Polytechnic University of Turin

Master of Science in Computer Engineering

Machine learning and pattern recognition

Binary classification
Pulsar Detection

Lorenzo D'Amico

s296273

Academic year 2021–2022

Problem overview

Dataset information

The goal of this study is to test different models and find a good one for the classification of the pulsars, which are a type of Neutron star that release a pattern of broadband radio emission that we can detect from Earth. The dataset used in this project is the HTRU2, taken from the UCI repository. It contains a group pulsar candidates. They are of considerable scientific interest as probes of space-time, the inter-stellar medium, and states of matter.
The signal detection is called "candidate", averaged over many rotation of the pulsar. Each candidate could be a Pulsar, however most of them are just radio frequency interference and noise.
The HTRU2 is composed of 17898 total samples, 1639 of which are real pulsar, while the other 16259 are just noise. As we can tell, classes are highly imbalanced.
Each sample presents 8 features, that are continuous variables, plus a class variable that indicates if the sample is a Pulsar (1 - Positive), or if it's not (0 - Negative). The 8 features are:

- Mean of the integrated profile.
- Standard deviation of the integrated profile.
- Excess kurtosis of the integrated profile.
- Skewness of the integrated profile.
- Mean of the DM-SNR curve.
- Standard deviation of the DM-SNR curve.
- Excess kurtosis of the DM-SNR curve.
- Skewness of the DM-SNR curve.

The first four of which are simple statistics obtained from the integrated pulse profile. The remaining four are obtained from the DM-SNR curve.

Feature analysis

The dataset was divided into two parts: one for training, and one for testing.
The training set is composed of 8929 total samples, 821 are labeled as class positive so they regard real pulsar, while 8108 are labeled as class Negative, so they represent noise. The following analysis will be done using only the training set, while the test set will be used only at the end for evaluation purposes.
We can immediately plot the different features to see how they're distributed Figure 1

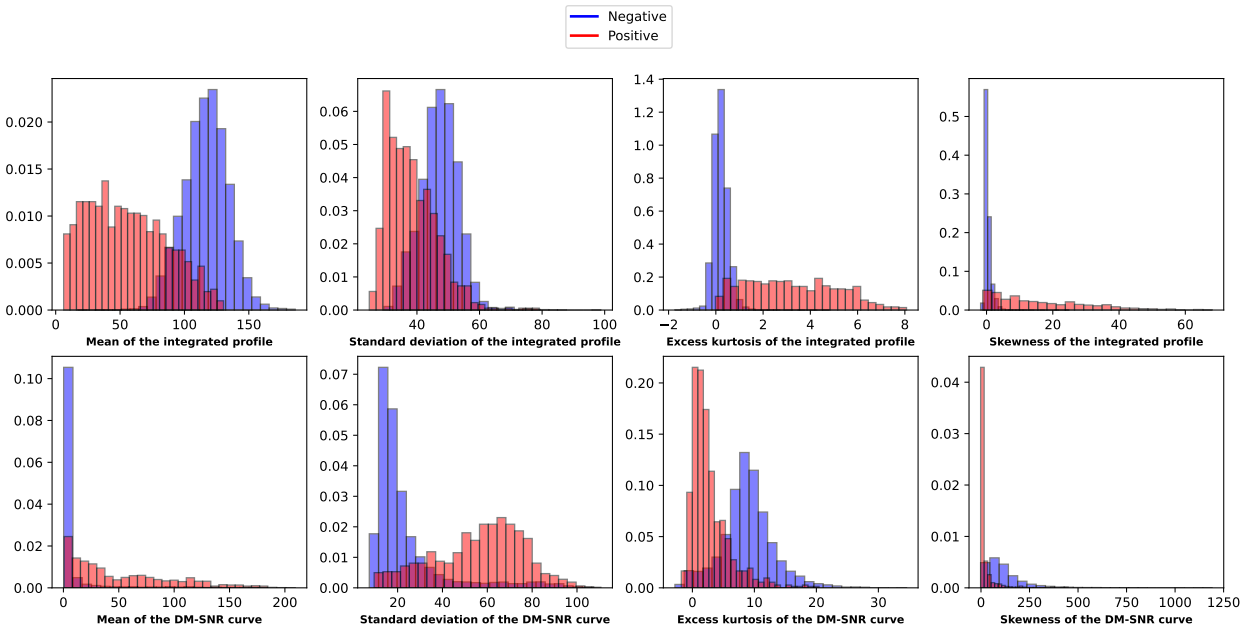

Figure 1: Histograms of raw features, separated by class

As we can see from the raw data, the features have different scales. To avoid issues with the algorithms we normalize the features using **Z-score normalization**, which consists in transforming the data such that the mean is 0 and the variance is 1.

$$Y = \frac{(X - \mu)}{\sigma}$$

where μ and *sigma* are the mean and variance of the feature, while X is the sample we want to normalize. The Z-normalized features can be seen in Figure 2. From now on, we will call the Z-normalized features "raw features".

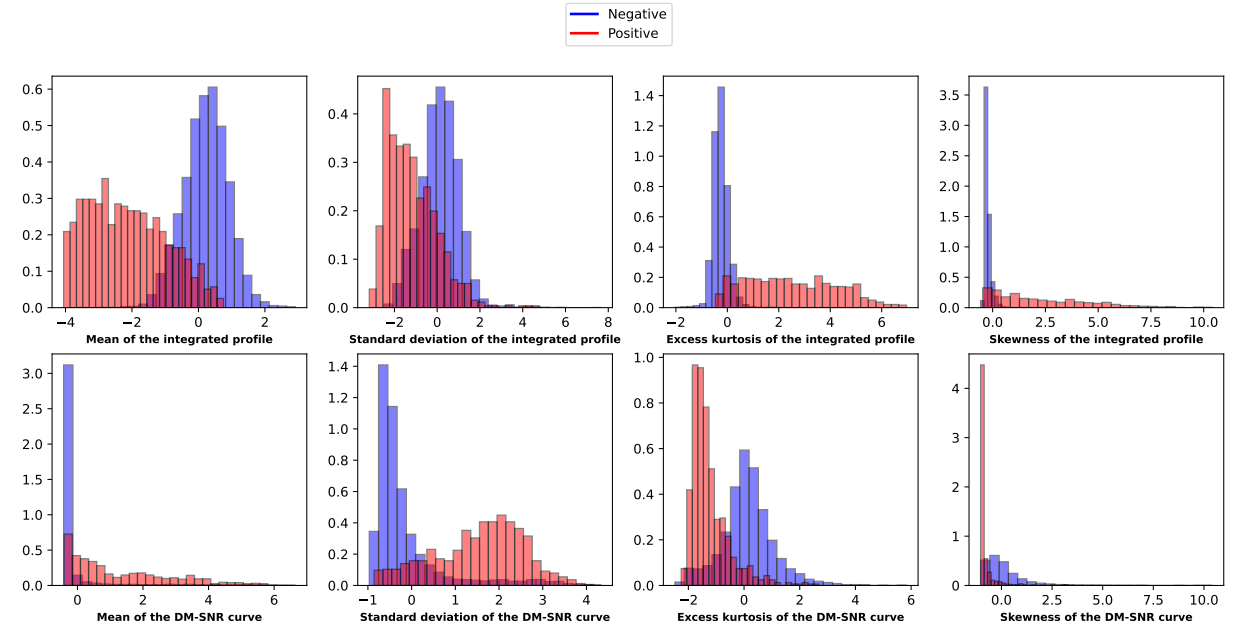

Figure 2: Histograms of z-normalized features, separated by class

As we can see now the features have all the same scale and have zero mean. From the data we don't see a significant presence of outliers, however we can still further pre-process our data trying the **Gaussianization** process to obtain more "gaussian-like" distributions.
Gaussianization is a procedure that allows mapping a set of features to values whose empirical comulative distributed function is well approximated by a Gaussian c.d.f. We don't expect gaussianized feature to give us better results because of the lack of outliers.

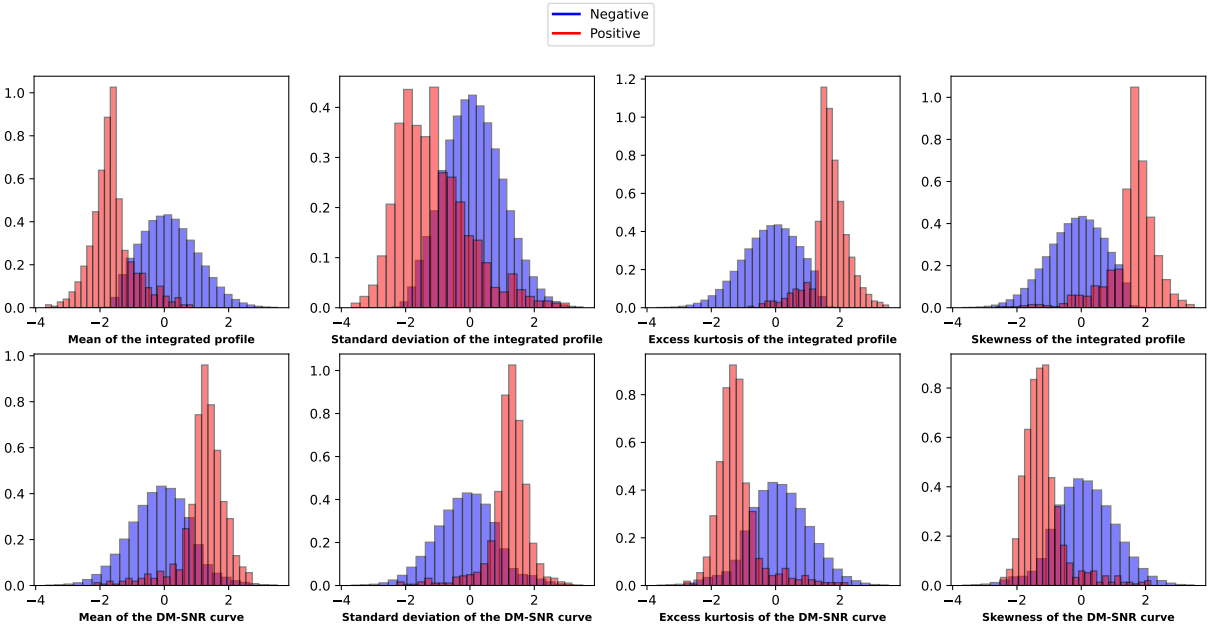

Figure 3: Histograms of gaussianized features, separated by class

What we can observe from Figure 3 is that the negative class seems to resemble a gaussian distribution, while the positive class doesn't. This is due to the fact that classes are highly imbalanced and the gaussianization procedure is done to the whole dataset, without taking in consideration the labels. Another relevant observation is that looking at both the raw and gaussianized features, the latter have a significant larger intersection between classes, which means that classes could be difficult to separate. Therefore we expect, in general, worse results with gaussianized features.

PCA

We can now calculate the *Pearson correlation coefficient* for each couple of features:

$$\left| \frac{Cov(X,Y)}{\sqrt{Var(x)}\sqrt{Var(Y)}} \right|$$

This tells us the correlation among features and gives us a hint of the features we can remove because they give the same information.

