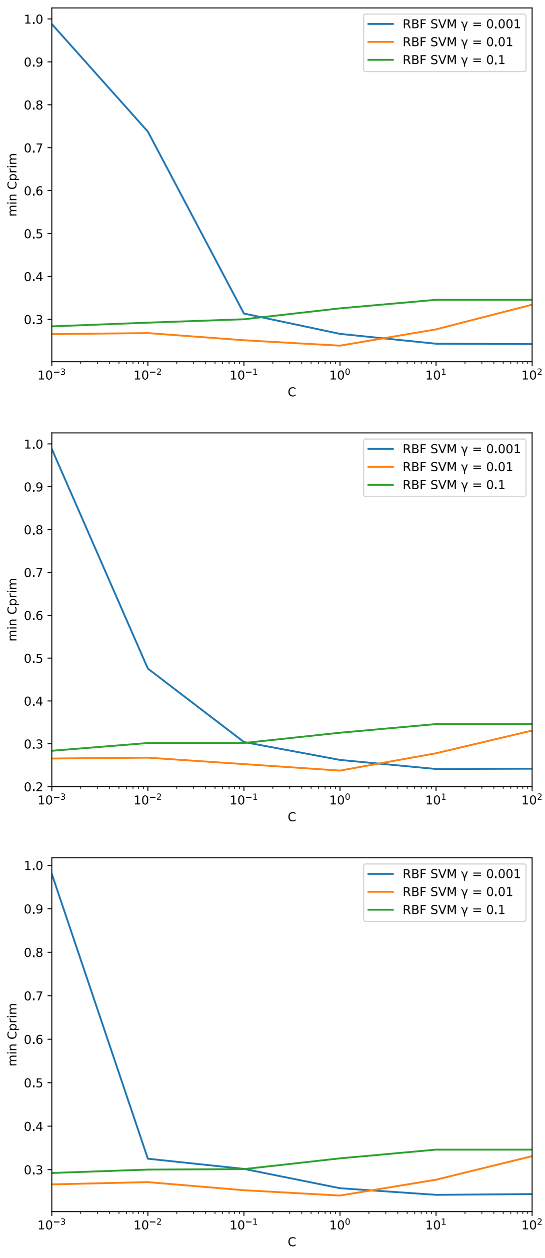


Figure 28: SVM RBF Classifier minCprim, varying values of K (K = 0.01 Top, K = 0.1 Middle, K = 1 Bottom) (Evaluation)

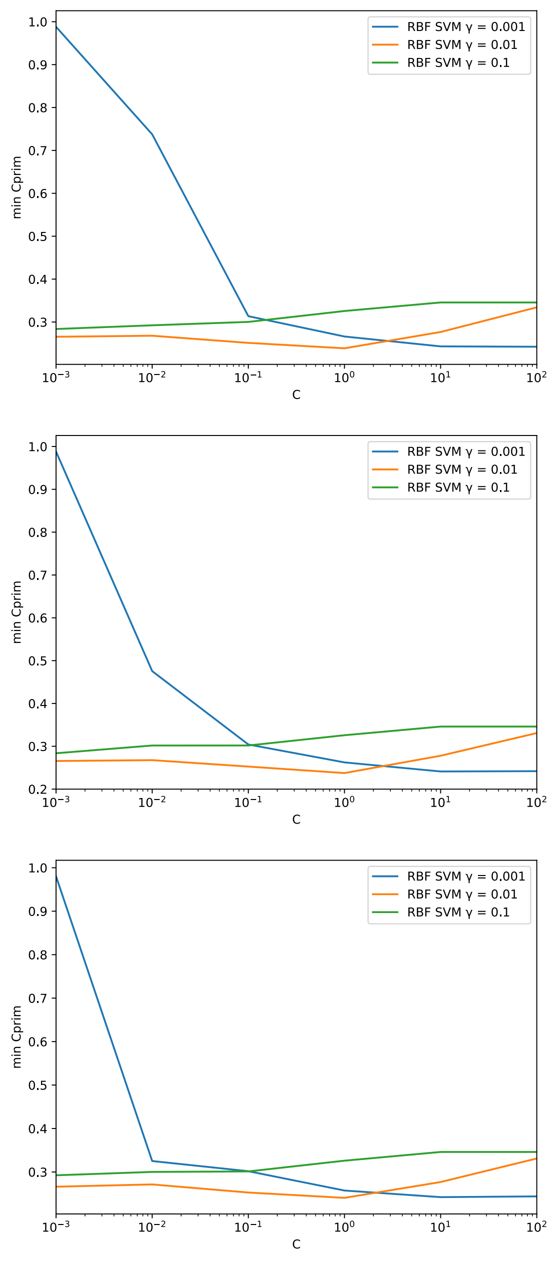


Figure 29: SVM RBF Classifier minCprim, varying values of K (K = 0.01 Top, K = 0.1 Middle, K = 1 Bottom) (Evaluation) with the use of PCA (5)

Given that in this case our best configuration is given by the hyper-parameters $K = 0.01$, $\gamma = 0.01$, $C = 1$ with $minCprim = 0.234$ without any employment of PCA, we now employ the balanced version of RBF with this configuration.