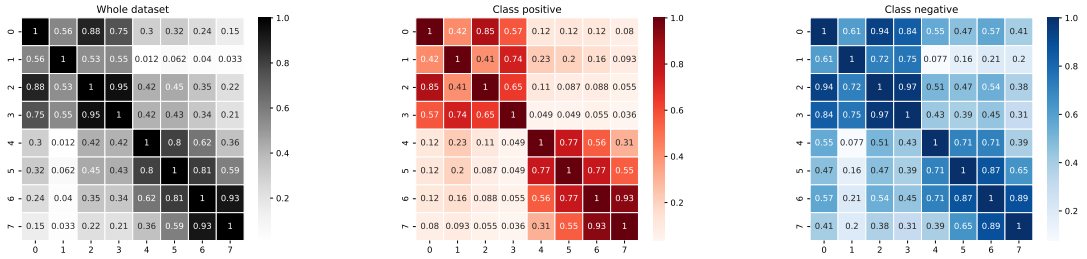

Figure 4: Heat map of the correlation matrix, raw features

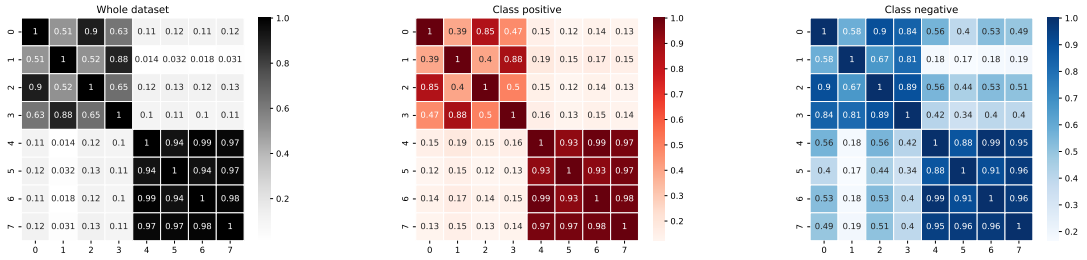

Figure 5: Heat map of the correlation matrix, gaussianized features

From the figure 4 of the raw features we can observe that there is a strong correlation between features 0 and 3, 2 and 3, 6 and 7, but also 6 and 5.
The gaussianized features reveal strong correlation between the last 4 features (Figure 5).
Overall is clear that there is stronger correlation between the first four features, and between the last four features. This can be explained by the fact that the first four features are obtained by the integrated pulse profile, while the last four are all obtained by the DM-SNR curve.
These correlations suggest that we can benefit from using PCA with a value of 7 or 6 features to reduce the number of parameters to estimate. We can even try a value of 5 features, however looking at the raw features it could be too much since the correlations are strong but not that much. Gaussianized features may benefit from the 5 features PCA reduction.

Analysis methodologies

To measure the performance of the different classifiers we're going to employ K-fold cross-validation with K = 5. This method was chosen over the single-fold to have more data for training and validation. Data has been shuffled before splitting, so that the data of different folds are homogeneous.
Classifiers are compared in terms of minimum DCF, which is the cost we would pay to make optimal decisions on test set using the recognizer scores.
We will consider different applications (π_T, C_{fp}, C_{fn}), reduced to the effective prior $\tilde{\pi}$. The applications we will consider are a unform prior and two unbalanced:

- $\tilde{\pi} = 0.5$ (Main application)
- $\tilde{\pi} = 0.1$
- $\tilde{\pi} = 0.9$

Classification

Gaussian classifiers

We start considering simple **Gaussian classifiers**. The models we consider are:

- Full covariance
- Naive Bayes (Diagonal covariance)
- Tied Full covariance
- Tied Diagonal covariance

The gaussian classifiers assume that the data from each class is gaussian distributed:

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

However looking at the histograms in Figure 2 we've seen that the classes are far from a gaussian distribution.

	5-fold		
	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$
Raw features - no PCA			
Full-Cov	0.141	0.283	0.68
Diago-Cov	0.193	0.314	0.734
Tied Full-Cov	0.112	0.223	0.562
Tied Diago-Cov	0.16	0.266	0.575
Gaussianized features - no PCA			
Full-Cov	0.153	0.248	0.716
Diago-Cov	0.152	0.278	0.612
Tied Full-Cov	0.131	0.235	0.53
Tied Diago-Cov	0.163	0.293	0.62
Raw features - PCA (m = 7)			
Full-Cov	0.14	0.3	0.649
Diago-Cov	0.214	0.504	0.721
Tied Full-Cov	0.112	0.223	0.56
Tied Diago-Cov	0.14	0.273	0.594
Gaussianized features - PCA (m = 7)			
Full-Cov	0.153	0.245	0.701
Diago-Cov	0.165	0.249	0.667
Tied Full-Cov	0.134	0.245	0.53
Tied Diago-Cov	0.136	0.253	0.561
Raw features - PCA (m = 6)			
Full-Cov	0.154	0.288	0.638
Diago-Cov	0.224	0.53	0.733
Tied Full-Cov	0.14	0.258	0.587
Tied Diago-Cov	0.165	0.3	0.614
Gaussianized features - PCA (m = 6)			
Full-Cov	0.154	0.243	0.69
Diago-Cov	0.158	0.244	0.642
Tied Full-Cov	0.137	0.247	0.551
Tied Diago-Cov	0.141	0.256	0.589
Raw features - PCA (m = 5)			
Full-Cov	0.15	0.25	0.653
Diago-Cov	0.22	0.455	0.737
Tied Full-Cov	0.149	0.26	0.597
Tied Diago-Cov	0.171	0.313	0.628
Gaussianized features - PCA (m = 5)			
Full-Cov	0.152	0.248	0.699
Diago-Cov	0.157	0.244	0.645
Tied Full-Cov	0.136	0.248	0.547
Tied Diago-Cov	0.142	0.257	0.592

Table 1: MVG Classifiers - min DCF on the validation set

Even with gaussianization, only the Negative class is gaussian distributed, while the Positive class is not. so we don't expect it to perform very good. The Tied model assumes that the covariance matrix is

shared among classes, while the diagonal one assumes that features are indipendent, so the covariance matrix becomes diagonal. From what we've seen from the correlation matrices, we also expect lower performance from the diagonal classifiers, since features are not independent. The results are visible in table 1 As we can observe, the best performance are obtained on the Raw features and on the PCA with m = 7.
As we expected gaussianization didn't perform better compared to the others, it outperforms the raw features only on tied full covariance with $\tilde{\pi} = 0.9$. In general, however, it performs slightly worse, but gives more robustness with PCA since it increases the correlation between features. The tied models perform better probably because covariance matrices are similar. Moreover, since the tied model uses linear classification rules, we expect also good results for other linear classifiers like Linear Regression and SVM.
As expected the diagonal models perform bad. PCA with m=7 does not increase the performance but it doesn't decrease them either. PCA with m=5 or m=6 decrease the performance, this is due to the fact we're removing too much information, in fact data is correlated but apparently not that much that we can remove 2 or 3 features. Given the limited effectiveness of PCA for generative models, we are going to focus on the whole set of features but still use PCA in the following analysis to be sure. Since we saw that gaussianization performs bad with the Gaussian classifiers, we're going to expect better results on raw features.

Logistic Regression

After looking at the generative approaches, we turn now our attention to the discriminative ones.

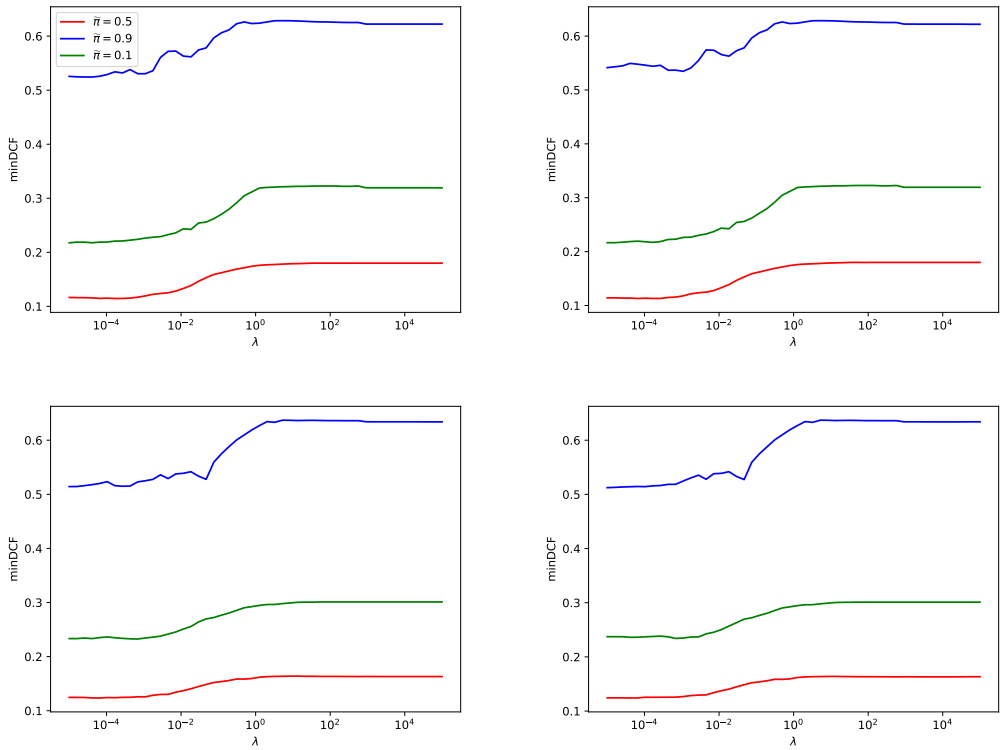

Figure 6: Linear logistic regression, top: Raw data, bottom: Gaussianized data, Left: no PCA, Right: PCA (m=7)

Linear Logistic Regression

We start with **Linear logistic regression**. Since classes are highly unbalanced, we rebalance the cost of the different classes, minimizing

$$J(\boldsymbol{w}, b) = \frac{\lambda}{2}\|\boldsymbol{w}\|^2 + \frac{\pi_T}{n_T}\sum_{i=1|c_i=1}^{n} \log\left(1 + e^{-z_i\left(w^T\boldsymbol{x}_i+b\right)}\right) + \frac{1-\pi_T}{n_F}\sum_{i=1|c_i=0}^{n} \log\left(1 + e^{-z_i\left(w^T\boldsymbol{x}_i+b\right)}\right)$$

We start by considering our main application, so we use $\pi_T = 0.5$ to estimate the value of λ. To do so, K-fold with 5 folds was employed to see how the *minimum DCF* changes against λ, for λ that goes from 10^{-5} to 10^5.
We're still considering PCA to demonstrate it does not improve significantly the performance.
From Figure 6 we can see that a good value for λ is 10^-5.
We can also observe that gaussianization does not help even in this case. However, it was even more expected since logistic regression does not require assumptions on the data distribution, so gaussianization is not useful at all. Nevertheless, is not detrimental either.

	5-fold		
	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$
		Raw features - no PCA	
MVG (Full-Cov)	0.141	0.283	0.68
MVG (Tied Full-Cov)	0.112	0.223	0.562
		Raw features - no PCA	
Log Reg ($\lambda = 10^{-5}, \pi_T = 0.5$)	0.116	0.218	0.526
Log Reg ($\lambda = 10^{-5}, \pi_T = 0.1$)	0.113	0.211	0.558
Log Reg ($\lambda = 10^{-5}, \pi_T = 0.9$)	0.119	0.220	0.515
		Gaussianized features - no PCA	
Log Reg ($\lambda = 10^{-5}, \pi_T = 0.5$)	0.125	0.233	0.514
Log Reg ($\lambda = 10^{-5}, \pi_T = 0.1$)	0.128	0.23	0.522
Log Reg ($\lambda = 10^{-5}, \pi_T = 0.9$)	0.128	0.231	0.516
		Raw features - PCA (m = 7)	
Log Reg ($\lambda = 10^{-5}, \pi_T = 0.5$)	0.114	0.217	0.541
Log Reg ($\lambda = 10^{-5}, \pi_T = 0.1$)	0.112	0.212	0.558
Log Reg ($\lambda = 10^{-5}, \pi_T = 0.9$)	0.116	0.218	0.53
		Gaussianized features - PCA (m = 7)	
Log Reg ($\lambda = 10^{-5}, \pi_T = 0.5$)	0.124	0.237	0.512
Log Reg ($\lambda = 10^{-5}, \pi_T = 0.1$)	0129	0.231	0.521
Log Reg ($\lambda = 10^{-5}, \pi_T = 0.9$)	0.126	0.237	0.508

Table 2: Linear logistic regression - min DCF on the validation set

As we can see from table 2, Results from the raw feature are nearly the same as the MVG with Tied Full Covariance matrix, while it clearly outperforms the MVG Full Covariance. Using different values for π_T does not help significantly in the classification performance for specific application. We can however notice that slightly better results are obtained using $\pi_T = 0.1$, probably because of the class imbalance of the dataset.
As we mentioned, PCA gives very small improvement but, as we said before, it's basically the same performance we obtain using whole dataset.

Quadratic Logistic Regression

Since we only have 8 features in the Raw dataset, we can try applying **Quadratic logistic regression**. In fact the expanded dataset will have $8^2 + 8 = 72$ features, which is still easily manageable.
We look for the best λ, and we find out from Figure 7 that the best value of lambda is, once again, a really small number, so regularization provides once again little to no benefit.

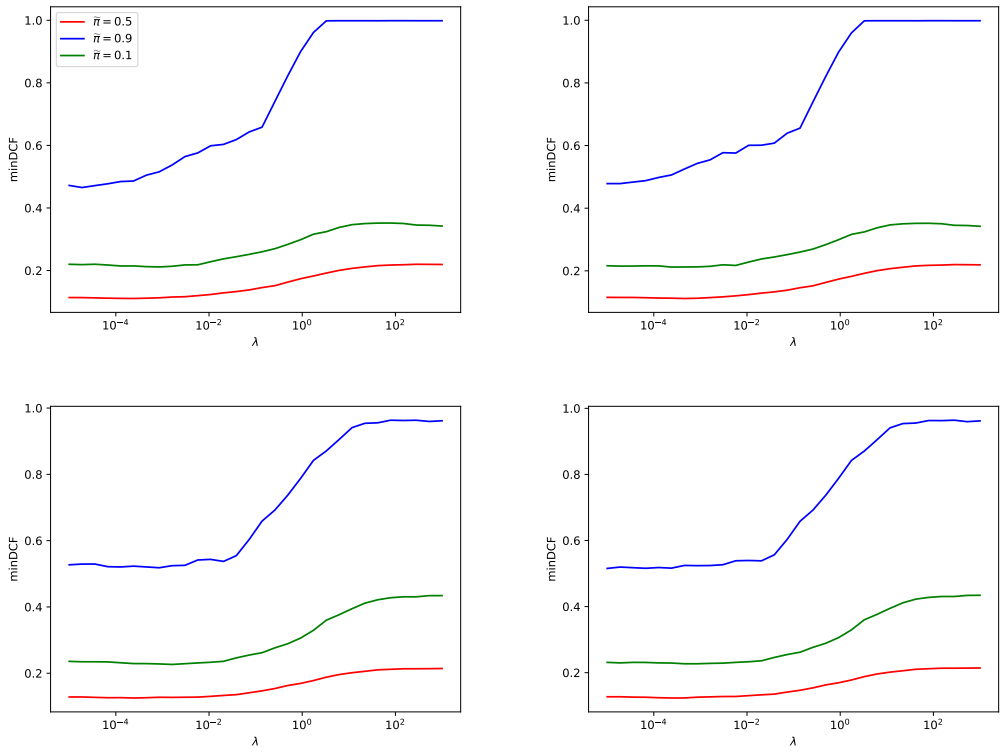

Figure 7: Quadratic logistic regression, top: Raw data, bottom: Gaussianized data, Left: no PCA, Right: PCA (m=7)

	5-fold		
	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$
		Raw features - no PCA	
MVG (Tied Full-Cov)	0.112	0.223	0.562
		Raw features - no PCA	
Quad Log Reg ($\lambda = 10^{-5}, \pi_T = 0.5$)	0.114	0.22	0.472
Quad Log Reg ($\lambda = 10^{-5}, \pi_T = 0.1$)	0.109	0.209	0.494
Quad Log Reg ($\lambda = 10^{-5}, \pi_T = 0.9$)	0.125	0.239	0.521
		Gaussianized features - no PCA	
Quad Log Reg ($\lambda = 10^{-5}, \pi_T = 0.5$)	0.128	0.236	0.527
Quad Log Reg ($\lambda = 10^{-5}, \pi_T = 0.1$)	0.13	0.214	0.534
Quad Log Reg ($\lambda = 10^{-5}, \pi_T = 0.9$)	0-131	0.256	0.534
		Raw features - PCA (m = 7)	
Quad Log Reg ($\lambda = 10^{-5}, \pi_T = 0.5$)	0.115	0.216	0.479
Quad Log Reg ($\lambda = 10^{-5}, \pi_T = 0.1$)	0.111	0.205	0.486
Quad Log Reg ($\lambda = 10^{-5}, \pi_T = 0.9$)	0.124	0.231	0.5
		Raw features - PCA (m = 7)	
Quad Log Reg ($\lambda = 10^{-5}, \pi_T = 0.5$)	0.127	0.231	0.515
Quad Log Reg ($\lambda = 10^{-5}, \pi_T = 0.1$)	0.129	0.213	0.523
Quad Log Reg ($\lambda = 10^{-5}, \pi_T = 0.9$)	0.13	0.25	0.512

Table 3: Quadratic logistic regression - min DCF on the validation set

The results for the minimum DCF using Quadratic logistic regression can be seen in 3. We can observe that it performs nearly the same as linear logistic regression for our main application and $\tilde{\pi} = 0.1$, but performs better for application $\tilde{\pi} = 0.9$.
Unlike the Linear case, here gaussianization is indeed detrimental. PCA provides, once again, basically

the same results as the whole dataset.
From this last analysis we can see that Gaussian assumption may not be the best for this dataset, especially if we consider the last application. We noticed however that linear decision rules and quadratic ones obtained by logistic regression provide good results. That said we can turn our attention to Support Vector Machine and Gaussian Mixture Model.

Support Vector Machine

From now on we're only considering raw and gaussianized feature with no PCA since we showed it does not provide significant help. We expect, from the linear and polynomial SVM good results, similar to the linear regression ones since the separation rules have the same shapes.

Linear Support Vector Machine

The analysis was done on the model that does not balance the two classes, but also on a model that does. To rebalance the classes, a different value of C is used depending on the class.

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

subject to:

$$0 \leq \alpha_i \leq C_i, i = 1, ..., n$$

Where $C_i = C_T$ for samples of class True and $C_i = C_F$ for samples of class False. n is the number of samples, $\mathbf{1}$ is a vector of ones, while $\mathbf{H}$ is a matrix for which:

$$\mathbf{H}_{ij} = z_i z_j x_i^T x_j$$

where $z_i = 1$ if x_i is from class Positive, $z_i = -1$ otherwise.
We select $C_T = C\frac{\pi_T}{\pi_T^{emp}}$ and $C_F = C\frac{\pi_F}{\pi_F^{emp}}$ where π_T^{emp} and π_F^{emp} are the epirical priors for the two classes computed over the training set.
We need to tune the hyper-parameter C. Once again we resort to K-fold cross validation with 5 folds. Figure 8 represent the minDCF against C for C that goes from 10^-3 to 10^0

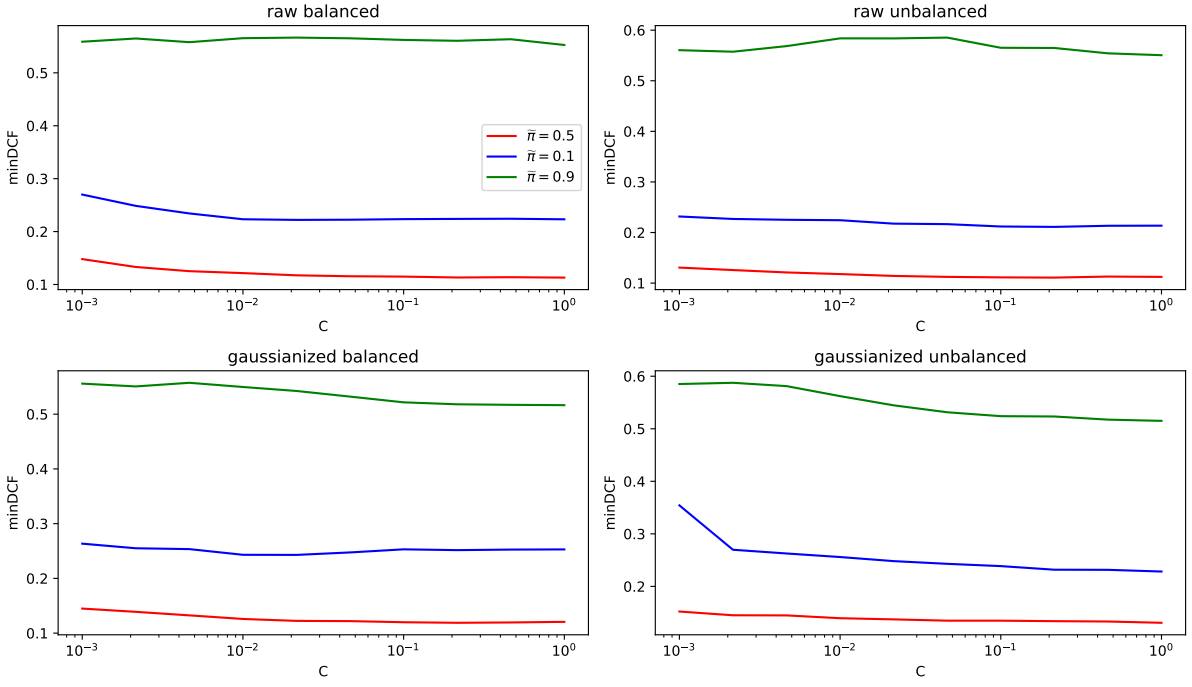

Figure 8: C tuning linear SVM. top: Raw data, bottom: Gaussianized data, left: balanced, right: unbalanced

	5-fold		
	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$
	Raw features - no PCA		
MVG (Tied Full-Cov)	0.112	0.223	0.562
Log Reg ($\lambda = 10^{-5}, \pi_T = 0.5$)	0.116	0.218	0.526
Linear SVM ($C = 0.1$)	0.111	0.212	0.565
Linear SVM ($C = 0.1, \pi_T = 0.5$)	0.115	0.223	0.562
Linear SVM ($C = 0.1, \pi_T = 0.1$)	0.111	0.213	0.566
Linear SVM ($C = 0.1, \pi_T = 0.9$)	0.126	0.234	0.523
	Gaussianized features - no PCA		
MVG (Tied Full-Cov)	0.131	0.235	0.53
Log Reg ($\lambda = 10^{-5}, \pi_T = 0.5$)	0.125	0.233	0.514
Linear SVM ($C = 0.1$)	0.135	0.239	0.524

Table 4: Linear SVM - min DCF on the validation set

As we can observe from Figure 8, regularization is important but after $C = 10^-2$ the choice of C does not look critical. Another thing we can notice is balancing does not influence particularly the choice of C.
Furthermore, a numeric method was used to optimize the function (BFGS Algorithm), and for values bigger than $C = 0.1$ the approximation starts to be imprecise. So $C = 0.1$ was selected and the results in terms of min DCF can be seen in table 4. As we expected linear classifiers are good for our task and the result is similar to the linear Logistic Regression one, except for prior $\tilde{\pi} = 0.9$ for which linear regression performs better.
Once again, training for a specific application improves the results, except for $\pi_T = 0.1$ which gives better results on $\tilde{\pi} = 0.5$ than the $\pi_T = 0.5$, for the reason discussed before.