	Validation Set min C_{prim}	Evaluation Set min C_{prim}
π_T	K = 0.01, C = 0.1, γ = 0.01	K = 0.01, C = 1, γ = 0.01
π_T^{emp}(0.17)	0.233	0.234
0.1	0.233	0.233
0.2	0.230	0.238
0.5	0.241	0.269

Table 20: $minCprim$ value varying the values of π_T for the respective best parametric configurations.

From table 20 we can see that we obtain the best $minCprim$ with a $\pi_T = 0.1$. So our Optimal configuration differs from the chosen configuration and it is characterized by the parameters $K = 0.1$, $\gamma = 0.01$, $C = 1$ and $\pi_T = 0.1$.

6.4 Gaussian Mixture Models

Finally we analyze the Gaussian Mixture Models, here we will report the configurations considered in Validation to avoid any redundancy, however we can confirm that they are still the best configurations among the one tested for evaluation. We note that the choice for PCA was however optimal, so as in the Validation phase our best configurations do not employ PCA.

Tar K	Non Tar K	Validation Set min C_{prim}	Evaluation Set min C_{prim}
2 (FC)	32(FC-T)	0.234	0.303
2 (FC)	32(D)	0.242	0.253
2 (FC)	32(FC-T)	0.208	0.256
2 (FC)	32(D-T)	0.216	0.251
4 (D)	32 (FC)	0.218	0.267
1 (D)	32 (D)	0.210	0.238
1 (D)	32 (FC-T)	0.212	0.239
2 (D)	32 (D-T)	0.215	0.235

Table 21: GMM Best Model configurations for Diagonal and Full Covariance type, operating on the Validation and Evaluation Set.

Table 21 reports that the Chosen configuration is revealed to be sub-optimal compared to the optimal configuration $[2(D), 32(D-T)]$. This could be caused by the fact that our previous chosen model is affected by over-fitting.

6.5 Fusion Models

In this section we consider now the effects of model fusion on our classification task, comparing our Chosen configurations with the Optimal configurations we just found.

Model	Chosen Conf min C_{prim}	Optimal Conf min C_{prim}
[1] GMM	0.256	0.235
[2] RBF SVM	0.270	0.233
[3] Q-Log-Reg	0.264	0.259
[1] + [2]	0.250	0.232
[1] + [3]	0.239	0.233
[2] + [3]	0.260	0.241
[1] + [2] + [3]	0.236	0.234

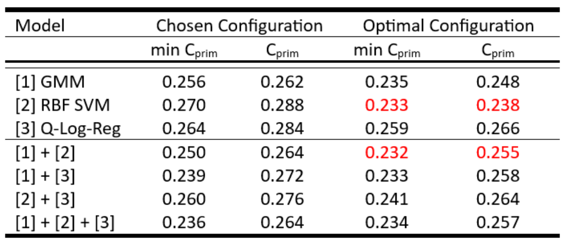
Table 22: Values of $minCprim$ for the fusion of the chosen configuration and the fusion of the optimal configurations found during the evaluation phase.

From 22 we can see that Fusion still helps us reduce over-fitting, lowering the values of $minCprim$.

In this case we obtain the best results with the SVM RBF + GMM configuration.

7 Conclusions

In this section, we summarize the findings and implications derived from our analysis, taking a look at table 23.



Model	Chosen Configuration		Optimal Configuration	
	min C_{prim}	C_{prim}	min C_{prim}	C_{prim}
[1] GMM	0.256	0.262	0.235	0.248
[2] RBF SVM	0.270	0.288	0.233	0.238
[3] Q-Log-Reg	0.264	0.284	0.259	0.266
[1] + [2]	0.250	0.264	0.232	0.255
[1] + [3]	0.239	0.272	0.233	0.258
[2] + [3]	0.260	0.276	0.241	0.264
[1] + [2] + [3]	0.236	0.264	0.234	0.257

Table 23: Values of minCprim and Cprim comparing the chosen configuration and the optimal configuration after calibration.

As we supposed from the beginning, the best models are in every case characterized by quadratic classification rules, as the linear variations of our models weren't able to perform well on the validation phase. This was supposed prior to our analysis as the features distribution was not linearly separable as seen during the feature analysis section. Therefore the best performing models selected during validation phase were:

- GMM $[2(FC), 32(FC - T)]$
- SVM RBF $\gamma = 0.01$, $C = 0.1$, $K = 0.01$, $\pi_T = 0.2$ + PCA(5)
- Q-Log-Reg $\pi_T = 0.1$, $\lambda = 10$
- GMM + RBF

During validation we noticed that the GMM models behave better than the other models, we suppose this happens because of the nature of the dataset, as utterances coming from the same language might be clustered in the same region, such as different clusters might refer to different languages in the dataset. We also notice that the Fusion models slightly improve our classification, probably because fusing the scores of different classifiers helps us achieve more generalization and therefore reduce over-fitting.