Polynomial quadratic kernel

For Non-Linear Support Vector Machines, we can employ a **kernel function** that efficiently computes the dot-products in the expanded space:

$$k(x_1, x_2) = \Phi(x_1)^T \Phi(x_2)$$

This allows us to train a SVM in a large dimensional Hilbert space without explicitly computing the mapping.
Since quadratic logistic regression performs good we can also try the **Polynomial quadratic kernel**. The kernel function in this case is:

$$k(x_1, x_2) = (x_1^T x_2 + 1)^2$$

Once again we tune C for both raw and gaussianized features, with and without class balancing. We can see the results in Figure 9

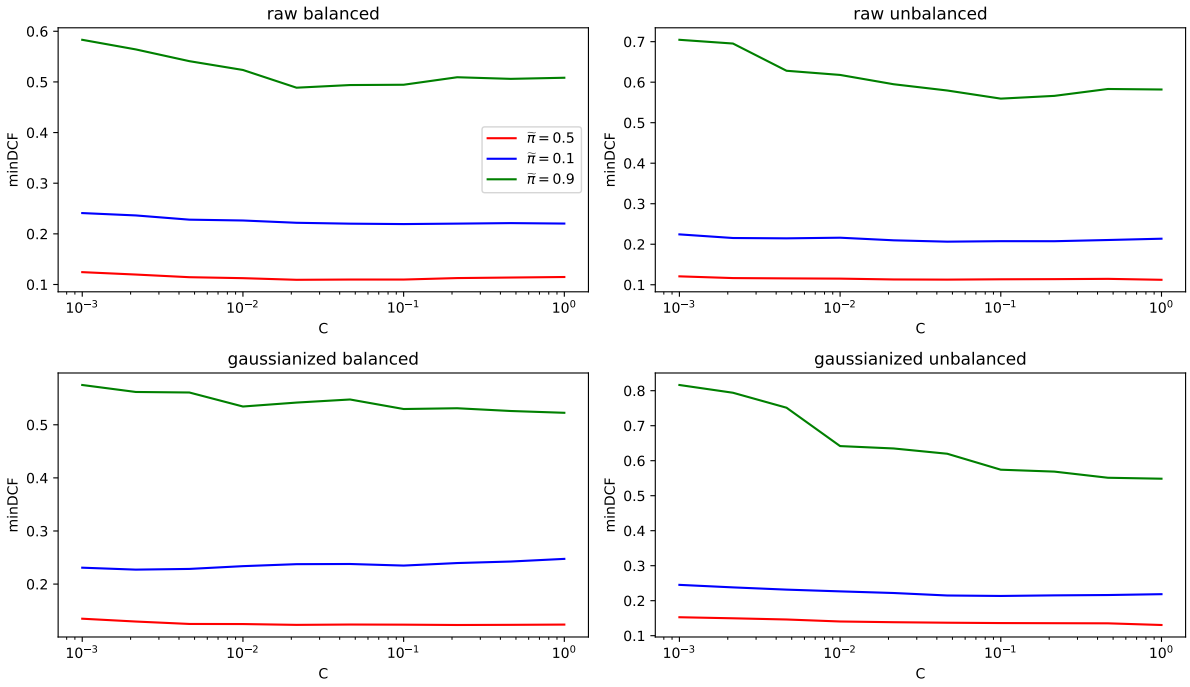

Figure 9: C tuning quadratic SVM. top: Raw data, bottom: Gaussianized data, left: balanced, right: unbalanced

Here the choice of C seems to impact especially for the application $\tilde{\pi} = 0.9$, while it does not really matter on other applications. Looking at the graphs we can say that a wise choice is C = 0.1. The results are shown in Table 5

	5-fold		
	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$
	Raw features - no PCA		
MVG (Tied Full-Cov)	0.112	0.223	0.562
Quad Log Reg ($\lambda = 10^{-5}, \pi_T = 0.5$)	0.114	0.22	0.472
Quad Log Reg ($\lambda = 10^{-5}, \pi_T = 0.1$)	0.109	0.209	0.494
Quadratic SVM ($C = 0.1$)	0.113	0.208	0.56
Quadratic SVM ($C = 0.1, \pi_T = 0.5$)	0.110	0.219	0.494
Quadratic SVM ($C = 0.1, \pi_T = 0.1$)	0.112	0.208	0.553
Quadratic SVM ($C = 0.1, \pi_T = 0.9$)	0.134	0.273	0.493
	Gaussianized features - no PCA		
MVG (Tied Full-Cov)	0.131	0.235	0.53
Quad Log Reg ($\lambda = 10^{-5}, \pi_T = 0.5$)	0.128	0.236	0.527
Quadratic SVM ($C = 0.1$)	0.136	0.214	0.574

Table 5: Quadratic SVM - min DCF on the validation set

We can clearly see that the quadratic SVM performs very similarly to the quadratic logistic regression, even though the latter actually performs slightly better.
Even in this case, like the linear SVM, balancing doesn't seem to impact that much, but training for the specific $\tilde{\pi}$ can help for our secondary applications, but not that much to be relevant.

RBF kernel

We can try another kernel which is the **Radial Basis Function kernel**

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

	5-fold		
	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$
	Raw features - no PCA		
MVG (Tied Full-Cov)	0.112	0.223	0.562
Quadratic Log Reg($\lambda, = 10^-5, \pi_T = 0.1$)	0.109	0.209	0.494
Quadratic SVM ($C = 0.1, \pi_T = 0.5$)	0.110	0.219	0.494
RBF SVM ($C = 10, \gamma = 0.01$)	0.112	0.206	0.801
RBF SVM ($C = 10, \gamma = 0.01, \pi_T = 0.5$)	0.109	0.215	0.516
RBF SVM ($C = 10, \gamma = 0.01, \pi_T = 0.1$)	0.110	0.206	0.509
RBF SVM ($C = 10, \gamma = 0.01, \pi_T = 0.9$)	0.133	0.287	0.514
	Gaussianized features - no PCA		
MVG (Tied Full-Cov)	0.131	0.235	0.53
Quadratic Log Reg ($\lambda = 10^-5, \pi_T = 0.1$)	0.13	0.214	0.534
Quadratic SVM ($C = 0.1$)	0.136	0.214	0.574
RBF SVM($C = 10, \gamma = 0.1$)	0.123	0.210	0.788

Table 6: RBF SVM - min DCF on the validation set

In this case the parameters to tune were two: γ and C.

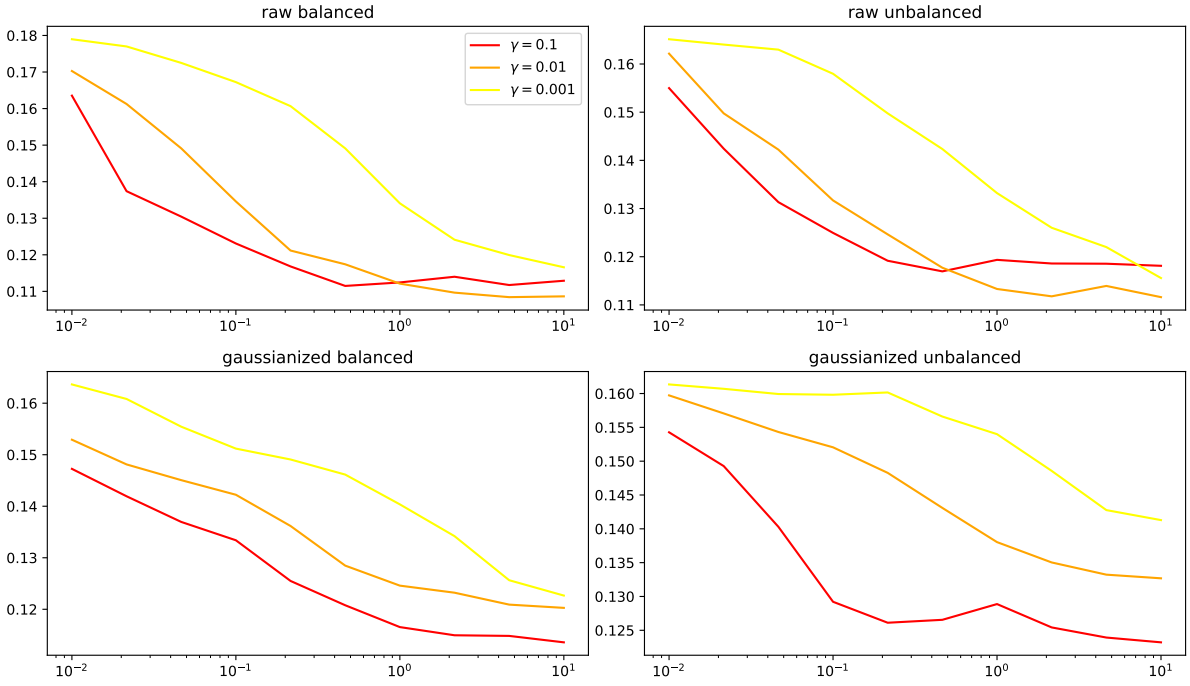

Figure 10: C tuning RBF SVM. top: Raw data, bottom: Gaussianized data, left: balanced, right: unbalanced

In Figure 10 we have the C tuning for application $\tilde{\pi} = 0.5$ for three values of γ. This time we have different results from gaussianized and raw data. So for Raw data $C = 10, \gamma = 0.01$ was selected, while for the gaussianized data we picked $C = 10, \gamma = 0.1$
From table 6 we can get that RBF SVM offers performances that are pretty similar to the polynomial quadratic kernel, but a little bit better (except for $\tilde{\pi} = 0.9$). We can immediatly notice that in this case class balancing is really important in the application $\tilde{\pi} = 0.9$. In fact for the model that does not balance the classes we have a model that, for that application, has a min DCF of 0.801, against the ~ 0.500 of the balanced classes models.
We can also notice that RBF performs like the other models, if not slightly better for our main application and application $\tilde{\pi} = 0.1$ if trained with $\pi_T = 0.1$.

Until now gaussianized features were reported but never really discussed since we assumed and proved that it's detrimental for our dataset.

Gaussian Mixture Models

The last model we consider is a generative approach based on training a **Gaussian Mixture Model** (GMM) over the data of each class. Full covariance, diagonal, tied and tied diagonal models were considered.
Once again a tuning to select the number of Gaussians G was done utilizing the K-fold approach. The model was trained using the **LBG algorithm**. Results are shown in Figure 11

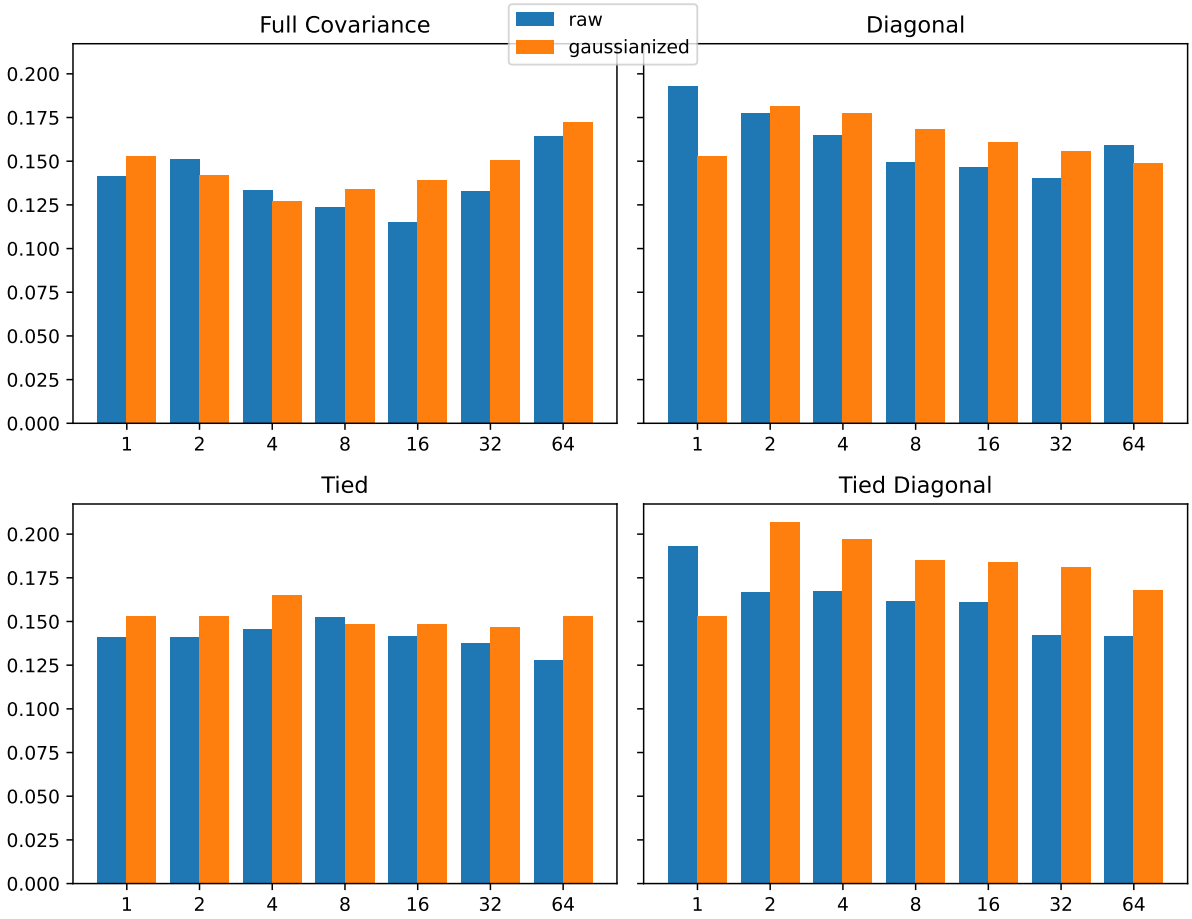

Figure 11: G tuning

We can immediatly observe that results are not particularly different for different G values, and they're not very promising compared to the other models. In the full covariance case, for both the raw and gaussianized features, we can see that with the increase of G, we have some overfitting. However GMM seems to perform worse than the other models (for the values of G that were considered). This could be due to the fact that the data is indeed well-separable in different clusters (otherwise we wouldn't be able to obtain good results), but not that much to outperform the other models without overfitting.
We can for sure say that in this case gaussianization here is way more important since sometimes it gives even better results than the raw features. This is odd since Gaussianization reduces the dynamic range of samples far from the data mean, decreasing the separability of some clusters. The reason is probably the same as before: the separability in clusters of the raw features is not that good.

	5-fold		
	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$
	Raw features - no PCA		
Quadratic Log Reg($\lambda, = 10^-5, \pi_T = 0.1$)	0.109	0.209	0.494
RBF SVM ($C = 10, \gamma = 0.01, \pi_T = 0.5$)	0.109	0.215	0.516
RBF SVM ($C = 10, \gamma = 0.01, \pi_T = 0.1$)	0.110	0.206	0.509
Full cov, 16 Gau	0.115	0.235	0.563
Diagonal cov, 32 Gau	0.140	0.257	0.650
	Gaussianized features - no PCA		
Quadratic Log Reg ($\lambda = 10^-5, \pi_T = 0.1$)	0.13	0.214	0.534
RBF SVM($C = 10, \gamma = 0.1$)	0.123	0.210	0.788
Full cov, 4 Gau	0.139	0.236	0.571

Table 7: GMM - min DCF on the validation set

Since we can see from the histograms that the minDCF values are not particullary good, in Table 7 only the best performing models were reported. We can immediatly see that previous models perform better, especially for our main application.

Model selection

At this point we've seen that using raw features provides better results over the gaussianized one. Furthermore, almost all the models we tried offer similar results in performance. We focus on our main application but we also want good results even for other applications, so in the end two models trained on raw features were selected:

- Quadratic Logistic regression ($\lambda = 10^{-5}, \pi_T = 0.1$)
- RBF SVM ($C = 10, \gamma = 0.01, \pi_T = 0.5$)

To see which one is better for different applications we can use a **ROC Plot** (Figure 12)

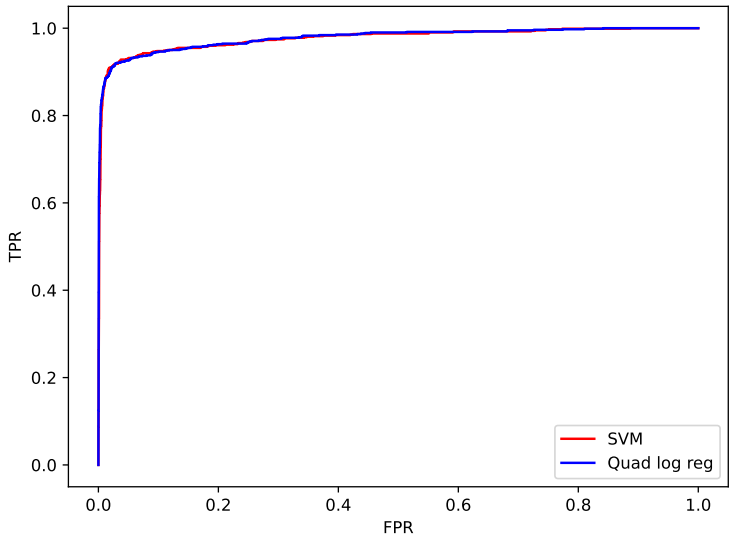

Figure 12: ROC plot RBF and Quadratic Logistic Regression models

As expected the two models perform basically the same, with each being slightly better than the other for different applications.
RBF SVM should have more complex decision rules, so the fact that it results similar to the quadratic logistic regression can be explained by how data is distributed, and cannot be separated further, or it

	$\tilde{\pi} = 0.5$		$\tilde{\pi} = 0.1$		$\tilde{\pi} = 0.9$	
	min DCF	act DCF	min DCF	act DCF	min DCF	act DCF
SVM	0.109	0.110	0.215	0.364	0.516	0.876
QLR	0.109	0.109	0.209	0.215	0.494	0.520

Table 8: Minimum and actual DCF for Quadratic Logistic Regression model (QLR) and SVM RBF (SVM)

could have bad calibrated scores. Moreover, logistic regression is more prone to overfitting than the SVM. Thus we select RBF SVM as candidate model and Quadratic Logistic regressiona as secondary system.

Score calibration

Up to now only minumum DCF was considered, which measures the cost we would pay if we made optimal decisions for the evaluation set using the recognizer scores. The cost that we actually pay, however. depends on the goooodness of the decisions we make using those scores, we therefore turn our attention to actual DCF.
The optimal threshold that optimizes the Bayes risk is $t = -\log \frac{\tilde{\pi}}{1-\tilde{\pi}}$, so to form decisions from scores we either assume scores are calibrated and use the theoretical threshold, or we can estimate a good threshold for the target application.
In Table 8, minimum and actual DCF for our candidate models are reported.
We can immediately see that, for our considered applications, while Quadratic logistic regression provides almost calibrated scores, SVM RBF does only for the main one.
We can clearly see it in the Bayes error plot in Figure 13. In fact while for our main application both models present little to no calibration loss, this becomes very relevant for other applications for the SVM model.
The QLR model, however, starts having some calibration loss only for high values of $\tilde{\pi}$.

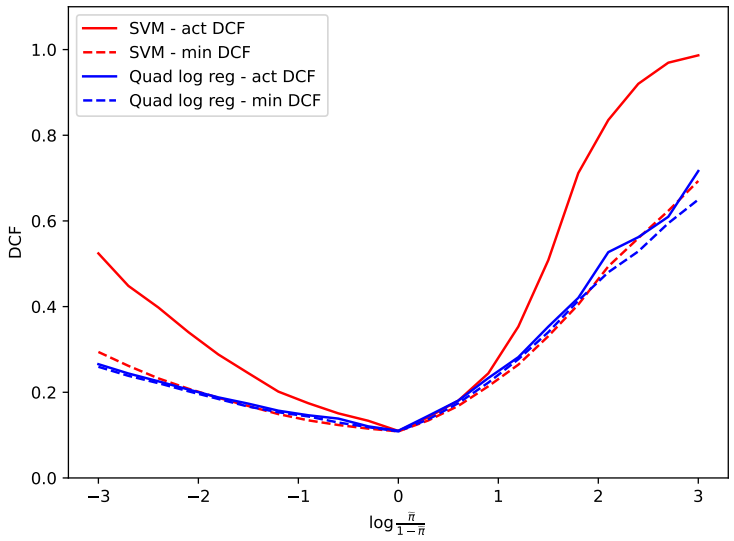

Figure 13: Bayes error plot

There are two solutions to this problem, and afterwards both are reported. The first consists in estimating an optimal threshold for a given application, which means taking the threshold corresponding to the minimum DCF on the validation set. This approach has the advantage that is simple, but does not produce well calibrated scores, meaning that, for different applications, the threshold needs to be estimated again. Taken the scores obtained using K-fold cross validation, they were split in two

partitions, on the first one the threshold was estimated, on the other minimum DCF and actual DCF were calculated and compared. The results are in Table 9 where t^* is the estimated threshold.

	min DCF	act DCF $t = -\log \frac{\tilde{\pi}}{1-\tilde{\pi}}$	act DCF t^*
		$\tilde{\pi} = 0.5$	
SVM	0.111	0.113	0.117
QLR	0.111	0.111	0.13
		$\tilde{\pi} = 0.1$	
SVM	0.215	0.368	0.226
QLR	0.203	0.203	0.225
		$\tilde{\pi} = 0.9$	
SVM	0.478	0.888	0.514
QLR	0.467	0.467	0.516

Table 9: minimum and actual DCF for different thresholds

We can immediately see an improvement for the SVM classifier, the one that produced uncalibrated score. However, we have worst results for the QLR model, since the scores it produces are already well calibrated, the threshold we obtained is optimal for the first split (the one we obtained the threshold from), but not on the validation split.
A second approach consists in re-calibrating the the scores so that the theoretical threshold provides close to optimal values over a wide range of effective priors. So what we want to do is to find a function that, given a score, returns the calibrated one. Furthermore this function should be monotone since we want to preserve the fact that higher non-calibrated scores favor class Positive ($\mathcal{H}_T$), while non calibrated ones favor class Negative ($\mathcal{H}_F$). The discriminative score model is used so we assume f an affine function

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

That should produce well calibrated score, so it can be interpreted as:

$$f(s) = \log \frac{f_{S|C}\left(s \mid \mathcal{H}_T\right)}{f_{S|C}\left(s \mid \mathcal{H}_F\right)} = \alpha s + \beta$$

So the class posterior probability for prior $\tilde{\pi}$ is:

$$\log \frac{P\left(C = \mathcal{H}_T \mid s\right)}{P\left(C = \mathcal{H}_F \mid s\right)} = \alpha s + \beta + \log \frac{\tilde{\pi}}{1-\tilde{\pi}}$$

Interpreting scores as features and letting $\beta' = \beta + \log \frac{\tilde{\pi}}{1-\tilde{\pi}}$ we have the log posterior ratio of the logistic regression model. So we can use the prior-weighted Logistic Regression model to learn the model parameters α and β'.
The same split as before was utilized and we can see the results in Tables 10 and 11. This method provide clear improvement for the SVM but not for the QLR, because, as we said, scores were already calibrated.