The evaluation phase revealed the effects of over-fitting of our models, as the configurations chosen in the validation phase were revealed to be sub-optimal. Therefore we report here our optimal configurations we found in the evaluation phase, where we noticed that our best performing model is by some points the RBF SVM model.

- GMM $[2(D), 32(D - T)]$
- SVM RBF $\gamma = 0.01$, $C = 1$, $K = 0.01$, $\pi_T = 0.1$
- Q-Log-Reg $\pi_T = \pi_T^{emp}$, $\lambda = 0.1$
- GMM + RBF

With the Fusion Model GMM + SVM RBF being the best system overall. However the calibration of the best system is not optimal, with the best configuration we would lose around 9% relative to poor calibration, while with the SVM we would only lose around 2%.

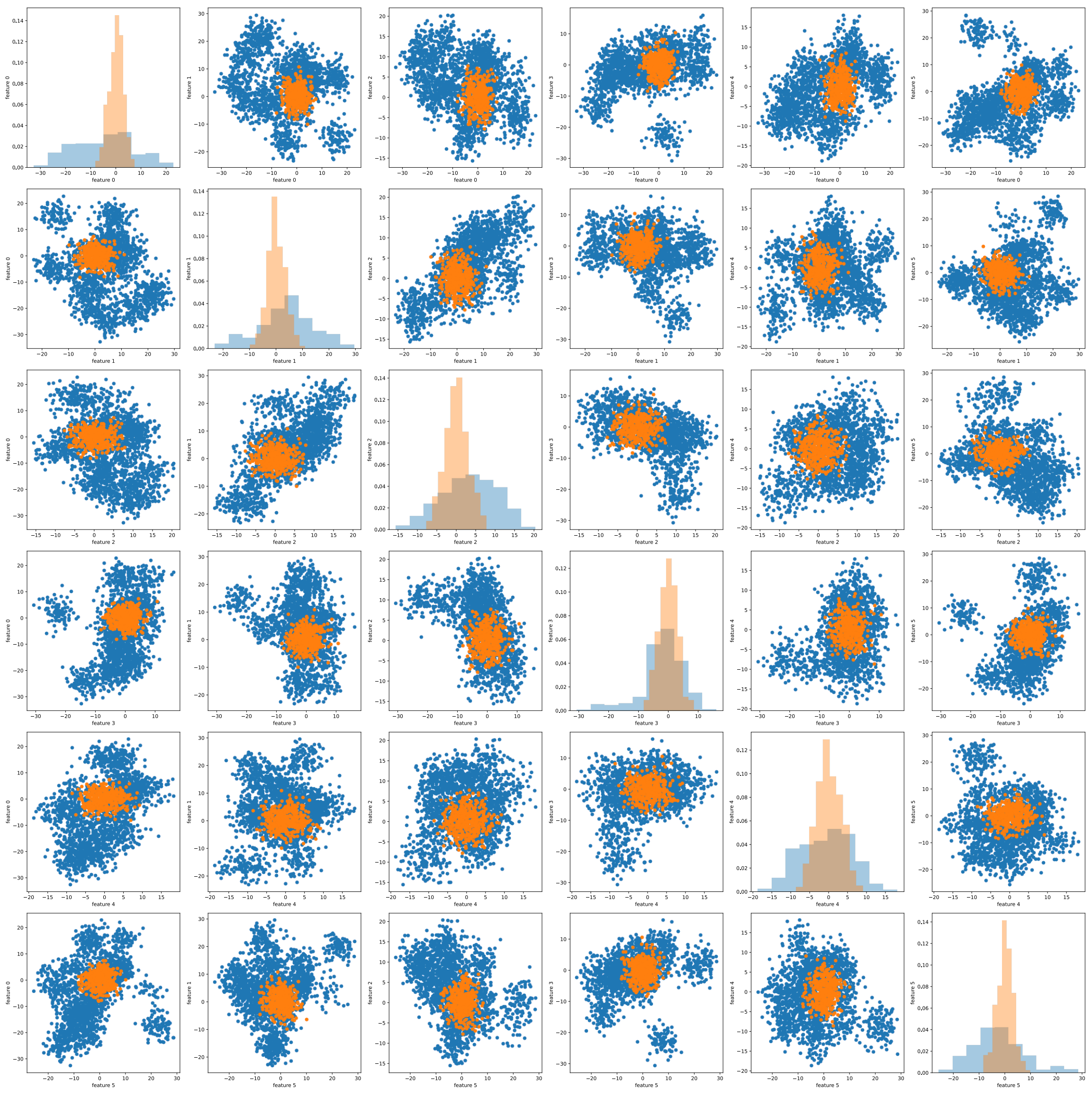


Figure 30: Scatter Plots