	min DCF ($\tilde{\pi} = 0.5$)	min DCF ($\tilde{\pi} = 0.1$)	min DCF ($\tilde{\pi} = 0.9$)
	0.109	0.203	0.467
	act DCF ($\tilde{\pi} = 0.5$)	act DCF ($\tilde{\pi} = 0.1$)	act DCF ($\tilde{\pi} = 0.9$)
Uncalibrated	0.113	0.215	0.505
QLR $\tilde{\pi} = 0.5$	0.118	0.216	0.51
QLR $\tilde{\pi} = 0.1$	0.117	0.209	0.5
QLR $\tilde{\pi} = 0.9$	0.116	0.228	0.483

Table 10: prior-weighted logistic regression calibration for the QLR classifier

	min DCF ($\tilde{\pi} = 0.5$)	min DCF ($\tilde{\pi} = 0.1$)	min DCF ($\tilde{\pi} = 0.9$)
	0.11	0.215	0.478
	act DCF ($\tilde{\pi} = 0.5$)	act DCF ($\tilde{\pi} = 0.1$)	act DCF ($\tilde{\pi} = 0.9$)
Uncalibrated	0.113	0.368	0.888
SVM $\tilde{\pi} = 0.5$	0.115	0.215	0.501
SVM $\tilde{\pi} = 0.1$	0.115	0.22	0.502
SVM $\tilde{\pi} = 0.9$	0.114	0.215	0.495

Table 11: prior-weighted logistic regression calibration for the SVM RBF classifier

This second method actually gives better results than the first one.
Even though we're optimizing for a specific application $\tilde{\pi}$, this method provides good calibration for a wide range of applications, and this can be seen by the bayes error plots in Figure 14

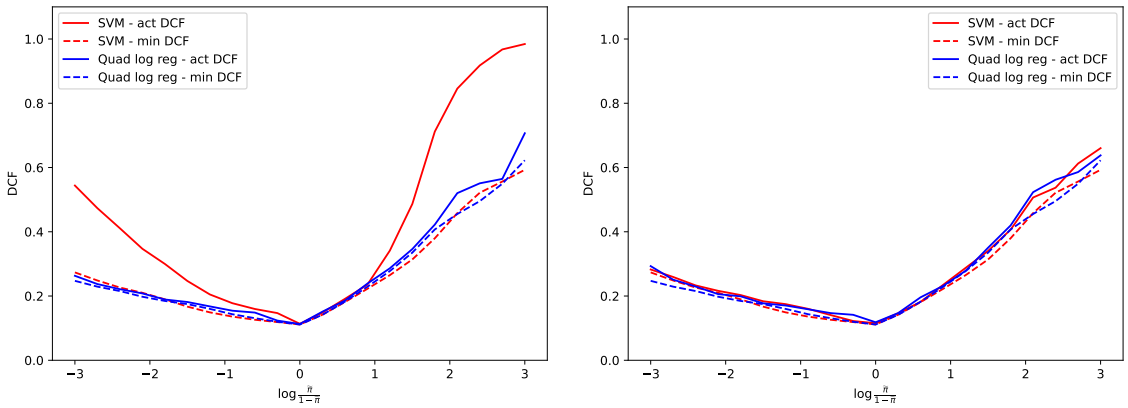

Figure 14: Bayes error plots SVM and QLR, left: uncalibrated, right: calibrated scores with $\tilde{\pi} = 0.5$

Fusion model

Looking at the results obtained so far, the models agree on almost all decisions, but no all, because they are based on different approaches. The decisions of the two classifiers can be combined to try to obtain greater performance. However we don't expect to obtain much better results for the observations we did before, which were that since most classifiers had similar performance, there are points in the intersection of the two classes that are hard to classify. So the points that the classifiers agree on are probably the same. Furthermore the performance is already very good. However a fusion approach was tried to demonstrate it.
The approach that was used is a **score-level** fusion, where a function of the scores of the two classifiers gives a new score (the fusion model one). This new score is used to perform the decision.
A simple linear form for f was used:

$$f(s_{t,A}, s_{t,B}) = \alpha_A s_{t,A} + \alpha_B s_{t,B} + \beta$$

Again we can use prior-weighted logistic regression to train the model parameters, moreover using this method allows us to get calibrated scores as we've seen before. Once again the two split method as before was used. Training was done using one of two partitions of the scores, while the results refer to the other partition. As can be seen from Figure 15, the fusion score does not provide much better results than the two models we considered. However from Figure 16, we can see that the fusion model provides well calibrated scores, except for high values of $\tilde{\pi}$.
From Table 12 is clear that the Fusion model helps but very little for our considered applications.

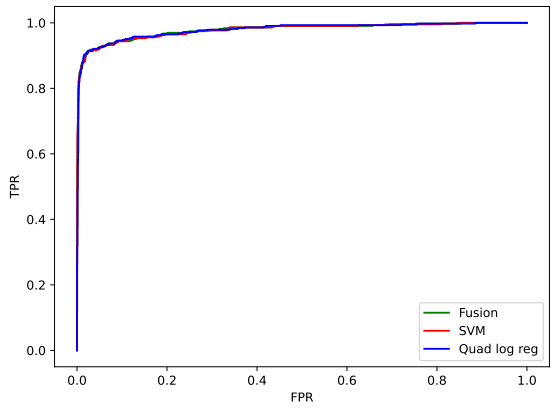

Figure 15: ROC plots of the two models and the fused model

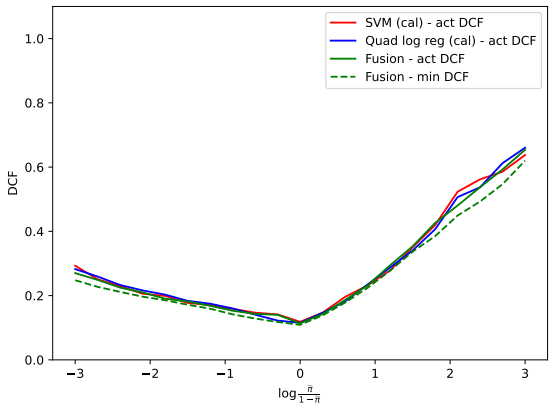

Figure 16: Bayes error plots of the two models and the fused model

	$\tilde{\pi} = 0.5$		$\tilde{\pi} = 0.1$		$\tilde{\pi} = 0.9$	
	min DCF	act DCF	min DCF	act DCF	min DCF	act DCF
SVM	0.11	0.115	0.215	0.215	0.478	0.501
QLR	0.109	0.118	0.203	0.216	0.467	0.51
Fusion	0.109	0.114	0.201	0.216	0.465	0.487

Table 12: DCFs for the fused system

Concluding, since SVM and QLR both perform similarly, and the fusion helps very little, the final model can be either of them. Since the performance seem to be slightly in favor to the fused system, the latter can be chosen as final model.

Experimental Results

We now need to assess the quality of our models trained on the whole training set, on the evaluation set. We analyze the choices that were made and see how the models perform on unseen data.

Gaussian classifiers

Once again we use as a measure of performance the minimum DCF, to verify if the proposed solution is the one that can achieve the best accuracy.

	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$
		Raw features - no PCA	
Full-Cov	0.14	0.283	0.646
Diag-Cov	0.185	0.33	0.621
Tied Full-Cov	0.11	0.207	0.591
Tied Diag-Cov	0.152	0.262	0.544
		Gaussianized features - no PCA	
Full-Cov	0.151	0.241	0.622
Diag-Cov	0.154	0.285	0.574
Tied Full-Cov	0.125	0.221	0.54
Tied Diag-Cov	0.158	0.301	0.568
		Raw features - PCA (m=7)	
Full-Cov	0.139	0.293	0.574
Diag-Cov	0.201	0.514	0.755
Tied Full-Cov	0.11	0.208	0.587
Tied Diag-Cov	0.141	0.257	0.556
		Gaussianized features - PCA (m=7)	
Full-Cov	0.151	0.246	0.64
Diag-Cov	0.153	0.25	0.622
Tied Full-Cov	0.126	0.229	0.523
Tied Diag-Cov	0.128	0.242	0.518
		Raw features - PCA (m=6)	
Full-Cov	0.146	0.277	0.6
Diag-Cov	0.218	0.544	0.725
Tied Full-Cov	0.133	0.244	0.593
Tied Diag-Cov	0.159	0.293	0.593
		Gaussianized features - PCA (m=6)	
Full-Cov	0.147	0.249	0.628
Diag-Cov	0.15	0.251	0.594
Tied Full-Cov	0.129	0.241	0.519
Tied Diag-Cov	0.135	0.251	0.535
		Raw features - PCA (m=5)	
Full-Cov	0.152	0.249	0.632
Diag-Cov	0.211	0.508	0.707
Tied Full-Cov	0.143	0.253	0.596
Tied Diag-Cov	0.163	0.31	0.591
		Gaussianized features - PCA (m=5)	
Full-Cov	0.149	0.255	0.626
Diag-Cov	0.148	0.254	0.587
Tied Full-Cov	0.129	0.241	0.523
Tied Diag-Cov	0.135	0.251	0.539

Table 13: MVG Classifiers - min DCF on the evaluation set

We can see that the results (Table 13) are consistent with those obtained on the validation set. In particular, we can notice that even here PCA (m=7) does not help, while PCA with m $<$ 7 can

be detrimental. The Tied model over Raw features is confirmed to be the best option for our main application.

Linear Logistic Regression

Now we focus on linear regression and we can clearly see from Table 14, again, that results are consistent. However here PCA performs slightly better but still not significant, so it's in line with our training phase.

	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$
		Raw features - no PCA	
Log Reg ($\pi_T = 0.5$)	0.107	0.199	0.542
Log Reg ($\pi_T = 0.1$)	0.11	0.199	0.531
Log Reg ($\pi_T = 0.9$)	0.115	0.201	0.51
		Gaussianized features - no PCA	
Log Reg ($\pi_T = 0.5$)	0.122	0.215	0.485
Log Reg ($\pi_T = 0.1$)	0.125	0.21	0.504
Log Reg ($\pi_T = 0.9$)	0.123	0.212	0.482
		Raw features - PCA (m=7)	
Log Reg ($\pi_T = 0.5$)	0.106	0.201	0.541
Log Reg ($\pi_T = 0.1$)	0.109	0.199	0.532
Log Reg ($\pi_T = 0.9$)	0.113	0.203	0.531
		Gaussianized features - PCA (m=7)	
Log Reg ($\pi_T = 0.5$)	0.119	0.217	0.481
Log Reg ($\pi_T = 0.1$)	0.121	0.209	0.484
Log Reg ($\pi_T = 0.9$)	0.121	0.217	0.469

Table 14: Linear logistic regression - min DCF on the evaluation set

In Figure 17 we confirm that the chosen λ was the one that provides close to optimal results. In fact the curves show the same trend (not quite for $\tilde{\pi} = 0.9$, but the choice remains the same).

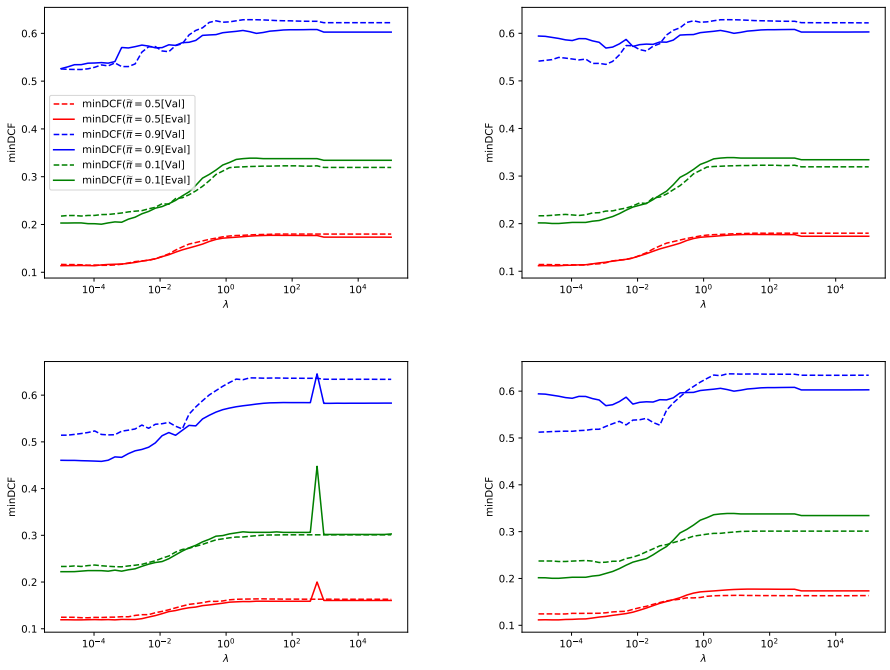

Figure 17: Linear logistic regression, top: Raw data, bottom: Gaussianized data, Left: no PCA, Right: PCA (m=7)

Quadratic Logistic Regression

The same analysis was done on Quadratic Logistic Regression, and the results shown in Table 15 show congruous results, and once again we confirm that Quadratic Logistic Regression trained with $\pi_T = 0.1$ on raw features give the best results so far even on the evaluation set.

	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$
		Raw features - no PCA	
Quad Log Reg ($\pi_T = 0.5$)	0.108	0.207	0.424
Quad Log Reg ($\pi_T = 0.1$)	0.1	0.206	0.451
Quad Log Reg ($\pi_T = 0.9$)	0.113	0.213	0.443
		Gaussianized features - no PCA	
Quad Log Reg ($\pi_T = 0.5$)	0.117	0.216	0.481
Quad Log Reg ($\pi_T = 0.1$)	0.117	0.204	0.465
Quad Log Reg ($\pi_T = 0.9$)	0.119	0.238	0.485
		Raw features - PCA (m=7)	
Quad Log Reg ($\pi_T = 0.5$)	0.106	0.207	0.441
Quad Log Reg ($\pi_T = 0.1$)	0.101	0.205	0.465
Quad Log Reg ($\pi_T = 0.9$)	0.112	0.212	0.44
		Gaussianized features - PCA (m=7)	
Quad Log Reg ($\pi_T = 0.5$)	0.117	0.214	0.499
Quad Log Reg ($\pi_T = 0.1$)	0.116	0.203	0.475
Quad Log Reg ($\pi_T = 0.9$)	0.118	0.228	0.506

Table 15: Quadratic logistic regression - min DCF on the evaluation set

Once more, from Figure 18 we can see that the choice of λ was right for our main application. For application $\tilde{\pi} = 0.9$ better results could have been obtained with a slightly larger value of λ.

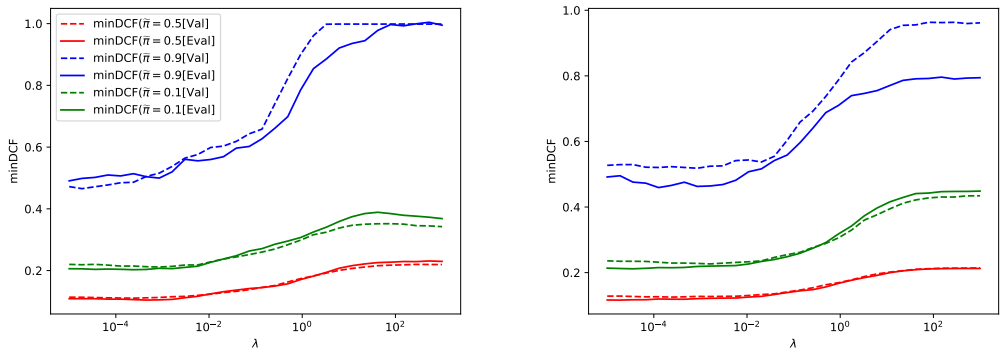

Figure 18: Quadratic logistic regression, left: Raw data, right: gaussianized data

Linear and Quadratic kernel SVM

Even though the chosen model is the RBF kernel SVM, we can check also the linear and quadratic results, respectively in Tables 16 and 17.

	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$
		Raw features	
Linear SVM (Unbalanced)	0.112	0.205	0.552
Linear SVM ($\pi_T = 0.5$)	0.11	0.203	0.567
Linear SVM ($\pi_T = 0.1$)	0.112	0.203	0.55
Linear SVM ($\pi_T = 0.9$)	0.119	0.215	0.508
		Gaussianized features	
Linear SVM (Unbalanced)	0.129	0.222	0.499

Table 16: Linear SVM - min DCF on the evaluation set

	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$
		Raw features	
Quadratic SVM (Unbalanced)	0.102	0.204	0.522
Quadratic SVM ($\pi_T = 0.5$)	0.098	0.205	0.451
Quadratic SVM ($\pi_T = 0.1$)	0.103	0.201	0.516
Quadratic SVM ($\pi_T = 0.9$)	0.123	0.259	0.475
		Gaussianized features	
Quadratic SVM (Unbalanced)	0.125	0.209	0.541

Table 17: Quadratic SVM - min DCF on the evaluation set

For the linear case results are slightly to the ones obtained earlier, probably due to the fact that we re-trained with the whole training dataset. However for the quadratic SVM results are clearly better, and even better than QLR. Since the decision rules are similar this could've been expected. To verify if our choice of C was good, we repeat the tuning on the unbalanced model. In Figures 19 and 20 we can see that for our main application the curves follow the same trend, and so our choice of C was close to optimal. Exception done for $\tilde{\pi} = 0.9$ on the raw dataset for which we have some better values, but not so much that results critical.

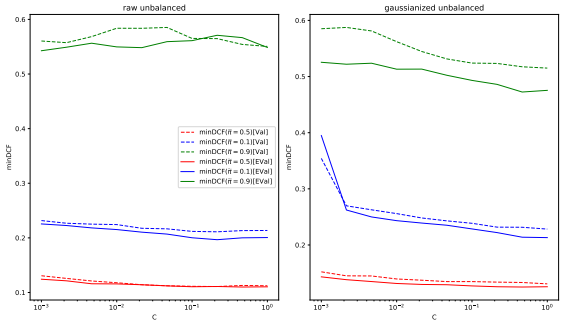

Figure 19: Linear SVM

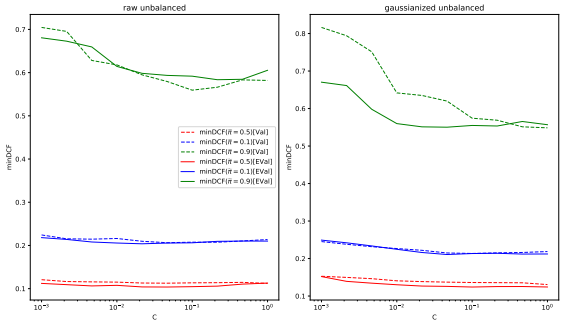

Figure 20: Quadratic kernel SVM

RBF kernel SVM

Focusing now on the model of our primary system. From Figure 21 we immediatly see that our choice of C was not optimal for the evaluation set, in fact $C = 1$ and $\gamma = 0.1$ give better performance on the raw dataset. For the guassianized one however the choice was good. Still, the choice doesn't seem too much critical.

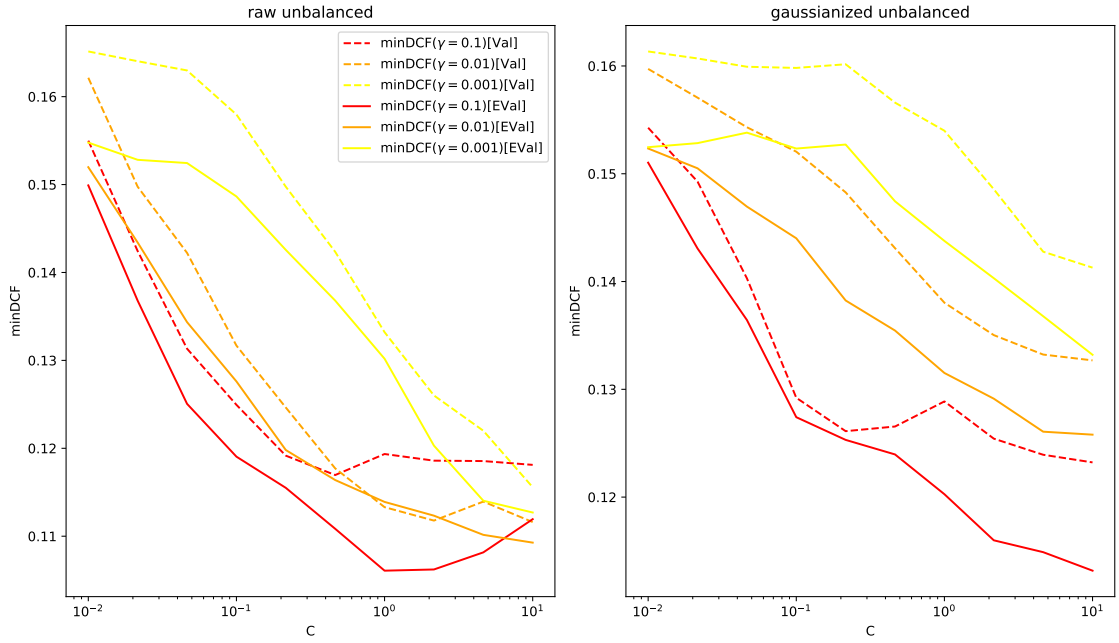

Figure 21: C tuning RBF SVM

The minimum DCF results can be seen in Table 18, where we can see that balanced RBF with $\pi_T = 0.5$ on the raw features is the best for our main application. Results are pretty similar to the quadratic SVM, but with a proper choice of C and γ we could've obtained better.

	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$
		Raw features	
RBF SVM (Unbalanced)	0.105	0.201	0.458
RBF SVM ($\pi_T = 0.5$)	0.098	0.2	0.478
RBF SVM ($\pi_T = 0.1$)	0.105	0.201	0.452
RBF SVM ($\pi_T = 0.9$)	0.131	0.298	0.458
		Gaussianized features	
RBF SVM (Unbalanced)	0.111	0.208	0.710

Table 18: RBF SVM - min DCF on the evaluation set

Gaussian Mixture Models

Even though GMM didn't provide very good performance, we can still try it to see if results are consistent. From tuning G we can see, in Figure 22, that for raw features full covariance, G=8 was a slightly better choice, not critical though. Overall the choice obtained by tuning was good because we have consistent results.

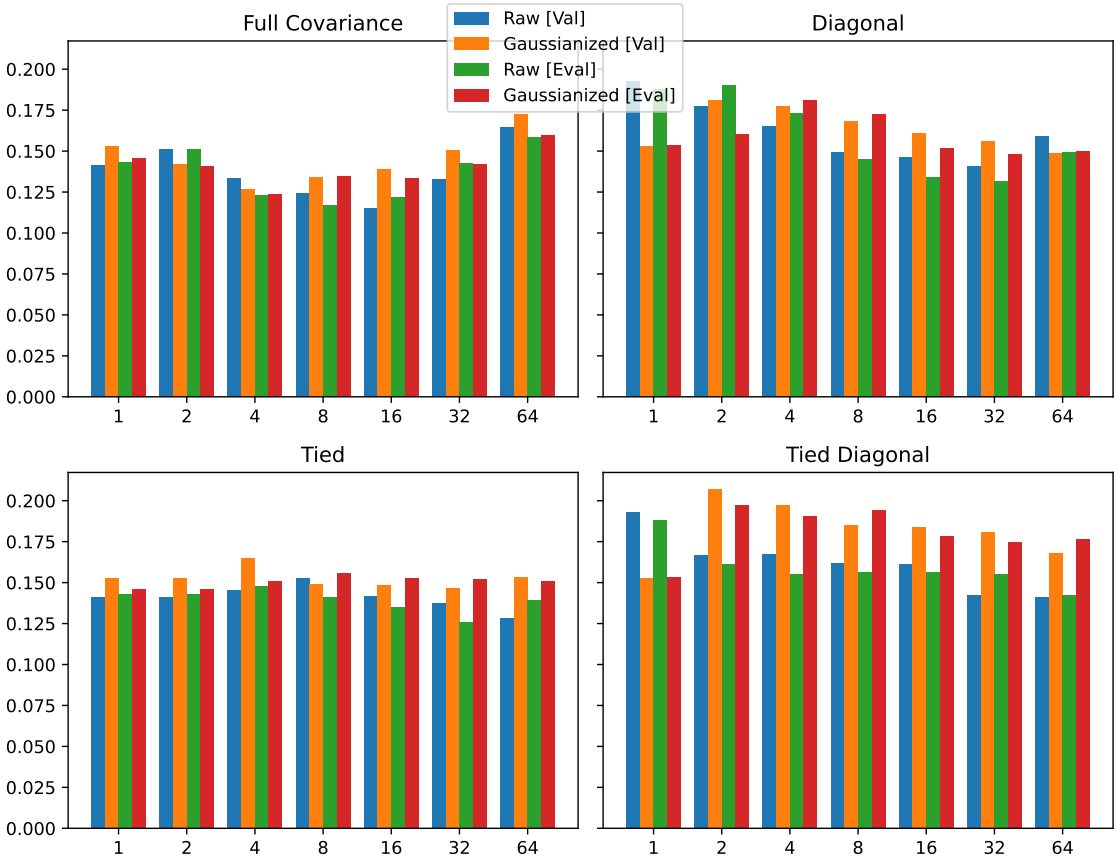

Figure 22: G tuning GMM

In Table 19 we can see that the outcomes are in line with what we obtain before. So we confirm that GMM is not a good choice for the data we have.

	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$
Raw Full-Cov (G=16)	0.123	0.23	0.525
Raw Diagonal (G=32)	0.148	0.238	0.781
Gaussianized Full-Cov (G=4)	0.119	0.244	0.553

Table 19: GMM - min DCF on the evaluation set

Fused System

Finally we evaluate the fusion system, for which we can see from Table 20 that gives very similar performance to the other two models we have chosen, also for the evaluation set.

	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$
RBF SVM	0.098	0.2	0.478
QLR	0.1	0.206	0.451
Fusion	0.099	0.197	0.453

Table 20: Fusion and the top two models - min DCF

These results are coherent with what we've seen before, and we can also see that the performance is basically the same from the ROC plot in Figure 23

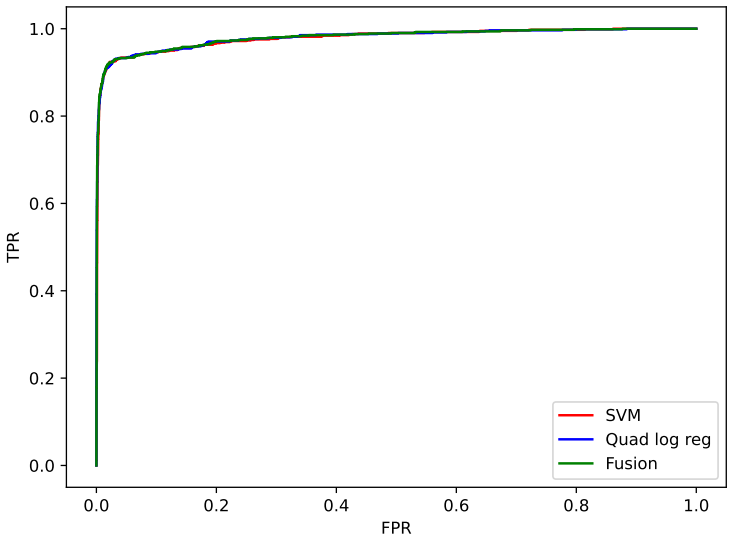

Figure 23: ROC plots

Actual DCF and calibration

Up to now we analyzed the systems in term of minimum DCF, however in practice we don't know the optimal rule for the evaluation data. We want to see how good are the decisions that we're actually able to make using the recognizer scores, that's why we now consider the actual DCF. Before, two methods were utilized to find a good threshold, and both were good. Now we apply those methods on the evaluation to see if that holds.

	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$
		min DCF	
SVM	0.098	0.2	0.477
QLR	0.101	0.204	0.459
		actual DCF $t = -\log \frac{\tilde{\pi}}{1-\tilde{\pi}}$	
SVM	0.104	0.362	0.841
QLR	0.105	0.21	0.475
		actual DCF $t = t^*$	
SVM	0.1	0.206	0.495
QLR	0.104	0.209	0.489

Table 21: minimum and actual DCF for different thresholds on the evaluation set

As we can observe from Table 21, the thresholds calculated on the training set are good also on the evaluation set.

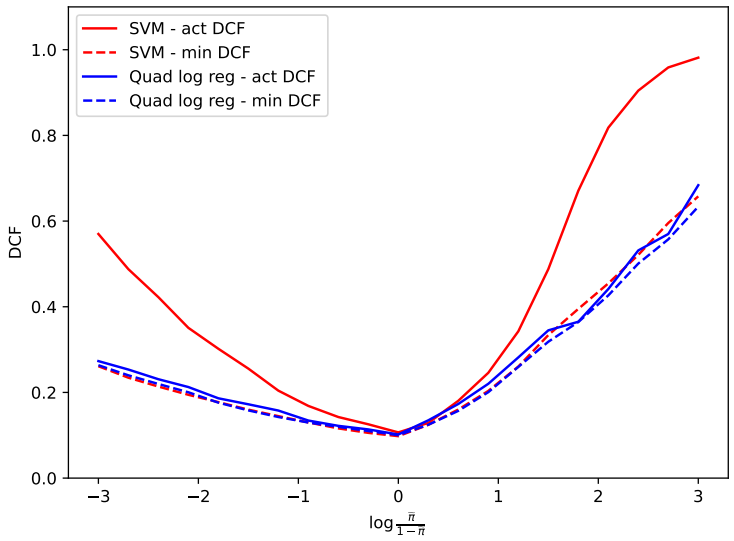

Figure 24: Bayes error plots for the SVM and QLR models on the evaluation data

However, as we said, this method requires recalculating t^* for each application, so we considered the second method that produces calibrated scores. From Figure 24, we can clearly see that while the QLR still produces well-calibrated scores, the SVM model does not. So we apply the second approach on the scores and we obtain the bayes error plots in Figure 25 that calibration still works well.

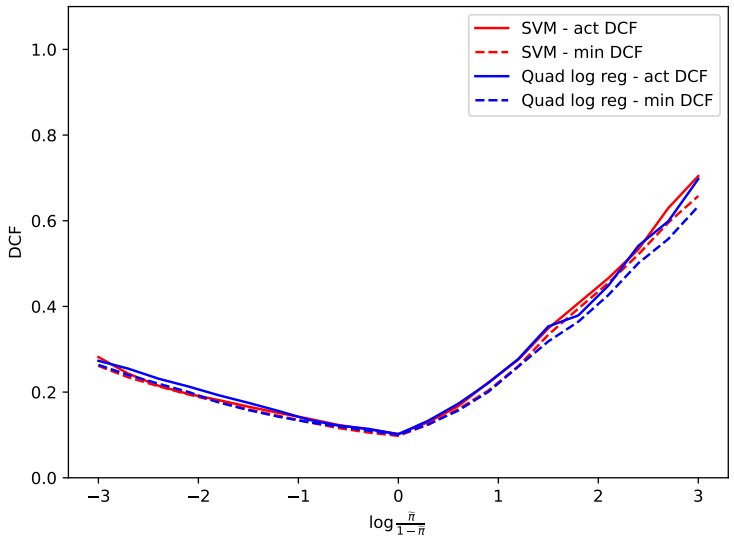

Figure 25: Bayes error plots for the SVM and QLR models on the evaluation data, calibrated

Finally we can see if the fusion scores we computed are, again, calibrated and compare them with the top two models. In Figure 26, we can observe that that's the case, however for large values of $\tilde{\pi}$ it performs slightly worse, while on the training set we observed the opposite.

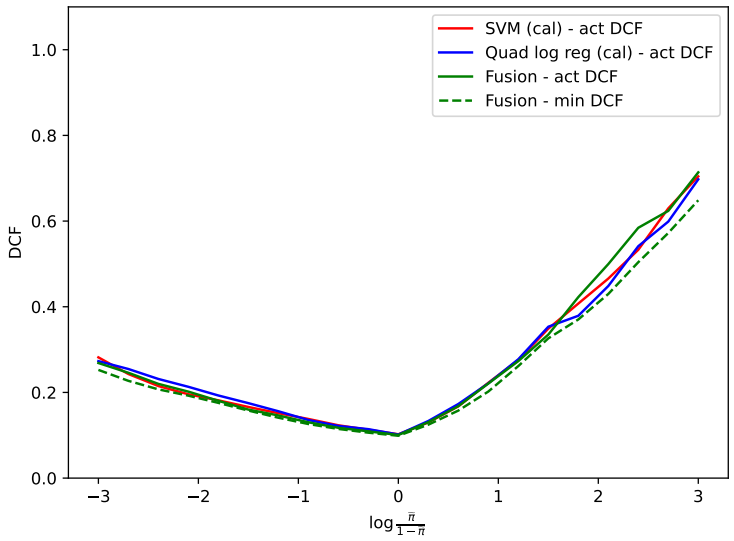

Figure 26: Bayes error plots for the SVM, QLR and Fusion models on the evaluation data, calibrated

From Table 12 we can see that, as we've seen in the training set, the performance is pretty much the same, but SVM and QLR outperform the Fusion model for the last application.

	$\tilde{\pi} = 0.5$		$\tilde{\pi} = 0.1$		$\tilde{\pi} = 0.9$	
	min DCF	act DCF	min DCF	act DCF	min DCF	act DCF
SVM (calibrated)	0.098	0.105	0.2	0.202	0.477	0.489
QLR (calibrated)	0.101	0.104	0.204	0.216	0.459	0.485
Fusion	0.099	0.101	0.197	0.208	0.453	0.526

Table 22: minimum and actual DCF for calibrated scores and Fusion

Conclusions

Concluding, we have seen that raw data performs better than gaussianized data for all the models that were tried. Moreover, PCA with m = 7 does not help, while for values m < 7 starts to be detrimental. We've observed that MVG models do not perform relatively well (except for the Tied-Cov one), and the GMM models cannot compete with LR and SVM. In particular, from the latter, we managed to see that linear classification rules are good, but more complex rules are better.
Our final choice of the fusion model was good for the main application $\tilde{\pi} = 0.5$ but it was not the best one in terms of actual DCF for the other two (especially $\tilde{\pi} = 0.9$). In the end we were able to achieve a DCF cost of ~ 0.1 for $\tilde{\pi} = 0.5$, ~ 0.2 for $\tilde{\pi} = 0.1$, and ~ 0.5 for $\tilde{\pi} = 0.9$).
Overall we obtained very good classification results and the similarity between validation and evaluation results show that the evaluation population is similar to the training population.
In the end the choices that were made on the training and validation sets proved to be effective on the evaluation data.

References

[1] R. J. Lyon, B. W. Stappers, S. Cooper, J. M. Brooke, and J. D. Knowles. Fifty years of pulsar candidate selection: From simple filters to a new principled real-time classification approach. *Monthly Notices of the Royal Astronomical Society*, 2016. https://doi.org/10.1093/mnras/stw656.